Developing Learning Methods for Non-stationary and Imbalanced Data Streams

Manal Abdullah N Almuammar

School of Computer Science and Electronic Engineering

University of Essex

A thesis submitted for the degree of

Doctor of Philosophy in Computer Science

April 2020

Abstract

Recent developments in technology have enhanced the abilities of systems to both generate and collect data from a variety of sources. There is an increasing number of Internet of Things devices generating continuous data streams rapidly. Mining these data streams brings new opportunities but also introduces new challenges. Learning from these data streams is challenging due to the characteristics of such streams: continuous unbounded high-speed data of an evolving nature which must be processed on the fly. An additional challenge emanates from the fact that many of the data streams generated by real-world applications are imbalanced within themselves. This difficulty is more acute in multi-class learning tasks. Despite "learning from non-stationary streams" and "class imbalance" problems having been investigated separately in the literature, too little attention has been paid to the multi-class imbalance problem as it can emerge in evolving streams. This thesis is devoted to the development of new techniques for mining evolving data streams which have skewed distributions and to tackling the multi-class problem related to such streams. It presents a new method for classifying heterogeneous data streams which extends the current concept drift adaptation techniques so they can deal with imbalanced classes' scenarios. To this end, an adaptive learning algorithm is developed which uses a windows based approach, and which modifies the make-up of the training set to enhance the accuracy of classification. In addition, this research proposes a new method for discovering patterns from evolving data streams with skewed distributions; it introduces a dynamically calculated support threshold; this allows the proposed method to tackle the rare patterns problem as this is encountered in non-stationary streams. Moreover, an experiment is conducted in relation to forecasting time series from heterogeneous data streams using a deep learning approach, to provide real-time parking prediction in a transportation domain.

To my wonderful children, Talal, Latin, and Meshaal,
who always make me proud.

Declaration

I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where stated otherwise by reference, the work presented is entirely my own.

Acknowledgements

I would like to express my sincere thanks to my supervisor, Prof. Maria Fasli for her invaluable guidance and constant support throughout my years of PhD study. Without her professional guidance, this work wouldn't have been achieved.

I am grateful to King Saud University for granting me the opportunity to do my PhD research, and to the Saudi cultural bureau in London. Great thanks to the School of Computer Science and Electronic Engineering at University of Essex for supporting my research, special thanks go to Emma McClelland and Claire Harvey. I would like to thank board members, Dr.Sam Steel, Dr.Spyros Samothrakis and Dr.Javier Andreu for their valuable discussions and useful comments. Great thanks to Prof.Peter McBurney and Dr.Ana Matran-Fernandez for examining my thesis, and for their valuable comments during the viva.

I would like to extend my thanks to the brilliant teachers at the Unity Primary Academy for their love and caring. I feel reassured that when I drop off my kids at school every morning, they have a fun day of learning.

I owe a debt of gratitude to my parents, Abdullah and Sarah, for simply everything; for their love at every step of my life, for giving me the strength to pursue my dreams. I am indebted to Muna, Amani, and Amjad who have constantly encouraged and empowered me, especially when I was distressed and hopeless. Special thanks to Nasser, Khaled, and Muhammad for listening to me, helping me to feel able, and hopeful right when I needed. I am grateful for Nouf, Hamad, Abdulaziz, Nada and Sarah for their love and support.

This journey would never have been possible without the dedicated support and encouragement of my loving husband, Abdullah Alzaid, who thoughtfully followed every step of my research. I am deeply grateful to him for helping me reach my goals and for supporting me every time I make new ones, for loving me, and for always having my back. No words would be enough to express my feelings towards my kids for believing in me and for their patient understanding of my constant work. I am forever grateful to have them in my life. Together we faced many challenges, travelled many miles, shared many stories and the PhD journey was a pleasant experience.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

We live in the era of data abundance. Sensors, financial markets, search engines, social media applications, text messages, YouTube, etc. have generated unprecedented amount of data. In 2018, an article on Forbes [191] noted that there were 2.5 quintillion bytes of data produced daily, and that 90% of the data in the world was generated between 2016 and 2018. The amount of data generated is growing at an increasing rate with the growth of the Internet of Things (IoT) [242].

The IoT is a network of physical devices, vehicles, buildings and other items [276]. These objects have embedded within them sensors, software, and network connectivity technology which together enable them to measure changes in their environment and generate data that reports on their status [276, 287]. According to IoT Analytics report [1], it was estimated that there were seven billion connected IoT devices at the end of 2018, this number exceeded the number of people on the planet. The IoT technology enhances the capabilities available for both generating and collecting data, and provides the means to develop applications that can create a true reflection of the real world and so support better decision making [242]. Qin et al. in [216] identified the main characteristics of data streams generated from IoT objects: dynamic; heterogeneous; generated at high speeds; and usually containing redundant, incomplete and/or uncertain data.

[1] `https://iot-analytics.com/iot-2018-in-review/`

Data streams, from the IoT devices and other sources (e.g, financial markets), can be considered as one of the main sources of what is referred to as Big Data [242]. Big Data is commonly used to describe huge amounts of heterogeneous data generated from a variety of sources and which can be analysed in order to reveal insights from these data streams [184, 281]. A study from Gartner [25] described Big Data as having three dimensions, known as three 'Vs': volume "the amount of data generated", variety "the different forms of data: structured (e.g., relational databases), semi structured (e.g., JSON documents) and unstructured data (e.g., emails messages)", and velocity "the speed at which data generated" [25, 150, 154, 281]. Later, the veracity "uncertainty of data" was considered as the fourth dimension [190].

Data mining and knowledge discovery have been attracting a significant amount of research effort recently, across both academia and industry [4, 63, 257, 286]. Data mining can be defined as the process of extracting implicit, previously unknown, and potentially useful information from data [286]. It provides a means by which to develop solutions that bridge the widening gap between data generation and data analysis — by providing techniques and algorithms which facilitate the analysing of vast amounts of data. Such analysis has been used to generate useful insights, for example, detecting air quality levels [89, 99, 152], monitoring traffic flows in urban areas [11, 212, 223, 255], and in healthcare industry to monitor emergency cases or detect diseases in their early stages [20, 56, 186, 218].

1.1 Motivation

Motivated by the increasing proliferation of IoT technologies affecting our daily lives, this thesis seeks to exploit the kind of real-time data streams which flow from IoT devices, and to illustrate the significance of heterogeneous data streams from diverse sources. In the IoT data stream model, data arrives continuously and at high speed[185, 220, 242]. Thus, a huge amount of data can be constantly recorded

and used for real-time and/or historical analysis [185, 242]. Deriving insights from the IoT data streams can be considered as one of the key opportunities; real-time or near-real time analysis of such heterogeneous data streams has the potential to have a major impact across a number of different domains [111, 220, 242]. However, dealing with the changing nature of such data streams over time (i.e., with concepts that drift or change completely) and the imbalanced distributions that may appear in these streams, is one of the main challenges in IoT data streams mining [74, 242].

In this thesis, we are interested in developing and extending techniques for mining evolving data streams [2] which have skewed distributions and tackling the multi-class problem related to such streams. In particular, we seek to develop learning techniques for non-stationary and imbalanced data streams from a variety of IoT devices in order to provide near-real time analysis of such heterogeneous data streams. This kind of analysis is being used in this thesis to provide efficient solutions to car parking issues, in a transportation domain (i.e., this is the underlying domain that we are considering from an experimental point of view). In this use case of intelligent parking system, where the parking lots are monitored by a variety of IoT devices, data streams come in too fast for processing (i.e., data velocity) from different IoT devices (i.e., data variety), and we develop learning techniques which are able to extract useful information from these uncertain data, learn behavioural patterns from various kinds of streams, and provide predictions of future behaviours, in order to support intelligent decision making.

1.2 Research Aim and Objectives

The aim of this thesis is to investigate and enhance machine learning techniques focused on mining data streams generated from non-stationary environments, and to explore the problems that can occur when mining and analysing such hetero-

[2] An evolving data stream is a stream of data whose distribution changes over time [27, 28, 46, 119, 224].

geneous data streams from different sources. The thesis' objectives are as follows:

- To identify the usefulness of integrating a variety of data sources together and analysing the resultant heterogeneous data streams.
- To develop a simulated environment of a parking lots setting (i.e., cars and their movements), where the parking lots are monitored by a set of embedded IoT devices; these devices generate heterogeneous data streams. The main purpose of this simulation is to create a dataset (i.e., data streams from the IoT devices) that captures the dynamic behaviour of objects (i.e., cars) in relation to a real-world application. This in order to study the problems that can be encountered when mining non-stationary and imbalanced datasets, and then to use this dataset to experiment with and develop new techniques to address these problems and validate the methods.
- To develop new learning techniques that are capable of classifying non stationary data streams, detecting and adapting to changes in the underlying concepts being represented in these streams; and dealing with imbalanced class distributions.
- To investigate and enhance methods for pattern discovery from dynamic streaming data so that they can be used to capture the relations between different items in dynamic environments, and handle changes in the patterns which emerge from mining the various types of stream.
- To explore time series forecasting techniques for multivariate time series from heterogeneous data streams in order to identify potential challenges when dealing with data from non-stationary environments which, in addition, exhibit complex seasonality.
- To investigate a deep learning approach to forecasting time series from data streams in order to provide as accurate forecasts as possible of future be-

haviour — so to support intelligent decision making in the parking lots environment.

The work described in this thesis addresses the joint problem of the changing nature (i.e., concept drift) and the imbalanced classes distributions in non-stationary data streams. It seeks to contribute to the wider literature by extending the existing techniques for adapting concept drift [45, 47, 88, 108, 166, 247, 267, 295] into class imbalance scenarios [79, 137, 243, 271, 274] and developing methods that learn from such dynamic data streams and tackle the multi-class imbalance issue in these streams. In addition, it enhances the current pattern discovery techniques over data stream [132, 141, 142, 170, 176, 258], in order to provide an efficent solution to the rare patterns problem [132, 141, 142, 179] that may appear in non-stationary streams and to detect changes (i.e., drifts) in the emerging patterns.

Moreover, this thesis attempts to deliver insights from the heterogeneous data streams generated by the IoT devices. It addresses the combined issue of the multiple variables and the complex seasonality in time series data streams generated at high speed [40, 41, 144]. Furthermore, it develops a deep learning approach for forecasting time series, over multiple time steps, in order to provide reliable car parking solutions, unlike the existing parking availability prediction systems [50, 219, 265, 293, 294] that require additional hardware and sensors to be installed for predicting the parking availability.

1.3 Contributions

Extracting insights from data streams, captured in real-time, can give decision makers a heightened awareness of real-time events [11, 12, 55, 212]. For example, streams from ubiquitous sensors which are deployed in infrastructure such as roads and buildings, or sensors which report on environmental conditions can be used to develop innovative solutions to traffic managements systems [223, 255]. This

requires a set of powerful tools and techniques for processing and reasoning these streams — with minimal latency so that real-time, or near-real-time responses can be generated [4, 100].

This present study has contributed to the development of data streams mining techniques for non-stationary and imbalanced data streams. It has investigated mining techniques which are designed to integrate and learn from data streams generated by different sources. In addition, it has addressed the most common challenges which are encountered when analysing such data streams. In particular, this work investigated the following challenges:

- Investigating classification algorithms which operate over data streams and addressing the weaknesses in the existing approaches to classification problems related to multi-class situations [79, 137, 243, 271, 274], and then developing a new method capable of classifying dynamic data streams which have skewed distributions.

- Investigating pattern discovery approaches and examining the rare pattern problem [132, 141, 142, 179] which is often encountered in real-world data stream applications.

- Investigating machine learning methods which can be used to forecast multivariate time series [40, 41, 144] derived from a variety of data streams.

These challenges were investigated through creating a simulation environment which has been inspired by real data. We have chosen the domain of transportation (i.e., car parking setting) as the domain of application because there is an increasing number of sensors and other IoT devices that are deployed in parking lots settings [234], wherein there are dynamic data streams from different sources which can readily be used in an investigation of mining techniques operating over data streams [148].

One difficulty encountered at the start of this research, was that of finding a real-dataset which exhibited the presence of heterogeneous data streams derived

from a variety of sources. IoT devices can potentially have a significant positive impact on people's quality of life [71, 256]. However, these devices are also pose serious threats to privacy as they can collect/reveal personal data such as user identity, location, car registration numbers, credit card information, etc. Therefore, datasets derived from these devices are not generally made publicly available [256]. Thus we decided instead to simulate a dynamic environment within which a set of IoT devices have been deployed — in its infrastructure. This simulation provided the means by which to generate the necessary heterogeneous data streams, having skewed distributions and evolving over time (Chapter 3).

Streaming data poses many challenges relating to limited storage and processing resources, as well as those relating to the need to adapt with evolving data [17, 87, 102]; moreover, it can exacerbate issues such as class imbalance [272, 273, 290]. This work has explored mining techniques which can classify the data items within evolving streams which have skewed class distributions. Class imbalance issues are even more intractable when more than two classes are involved [197, 272]. Most studies [46, 48, 57, 137, 178, 198, 271–274, 290] in the field have used ensemble learners to handle the concept drift in imbalanced data streams, furthermore, these works have only focused on the two classes situation and too little attention has been paid to multi-class imbalance problems [274]. Therefore, the present study aimed to contribute to this area by extending concept drift adapting techniques into multi-class imbalance scenarios. In particular, we have developed a new method (named ICE-Stream: Imbalanced Classes in Evolving Streams) for learning from evolving data streams which can handle the problem of class imbalance as it is encountered in these kinds of stream; this method is based on an adaptive learning algorithm which uses a windows based approach (instead of ensemble learners) in order to handle the concept drift in data streams (Chapter 4).

Pattern discovery has become a powerful tool for extracting valuable information from mass data [3, 100, 123]. The current trend towards applications

which generate massive data streams has heightened the need for pattern discovery techniques which can extract patterns from, and reveal hidden relations, in these data streams [3]. However, extracting patterns from evolving data streams is difficult and time consuming [177]. In particular, there are two classic problems which are closely linked to pattern discovery from data streams [155, 177, 179]: the extraction of a huge number of meaningless patterns, and/or the omission of patterns which do not appear frequently (i.e., rare pattern problem). This study investigated pattern discovery techniques which operate over data streams, and highlighted the need for better methods to capture the dynamic of patterns from heterogeneous streams efficiently. Then, it produced a new method (named FP-EStream: Frequent Patterns from imbalanced Evolving Streams) for extracting patterns from evolving streams with skewed classes distributions, which can efficiently handle the issue of concept drift as encountered in patterns that emerge from such streams and overcome the rare pattern problem — by using a dynamically calculated threshold to identify patterns contained within heterogeneous streams (Chapter 5).

Time series data from dynamic environments can capture the dynamic behaviours of these environments [144], and provide a means by which to monitor and predict the changes which occur in them over time [40, 41]. The explosive growth in the complexity and size of time series data streams, across many real-world applications, has presented a challenge to the existing statistical forecasting techniques [131, 184]. Thus, there is a definite need to develop further time series forecasting techniques that deal with data generated at high speed, and which exhibit complex seasonality and multiple variables [131, 144]. This work examined machine learning approaches to the forecasting of time series from heterogeneous data streams. In particular, it provides an empirical exploration of the use of a deep learning approach (using Gated Recurrent Units) to forecasting time series from heterogeneous data streams. Unlike the existing statistical methods [41, 42, 76, 77, 144] that are not suitable for dealing with multivariate time se-

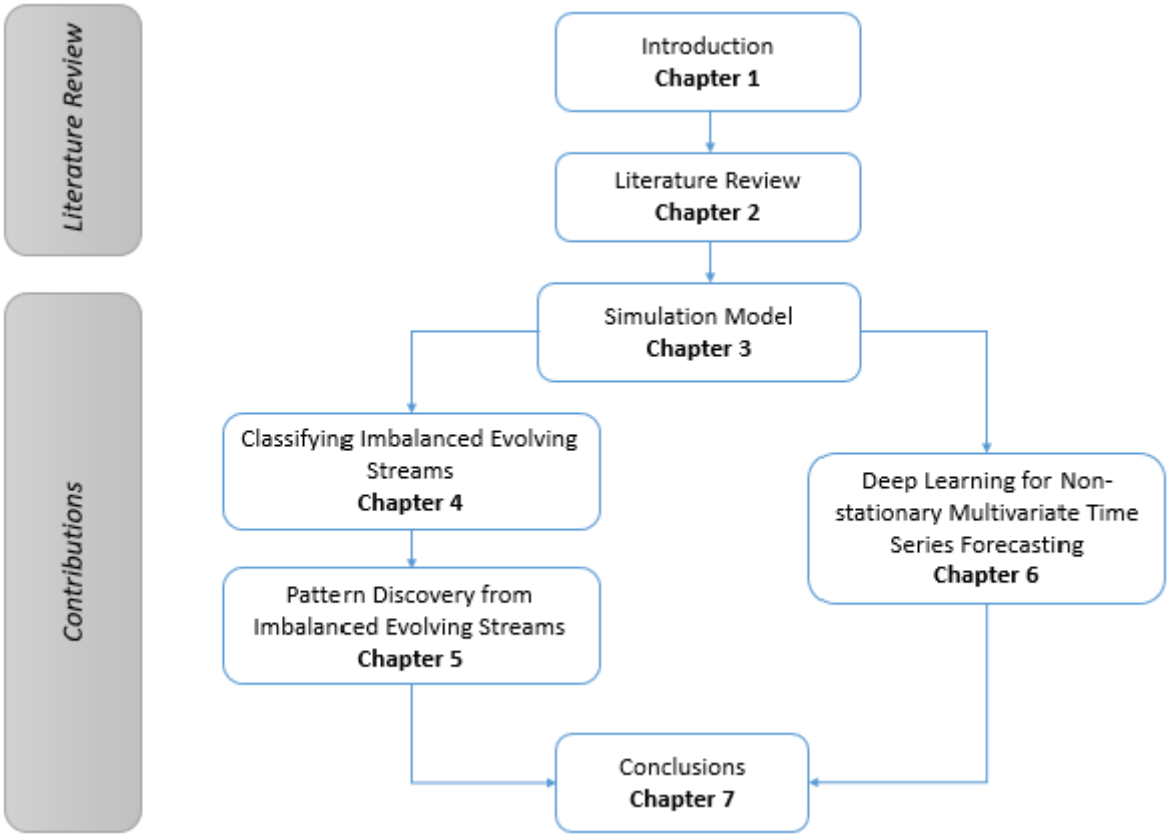


Figure 1.1: Thesis structure

ries with multiple seasonal components, the proposed approach can handle the complex seasonality in the multivariate time series data streams (Chapter 6).

1.4 Thesis Structure

The thesis structure is depicted in Figure 1.1. The overall structure of this thesis takes the form of seven chapters, including this introductory chapter; the rest of the thesis is structured as follows:

- Chapter 2 aims to lay out the theoretical dimensions of the research, it presents a concise description of key concepts which relate to this work and defines the scope of the thesis. In particular, it discusses the existing literature associated with the research area, focusing on data streams and stream mining techniques, and illustrating the difference between the mining of conventional data and the mining of data streams. In addition, Chapter

2 elucidates common challenges encountered when mining streaming data, namely: (i) concept drift, (ii) imbalanced class distributions, and (iii) mining frequent patterns. This will be followed by a discussion of approaches that have been used in the literature to solve the aforementioned problems, and identify the limitations with these approaches which then lead to the techniques proposed in the subsequent chapters. Furthermore, the chapter illustrates key concepts that relate to time series forecasting, including a review of current statistical techniques as well as of the most promising studies of machine learning approaches that have been undertaken in the literature as regards forecasting time series. Moreover, this chapter presents general background information concerning deep learning using artificial neural networks and some neural network architectures that will be used for this present study in particular.

- Chapter 3 defines the research problem using an illustrative scenario of a variety of data streams derived from a dynamic environment; and it presents a detailed description of the main components of the research framework used here. Moreover, this chapter provides the experimental framework, discusses the reasons behind the choice of simulation model and explains the criteria which have been used to create this model. Furthermore, it provides a detailed description of the characteristics of the sample that was used to create the simulation, and how this sample has been modelled in a simulation environment.

- Chapter 4 is concerned with investigating classification techniques on evolving data streams which exhibit internal imbalance, and discusses the methods used for this in here. It begins by identifying some common problems that may occur when performing classification over heterogeneous data streams derived from dynamic environments, and highlights the limitations of previous studies. Then, it suggests a novel technique for learning from

imbalanced evolving data streams. After that, it moves on to describe in greater detail the experiments that were conducted on a simulated dataset and two other, real, datasets. This will be followed by a comparison between the results obtained using the proposed methods with the results obtained by the use of other learning approaches; this is in order to evaluate the proposed technique.

- Chapter 5 is focused on discovering patterns from differing data streams with skewed distributions, and which emerge from dynamic environments. The chapter begins by providing a brief overview about pattern mining in general. Then it goes on to describe the methods used in this investigation. In particular, the chapter proposes a new method for extracting pattern from imbalanced evolving data streams. This discussion is followed by a description of a set of experiments performed on both the simulated dataset which was used for the previous chapter and a real dataset. Subsequently, the chapter discusses the findings which emerged from these experiments, then the results are compared with state-of-the-art pattern mining techniques.

- Chapter 6 provides an empirical exploration of deep learning in relation to the task of forecasting non-stationary multivariate time series. In particular, it investigates using deep learning approaches to the forecasting of the availability of parking spaces across a number of different parking lots. It begins by providing a brief overview of the related work as regards predicting the availability of parking spaces. Then it describes the problem using an illustrative scenario and discusses the methods used in this exploration. It goes on to describe in greater detail how the experiments were conducted and presents a discussion of the findings.

- Chapter 7 is the final chapter of this thesis; it draws upon the discussions presented across the entire thesis, tying up the various theoretical and empirical strands. The chapter provides a brief summary of the main contri-

butions of this work, and a critique of its findings, indicating the strengths and weaknesses of the approaches taken. It then goes on to identify areas for further research.

1.5 Publications

In the course of conducting this research, the following publications were produced:

1. Poster presentation, 2017 CSEE Joint Workshop with Industry, Public and Charitable Sector, Friday 30 June 2017, Colchester Campus.

2. Almuammar, M. and Fasli, M., "Pattern Discovery from Dynamic Data Streams using Frequent Pattern Mining with Multi-support Thresholds," in *the 2017 IEEE International Conference on Frontiers and Advances in Data Science (FADS)*, Xi'an, October 2017. pp. 35-40.[8]

3. Almuammar, M. and Fasli, M., "Learning Patterns from Imbalanced Evolving Data Streams," in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, December 2018. pp. 2048-2057. [9]

4. Almuammar, M. and Fasli, M., "Deep Learning for Non-stationary Multivariate Time Series Forecasting, in *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, December 2019. pp.2097-2106 [10]

Chapter 2

Background and Literature Survey

2.1 Introduction

Recent developments in technology have enhanced our capability for collecting data from diverse sources which produce data continuously. Deriving insights from such data streams is challenging, because data items arrive at high speed and algorithms that process these streams must deal with the changing nature of such streams, using limited computational resources — in terms of memory and time [80, 107]. An additional challenge emanates from the fact that many of data streams generated by many real-life applications have skewed distributions [46, 197, 273], this poses a difficulty for learning algorithms, as they will be biased towards the majority group [165]. Although "learning from evolving streams" [80, 107, 108] and "learning from imbalanced distributions" [178] have been investigated separately in the literature, only few works [46, 165] have addressed the combined challenge of imbalanced distribuations in the context of data streams, and too little attention has been paid to the multi-class imbalance problem as it can emerge in evolving streams [273].

This chapter begins by giving a brief overview of data stream and data mining, describes characteristics of streaming data and its potential usefulness, and discusses the requirements of data streams mining techniques. It then goes on to review the literature on some common challenges in mining data streams, namely concept drift, imbalanced classes distributions, and mining frequent patterns. Moreover, it looks at time series forecasting, and examines some previous studies which were conducted for forecasting time series. Moreover, it provides an overview of deep learning in artificial neural networks, and then gives a detailed descriptions of some neural network architectures: Multi-layer Perceptron and Recurrent Neural Networks (including the Gated Recurrent Units layer variant). The chapter concludes with a short summary that identifies the main challenges related to this work, in order to assist the reader to understand the purpose of this thesis.

2.2 Data Streams

Recent years have seen significant advancements in information technology, and this has led to the creation of large flows of data. An increasing number of applications involved in everyday life generate a considerable amount of data on a continuous basis. Examples include the stock market, sensor networks, information from social networks or geo-spatial services, customer click streams, etc. These datasets which grow rapidly and constantly are referred to as *data streams* [5, 100, 117, 143].

A data stream is defined as an unbounded and ordered sequence, ordered implicitly by arrival time or explicitly by time stamp, of data items [17, 117]. There are a variety of streaming data scenarios; for example, streams may be a flow of data items that arrive in a continuous regular manner, or it may consist of huge amounts of data items of potentially infinite length which arrive in unpredictable bursts — such as streaming data about player interactions within online games

[97, 210, 241]. Another common scenario is where data objects' values change frequently and at high speed, e.g., stock market values [139]. A data stream can also be generated continuously by multiple data sources which typically send out their data records simultaneously in small packages — this description applies to video or sensor streams [102].

Data streams are different from data stored in conventional relational databases in a number of respects. Gama et al. [102] summarized the differences between streaming and conventional models in terms of two issues. First, there is no control over the arrival of data stream elements because data elements from the stream arrive online. Second, the complexity of storing and retrieving data stream, rather than conventional, elements is far greater; the data stream is potentially of infinite size; therefore, once an element from a data stream has been processed, it is usually discarded and thus cannot be retrieved easily (unless, in fact, it has been specifically stored away). Similarly, a number of studies [17, 246] have identified important features of data streams to distinguish them from data managed using traditional data models: (i) ordered, (ii) infinite, (iii) arrive at high speed, (iv) dynamic, and (v) high dimensional.

2.3 Data Mining

Data mining, also referred to as knowledge discovery, emerged during the late 1980s [126]. It is a broad umbrella term that describes variety of processes which are carried out on large datasets in order to discover previously unknown patterns and to identify correlations and relationships among data [4, 100, 126, 127, 286]. The results of data mining may be used then to predict future trends and/or support decision making [4, 100]. Witten and Frank in [285] defined data mining as "the extraction of implicit, previously unknown, and potentially useful information from data". Later, Han et al. in [126] defined data mining as "the process of discovering interesting patterns and knowledge from large amount of data".

Typically, data mining is considered one of the steps in the Knowledge Discovery in Databases (KDD) process [91] and its aim is to provide tools to enable the transformation of vast amounts of (often messy) data into useful information and knowledge [286]. Most of techniques described in KDD have been developed within a field known as machine learning[96, 286]. Machine learning emanated from artificial intelligence [51] and its focus is on developing techniques for automatically learning from data in order to improve the performance of a system [51, 286]. Machine learning provides the technical basis for data mining, it presents algorithms for inferring structure from data [91, 286], but data mining is more than select a machine learning technique and apply it on a dataset [91], hence there are additional challenges in many real world applications related to data (e.g., incomplete or dynamic data) or/and the appropriate choice of the learning technique parameters [126, 286]. In this thesis, the term of data mining techniques is used to refer to data mining that utilize machine learning techniques.

From the literature [4, 35, 126, 183, 199, 286], it can be seen that there are a number of different types of learning paradigms associated with machine learning and data mining, namely:

- Supervised Learning (SL), in this approach, the learning model is trained on a labelled dataset where the task is to learn the mapping function between a set of input and output pairs, this mapping function is then used to process another dataset, i.e., to predict the labels of a test set.[52].

- Unsupervised Learning (UL), where the training data consists of a set of input data without any corresponding target values and the task is to make inferences from this unlabeled data [35, 188].

- Semi-supervised Learning (SSL), this approach is a mixture of supervised and unsupervised learning, where the training data typically consists of a small amount of labeled data with a large amount of unlabeled data [54, 194].

- Reinforcement Learning (RL), this approach is concerned with the problem of finding suitable actions to take in a given situation in order to maximize a reward, where the learning model discovers the target values by a process of trial and error (i.e., the target values are not provided) [249].

- Deep Learning (DL), this approach is concerned with building more accurate in-depth models which solve problems by mimicking how the human brain works [120, 236]

- Active Learning (AL), where the learning model is allowed to obtain the desired outputs at new data points, by asking queries in the form of unlabeled instances to be labeled (e.g., by a human annotator) [70].

The data mining techniques used vary according to differing requirements of different applications. The most widely used techniques, across many real world applications, include but are not limited to [35, 126, 174]:

- **Classification**, a supervised learning technique that learns the structure of a dataset's items, where these items belong to distinct groups (i.e., categories) known as classes. A classifier model is used then to estimate the class labels of new observations.

- **Regression**, another supervised learning technique for determining the strength of the relationship between a dependent variable and a number of other varying, independent variables. Here, the output variable is a real value (instead of class in the classification).

- **Association rule learning and frequency counting**, discovering of relationships between the variables exhibited by the dataset, and the identifying of the most significant values in the dataset. Association rule learning can be considerd as unsupervised learning technique [130].

- **Clustering**, an unsupervised learning technique that is concerned of placing objects and structures from the dataset into groups, based on their degree of similarity.

- **Anomaly detection**, the identifying of items, events or observations which do not conform to expected patterns or to the incidence of other items in the dataset. There are three approaches to the problem of anomaly detection: supervised, unsupervised, and semi-supervised learning [136].

This thesis seeks to investigate both the supervised and the deep learning approaches to learning from streaming data. Later in this chapter, we will discuss the existing literature on deep learning and artificial neural networks (this is related to Chapter 6). Furthermore, a detailed discussion about the existing classification techniques for data streams is presented in Chapter 4. However, this thesis does not aim to provide a broad overview of the data mining techniques or the learning paradigims and the interested reader is referred to [4, 62, 120, 126, 174, 199, 236, 286].

2.4 Mining and Processing Data Streams

Mining data streams presents inherent difficulties, because it requires processing of an unlimited amount of data and performing efficient calculations on the fly with bounded computing memory and limited processing cababilities [17, 87]. Obviously, such continuous and rapidly arriving data must be processed quickly and efficiently — in one or only a few passes. The analysis becomes more valuable when it is done in real time while the data is in motion, and in most cases, data loses its importance as time passes [117]. Therefore, conventional data mining techniques cannot be directly applied to data streams. Most of conventional data mining techniques entail the availability of the whole dataset before the processing (i.e, the training) can begin [126, 286]. Moreover, multiple scans of the data are

often required in order to the sought-for information; this is unrealistic in relation to data streams [3, 102].

A considerable amount of literature has been published on processing data streams [3, 69, 81, 102]. The existing approaches fall into two categories: Data Stream Management Systems (DSMSs), and Complex Event Processing Systems (CEPs) [68]; these are both aimed at providing low delay processing even in the presence of large volumes of input data generated at rapid rates [17, 189]. These systems are based on data models, such as relational models, which apply only a limited predefined set of operations with a fixed structure on the data streams [117]. Particularly, Data Stream Management Systems (DSMSs) inherit their data and query model from conventional Database Management Systems (DBMSs). With these latter systems DSMSs, data stream processing operates by processing time windows or sliding windows of the data, to select only the most recent elements in a stream — either based on their element number (in count-based window systems), or on their time stamps (in time-based windows) [189]. The basic concept of Complex Event Processing Systems (CEPs) is to identify patterns in data streams which represent meaningful situations in the application domain [44]. Surveys such as that conducted in [17, 33, 213] show the differences between Database Management Systems (DBMSs) and Data Stream Management Systems (DSMSs). Table 2.1 summarises the main differences between the two processing systems [17, 33, 44, 68, 117, 189, 213].

Table 2.1: Differences between DBMSs and DSMSs.

Database Management Systems	**Data Stream Management Systems**
Scheduled data arrival	Non-controlled data arrival
Bounded data	Infinite data
Persistent data	Volatile data streams
One-time queries, random access	Continuous queries, sequential access
Fewer limitations on storage and retrieval	More limitations on storage and retrieval
Relatively low update rate	Potentially extremely high update rate
Little or no time requirements	Real time requirements

2.4.1 Evaluating Data Streams Mining Techniques

Data streams set new requirements, compared to the conventional databases, in regard to the evaluation of the performance of data mining techniques which learn from these streams [106]. As stated before, while all available data are considered in data mining techniques which operate on conventional databases, streaming scenarios restrict the processing to a certain window of concern (i.e., the most recent elements in the stream) [117, 126, 286]. Thus, existing methods which are used to evaluate data mining techniques that learn from conventional databases (e.g., cross-validation or where the dataset is splitted into train and test sets) are not applicable when learning from dynamic and time-changing data streams [105, 106]. So in order to build a picture of the performance of data mining techniques which learn from evolving data streams, two common approaches have been proposed in the literature [106–108, 168] to evaluate these learning techniques over data streams: holdout evaluation, and predictive sequential evaluation.

In holdout evaluation approach, the learning model is tested on an independent set of the dataset. Typically, this set is chosen at regular time intervals (e.g., the end of the window). Then the estimated loss on test set is used as performance indicator [106, 168]. While in predictive sequential "Prequential [73]" evaluation, the error of a learning model is computed from each example (or a sequence of examples) in the data stream [106, 107]. In particular, the learning model makes a prediction for each example in the stream, then the prequential-error is computed based on an accumulated sum of a loss function, L, between the prediction, $\hat{y}_i$ and observed values, y_i, this can be expressed as [106, 168]:

$$prequential - error = \sum_{i=1}^{n} L(y_i, \hat{y}_i) \tag{2.1}$$

Both approaches provide a way to monitor the evolution of learning as a process, however, the performance estimates from both approaches can be affected by the order of the examples [107].

2.5 Common Issues in Mining Data Streams

Mining data streams is challenging due to the characteristics of such streams, viz: being continuous, unbounded and 'high-speed' data, of a non-stationary nature, which must be processed on the fly using minimal computational resources. An additional challenge is imposed by presence of imbalanced data streams in many real-world applications. This section discusses challenges that may occur when mining heterogeneous data streams; these issues should be taken into consideration when developing mining techniques for streaming data.

2.5.1 Concept Drift

One of the biggest challenges in classifying data items from real-world data streams is concept drift [280]; this phenomenon occurs when data streams are generated in non-stationary environments. A non-stationary environment is where the probabilistic properties of the data change over time (i.e., its underlying distribution can change dynamically over time) [80, 107], for example, the weather in one place is different from one season to another, and a customer's shopping behaviour changes over time, according to fashions and the season [107, 247]. Changes to data stream distributions over time may lead to the emergence of new classes or to changes in the existing classes [108, 199].

Simply, if X represents a set of input variables and Y represents a set of target variable, then concept drift between two different time stamps, t_0 and t_1, can be defined as follows [108]:

$$\exists x \in X : p_{t_o}(x, y) \neq p_{t_1}(x, y) \tag{2.2}$$

Where p_{t_o} and p_{t_1} denote the joint probabilities at time t_0 and t_1 , respectively, between the input variable (or set of variables) x and the target variable y. This means that [108]:

- the prior probabilities of class $p(y)$ may change;
- and/or the class conditional probabilities $p(x \mid y)$ may change;
- and as a result, the posterior probabilities of class $p(y \mid x)$ may change, affecting the prediction.

In fact, concept drift can cause the predictive performance of the classifiers to be reduced over time [196]; therefore, the learning model must detect such changes (drifts) and adapt itself to the current state of the environment [166, 237]. Previous studies have distinguished between two types of drift [108, 199, 247]:

- Real concept drift, which indicates changes in the conditional distribution of the target variable given a set of input variables, regardless of any changes in the input distributions.
- Virtual drift occurs when the distribution of the incoming data changes, $p(x)$, without affecting the conditional distributions of the target variables $p(y \mid x)$.

Hence, in both cases the joint distribution, $p(x, y)$, changes.

Drift may occur suddenly (e.g., the changes in a customer' reading interests from one topic to another) [108, 205, 206, 222], or it may happen gradually (e.g., the change in daily behaviour between the week days and the week end) [108, 158, 159, 288] or it may be the case that drift is reoccurring when previously seen concepts reoccur after some time (e.g., the seasonal changes in weather or fashions). Furthermore, another scenario of drift is when new classes that were not seen before are introduced, one or more types of drifts may occur in data simultaneously [108, 196, 296]. It is interesting to note here, that anomaly (random changes) cannot be considered as concept drift, as learning techniques do not need to adapt with such changes [53, 108].

2.5.1.1 Dealing with Concept Drift

Techniques developed to handle concept drift can be divided into three main groups [137, 247, 259]:

- Adaptive base learners, the simplest way to cope with concept drift, where the learner can dynamically adapt to new training data.
- Learners which modify the training set, commonly, the training set is modified (i.e., to include a subset of the previously seen examples) either by using windows approach or instance weighting approach.
- Ensemble techniques, where multiple (often weak) classifiers are combined to form one ensemble model, these classifiers are trained on the same dataset and then the results are used to provide the final prediction .

The first algorithms designed to deal with drifting data were STAGGER [235], IB3 [7], and the suite of FLORA algorithms [279, 280]. STAGGER [235] and IB3 [7] are instance-base learners that learn from one example at a time (i.e., without keeping the previous examples), the learning model is updated continuously with the most recent example. Learning models, in both STAGGER and IB3, deal explicitly with concept drift, but the main limitations in these models that they only able to adapt to gradual concept drifts, and that their adaptation is relatively slow [108]. Whereas the FLORA [279, 280] algorithms use a sliding window of the most recent examples (instead of one example). Here, the learning model is updated with the arrival of new training window, and the old examples from the previous windows are discarded. FLORA [279] was the first method that used an explicit forgetting mechanism (using adaptive windows) to deal with evolving data with unpredictable changes [108]. The original FLORA algorithm used a fixed length window size. Later, an adaptive windows approach was introduced in FLORA2 algorithm which adjusts the window size dynamically when a change is detected — in order to improve the learning performance [108, 279, 280]. The

rigid window adjustment heuristic is a major weakness of FLORA algorithms, moreover, it is difficult for the learning model to recover after long drifts [244]. Several techniques [2, 82, 161, 163] have been proposed then in the literature based on forgetting the outdated observations in order to detect and cope with drift in evolving data streams [108]. Forgetting can be done abruptly when learning models discard the old examples in the previous windows[2, 82, 279, 280], or can be done gradually when old examples are not completely discarded, instead, they are associated with weights that reflect their age [161, 163].

One of the first adaptive ensemble techniques was proposed in [34]. It was developed for classifying streams of text in order to detect user interests and build a content-based filter. Accordingly, two classifiers were used: a naive Bayes classifier for modeling the long-term interests and a nearest neighbor classifier for capturing the short-term interests of the user. Typically, adaptive ensembles consist of set of individual models which are combined to form a single new model [261]. The average or a weighted average predictions of these individual models are combined to predict the new incoming instances [161, 261]. Later, many ensemble techniques have been presented in the literature to deal with concept drift in streaming data. For example, the Streaming Ensemble Algorithm (SEA) [245] which trains a separate classifier on non-overlapping batches of training examples, the trained classifier is added then to a fixed-size ensemble, while the worst-performing classifier is discarded, then a majority voting is used to make the prediction. Another example is presented in [267], Wang et al. proposed the Accuracy Weighted Ensemble technique (AWE) which used a weighted voting instead of the majority voting. The weight reflects the benefit of using a specific individual model in comparison to a random classifier. However, the performance of the AWE technique depends on the size of training batch [45]. Despite that small batches are better than large batches with regard to drift detection, using small batches can worsen the classifier performance (only few examples are used to train the classifer). Therefore, authors in [45] presented the Accuracy Updated

Ensemble technique (AUE) which extends the AWE by using online component classifiers and updating them according to the current distribution. In this technique, the component classifiers can be trained on more examples and become more accurate when no change occurs. However, updating many components with similar examples may reduce their diversity [45]. Another ensemble technique, Adaptive Random Forest algorithm (ARF), was proposed in [119]. ARF trains a background tree after a warning has been detected and then only replace the primary model if the drift has occurred, this resampling technique used in order cope with different types of concept drifts. In general, ensemble approaches have been found to be the most popular methods for stream classification due to their robustness [108, 161]. However, it is important to note that training many base classifiers in the ensemble technique is a time consuming process [119].

Furthermore, change detectors, such as the Drift Detection Method [104] and the Adaptive Sliding Window Algorithm [26] were introduced to detect concept drift in data streams. Change detectors are stand-alone techniques that can be used in combination with any stream classifier [262], typically, the evolution of the performance indicators or the raw data are monitored then they are statistically compared to a fixed baseline [108]. This can be used to provide information about the dynamics of the process generating data [108]. Many works proposed in the literature [18, 153, 266] have used detection approaches based on monitoring two distributions. In these approaches, a fixed reference window is used to summarize the past information with a sliding detection window over the most recent examples. The main limitation of these approaches is the memory requirements, as they need to store data in two windows [108].

2.5.2 Imbalanced Classes Distribution

Another potential issue that is associated with mining data stream is imbalanced classes distribution — which sometimes is referred to as unbalanced data. Im-

balanced classes problem is a common problem in real-world applications with regard to data mining and pattern recognition domain [46]. Typically, it occurs in classification scenarios when classes are not represented equally in the dataset, i.e., some of classes may be rare, or they may only appear on certain occasions in the data stream [178, 273]. This can be seen in the case of spam filtering, fraud detection, and fault diagnosis in computer monitoring systems [272, 273]. In such cases, classifiers tend to be biased toward the majority classes, resulting in a high accuracy classification performance, however these classifiers will not be able to detect any of the instances that belong to the minority class; classification becomes even more difficult if there are multiple classes [273].

Several works have been developed in the literature to tackle the imbalanced classes problem as it emerges in static data; most approaches aim to achieve a more balanced distribution by using various forms of re-sampling [178]. However, parallel research in the context of data streams is limited to only few works [46], and too little attention has been paid to the multi-class imbalance issue [273]. Methods which deal with imbalanced classes in data streams can be categorized into data level and algorithm level approaches [273]. One of the earliest attempts to handle data streams with skewed distributions was proposed by Gao et al. in [109]. An ensemble learner was presented, where each incoming batch was divided into a set of positive and negative class instances. Then, all the positive examples and a subset of the negative class instances which is selected randomly, are combined to form the training set for the ensemble classifier. However, this technique did not consider the drift of the the minority class [137], in addition, it only dealt with two-classes imbalance problems. A recent work which deals with multi-class imbalance in data streams was presented in [272]; the authors used a time decay function to detect the imbalance rate dynamically and to work with this they proposed two re-sampling-based ensemble methods, Oversampling-based Online Bagging (OOB) and Undersampling-based Online Bagging (UOB). In addition, a concept drift detector and an adaptive online learner which take

corresponding actions when class imbalance and concept drift happen. Despite this technique was proposed for multi-class imbalance problems, it only tested in this work on two-class cases. A more recent technique was proposed in [197]; this technique depends on an ensemble method which made use of voting based classifiers, wherein different class weights are used to help detect the presence of an unbalanced (i.e., relatively low frequency) class in the validation set. However, this technique requires an initialization step before learning takes place, and it is not clear how the class weights will remain accurate in the case of evolving streams.

Class imbalance introduces challenges to classifiers which are used for stream mining in terms of the classifier performance metrics and evaluation procedures [92, 178, 272]. Simple performance metrics, such as accuracy (the percentage of the correctly classified instances in the test set), can be easily calculated and interpreted, it can be represented by Equation 2.3 (in binary classification):

$$Accuracy = \frac{TruePositives + TrueNegatives}{Positives + Negatives} \tag{2.3}$$

However, in relation to imbalanced classes classification scenarios, it it easy to obtain a high accuracy with the assumption of that errors are equally cost — often the cost of misclassifying instance of the minority class is much higher [92, 270]. Therefore, many other performance metrics have been introduced to replace the accuracy measure which yields overoptimistic estimates (because it entirely ignores the low frequency classes). For example, kappa statistic, κ, which was introduced by Cohen [65], has been used in the literature as a performance measure that takes class imbalance into account. Simply it compares the accuracy of the classifier with the accuracy of a random classifier [31, 92], this can be expressed as:

$$\kappa = \frac{Acc_0 - Acc_c}{1 - Acc_c} \tag{2.4}$$

The quantity Acc_0 is the classifier's prequential accuracy 2.1, it is computed based

on an accumulated sum of a loss function between the prediction and observed values [31, 147]. Whereas Acc_c is the expected accuracy (i.e., the probability that a random classifier which assigns the same number of examples to each class makes a correct prediction). The kappa value, κ, ranges between -1 and 1, and if κ is less than zero, it means that a random guessing surpasses the classifier's performance. In addition, mesures such F-score (the harmonic mean of recall and precision), G-mean, and the area under the Receiver Operating Characteristic curve (ROC curve) are commonly used in the literature, but they are originally designed for two-class problems, and can not be applied directly on multi-class tasks [31, 47, 178, 237, 269].

2.5.3 Mining Frequent Patterns

Frequent patterns are items that appear in the dataset with a frequency above a user specified threshold. Frequent pattern mining was introduced in 1994, when Agrawal and Srikant [6] proposed the Apriori algorithm, which generate set of association rules, for market basket analysis. Simply, association rules are set of statements which indicate that certain groups of items or events tend to occur together. There are two basic concepts related to such association rules: the first is the confidence or the accuracy which defines how often the rule is true; the second is the support or the coverage which indicates the frequency threshold of a rule. If I represents a set of items, an association rule is an implication of the form: $A \Rightarrow B$ where: $A \subseteq I, B \subseteq I$, and $A \cap B = \emptyset$. Then, the support, Sup, and the confidence, $Conf$, can be defined by the following Equations 2.5, and 2.6.

$$Sup(A \Rightarrow B) = p(A \cup B) \tag{2.5}$$

$$Conf(A \Rightarrow B) = p(B \mid A) \tag{2.6}$$

Association rule mining consists of two steps. First, find all frequent item sets

and then generate rules from these sets. An item set is considered to be frequent when its support is at least equal to the minimum support threshold [134].

Later, many other algorithms were developed for the purpose of detecting different types of pattern in the transaction data. In general, there are three basic frequent pattern mining methodologies: Apriori, FP-Growth and Eclat [113]. FP-Growth [123] is distinguished from the other methodologies by the fact that, with it, there is no candidate generation; the method works in a divide-and-conquer way and requires fewer database scans, so it is more suitable (than the other methods) for huge database transactions and long patterns. It simply scans the database first to construct a list of frequent items; the items are sorted by descending order of frequency as the method proceeds; then in accordance to the frequent-item list, the database is compressed into a frequent-pattern tree, FP-tree, which retains the itemset association information. The FP-tree, instead of the raw dataset, is mined to find frequent patterns — by starting from each frequent length-1 pattern, constructing its conditional pattern base, constructing its conditional FP-tree, and then performing mining recursively on each such tree [123].

However, a major issue encountered by the aforementioned frequent pattern mining approaches has been the rare pattern problem [179]. This is because they all use a single support threshold to identify what constitutes a frequent pattern across the entire dataset. In many real-world applications, choosing a low threshold value will identify a large number of meaningless patterns, whereas a high threshold value will lead to the skipping of rare but significant patterns [155]. To solve the rare patterns problem, multiple minimum supports were introduced by Liu et al. in [179], based on the Apriori algorithm. Subsequently, several proposals were made in the literature, such as Conditional Frequent Pattern-Growth (CFP-Growth) [140], an FP-growth-like algorithm which permitted setting minimum support thresholds for rare items. A list of minimum support thresholds, MIS, was proposed to indicate the minimum support threshold for each item instead of one threshold. Later, CFP-Growth++ [156] was developed as an extension of the

CFP-Growth algorithm with three pruning techniques: least minimum support (LMS) which represents the lowest support of all frequent patterns, a conditional minsup, and infrequent leaf node pruning in order to identify which suffix patterns can generate frequent patterns at higher order. However, a major limitation in these works lies in the fact that this MIS list is required as prior knowledge (i.e., both require scanning the whole dataset before algorithms commences). In addition, some approaches were introduced to detect the rare patterns only, for example, RP-Growth technique [258], an adaptation of the FP-Growth which was proposed to find rare item sets using two threshold values: Minsup and Minraresup. Minsup value represents the upper boundary of what is counted as rare (i.e., items which have support values less than Minsup are considered rare) and Minraresup value represents the lower boundary of rare items ((i.e., items which have support values less than Minraresup are considered noise), Minsup and Minraresup represent a range, where Minraresup is less than Minsup value. Like the FP-Growth, this algorithm scans the dataset first to calculate the supports, but different from the FP-tree in the FP-Growth which is built based on the frequent item list, the RP-Tree here is built based only on transactions that have at least one rare item set. However, the use of fixed values for support thresholds, in addition to the construction cost of the conditional sub-trees, limit the application of these algorithms within streaming scenarios.

One of the first attempts at discovering frequent patterns in streamed data was developed by Giannella et al [113]. In their work, the FP-Stream algorithm was proposed based on a batch environment. For each batch, the frequent patterns are extracted by means of the FP-Growth algorithm applied on a FP-tree structure representing the sequences in the batch. However, due to the use of a single support, rare patterns are not detectable by this method. Subsequently, several algorithms [132, 141, 142, 170, 176] have been proposed for discovering patterns in data streams. These works designed to detect rare patterns, by scanning the stream items once or multiple times, but due to the fixed supports used by these

algorithms, they still cannot handle efficiently the issue of concept drift in the emerging patterns.

2.6 Time Series Forecasting

Forecasting time series is simply the use of the available observations at time t, commonly referred to as the inputs, to predict the future values of the data items in the time series at time $t+1$, commonly referred to as the outputs [41, 42, 200]. The data used in many forecasting problems commonly falls into one of two categories: univariate time series, where only one variable (i.e., time series) is measured over time; or multivariate time series, that involve several variables. Hence, time series forecasting problems may be grouped into four broad types [42, 240]:

- Univariate inputs and univariate outputs
- Univariate inputs and multivariate outputs
- Multivariate inputs and univariate outputs
- Multivariate inputs and multivariate outputs

Brockwell et al. presented a general approach to time series modelling and forecasting in [41]:

1. Plot the series, then examine the main features of the graph, and check if there are any significant trends (i.e., increasing or decreasing) or any seasonality (i.e., patterns) exhibited by the series - or alternatively if there are any outliers that a domain expert can find a reason for (i.e., they are non-random).

2. Choosing and fitting a forecasting model based on the available historical data and the apparent relationships between variables; most of the statistical models require the removal of the trend and seasonal components in order to make the series stationary before application (of such a technique).

3. Use and then evaluate the forecasting model. It is common practice to divide the data in the time series into two sets: the training set, used for estimating the forecasting parameters; and the test set, for evaluating the model's accuracy. There are different methods to measure the forecast accuracy, the most commonly used are: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) [144]. MSE and RMSE are measures of the average (or the square root of the average) of squared differences between prediction and actual observation, whereas MAE is based on the absolute error calculation, the lower values of these metrics indicate better prediction, this can be represented as follows [238]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2 \tag{2.7}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{2.8}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |e_i| \tag{2.9}$$

Where the error e is the difference between prediction and actual observation.

The statistical approaches which are most widely used for time series forecasting are the Auto Regressive Integrated Moving Average (ARIMA) [39], and the Exponential Smoothing (ES) models [43, 144]. Many variations of these approaches have been proposed in the literature to deal with seasonal components and/or multivariate time series. For example, Holt in [138] developed a method for forecasting trend in time series based on the weighted ES, this method was extended by Winters in [284] to capture the seasonality. Later, the Holt-Winters method was extended in [251], in order to deal with more than one seasonal component (i.e., within-day and within-week seasonality) in a univariate time series

of half-hourly electricity demand. Furthermore, the seasonal ARIMA (SARIMA) [181] was developed by including an additional seasonal terms in the ARIMA models (i.e., it explicitly models the seasonal component) to deal with the seasonal components. Moreover, the Vector ARIMA (VARIMA) model was introduced to deal with multivariate time series, however, the VARIMA models are difficult to estimate because of the large numbers of parameters involved [75]. In addition, some models have attempted to deal with more complex seasonal components: e.g., BATS (Box-Cox transformation, ARMA errors, Trend and Seasonal components) [77] and TBATS (Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components) [76].

TBATS was developed by De Livera et al in [76], it uses a combination of Fourier representations and a Box-Cox transformationwith with an exponential smoothing state space model and presents trigonometric formulation as a means of decomposing complex seasonal time series, in order to deal with complex seasonality in automated way with no seasonality constraints. According to [76], TBATS is more accurate than BATS and requires fewer parameters to be estimated. The proposed decomposition in the TBATS leads to extraction of seasonal components which are otherwise not apparent in the time series. Moreover, the seasonality in the TBATS model is allowed to change slowly over time (unlike other methods where the seasonal patterns are forced to be repeated periodically without change). However, due to the complex computations, TBATS model can be slow to estimate, especially with long time series [144]. The TBATS model can be described by the equations in Figure 2.1.

More recently, researchers have shown an increased interest in using machine learning models for forecasting time series [37, 131, 211, 217]. In particular, deep learning approaches and neural network models have drawn attention because of these have been used across a number of different domains for forecasting time series. For example, a deep learning approach using Stacked Denoising Auto-Encoders (SADEs) was developed to forecast indoor temperature in [227]; an

Model:

$$y_t^{(\lambda)} = l_{t-1} + \phi\, b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t$$

$$l_t = l_{t-1} + \phi\, b_{t-1} + \alpha\, d_t$$

$$b_t = \phi\, b_{t-1} + \beta\, d_t$$

$$d_t = \sum_{i=1}^{p} \varphi_i\, d_{t-i} + \sum_{i=1}^{q} \theta_i\, e_{t-i} + e_t$$

Seasonal part:

$$s_t^{(i)} = \sum_{j=1}^{(k_i)} s_{j,t}^{(i)}$$

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cos(\omega_i) + s_{j,t-1}^{*(i)} \sin(\omega_i) + \gamma_1^{(i)}\, d_t$$

$$s_{j,t}^{*(i)} = -s_{j,t-1}^{(i)} \sin(\omega_i) + s_{j,t-1}^{*(i)} \cos(\omega_i) + \gamma_2^{(i)}\, d_t$$

$$\omega_i = 2\pi j / m_i$$

Where:

$y_t^{(\lambda)}$ - time series at moment t (Box-Cox transformed)

$s_t^{(i)}$ - ith seasonal component

l_t - local level

b_t - trend with damping

d_t - ARMA(p,q) process for residuals

e_t - Gaussian white noise

Model parameters:

T - Amount of seasonalities

m_i - Length of ith seasonal period

k_i - Amount of harmonics for ith seasonal period

λ - Box-Cox transformation

α, β - Smoothing

ϕ - Trend damping

φ_i, θ_i - ARMA(p, q) coefficients

$\gamma_1^{(i)}, \gamma_2^{(i)}$ - Seasonal smoothing (two for each period)

Figure 2.1: TBATS model description using mathematical equations [76]

electricity demand forecasting system used an ensemble of Deep Learning Belief Networks (DBNs) and a Support Vector Regression (SVR) model in [217], and deep recurrent neural networks were used in a more recent study [239]. In addition, a wind power forecasting approach based on wavelet transforms and convolutional neural networks was proposed in [268].

2.7 Artificial Neural Networks and Deep Learning

Deep Learning (DL) is one of the machine learning approaches that has seen huge growth in recent years. Although the concept of deep learning appears to be new, it has been known in the literature under different names for many years; the concept was first described in the 1940s as Cybernetics [120, 193, 239]. What makes deep learning models attractive is their ability to use existing data

to train algorithms so that they can learn complicated concepts and then use these to make predictions about new data. Therefore, deep learning has become a powerful tool for learning how to perform highly complex tasks across different domains, e.g., computer vision, speech recognition, and automated translation services [78, 120, 236].

Artificial Neural Networks (ANNs) are at the core of the current deep learning techniques. Typically, deep learning models are composed of multiple layers of artificial neurons. An ANN is basically a network of computing units (i.e., neurons) linked by directed connections; each computing unit performs a number of functions (e.g., aggregation then activation functions) on the inputs. Specifically, the aggregation function calculates the sum of the inputs from the incoming connections, then the activation function is applied on this aggregation result; subsequently, the outputs are spread over the outgoing connections as inputs to other units, of the next layer [180, 204].

The simplest ANN architecture is the Multi-Layer Perceptron (MLP) model, a feed-forward neural network consisting of multiple layers of nonlinear neurons [229]. Typically, such a network has three layers: the input layer, a single hidden layer which can be used to extract features from the inputs and to compute complex functions, and the output layer. Figure 2.2 shows a graphical representation of MLP with a single hidden layer. It has commonly been assumed that there should be more than one hidden layer in the neural network, for such to be considered capable of implementing deep learning; otherwise neural networks capabilities are referred to as shallow learning [120, 236].

It is important to note here, that the ANN architecture described above is not the only one; there are several kinds of artificial neural networks described in the literature, e.g., Convolutional Neural Networks (CNNs) [98]; and Recurrent Neural Networks (RNNs) [232] of which Long-Short Term Memorys (LSTMs) [135] and Gated Recurrent Units GRUs [61] are specific types of neurons layers.

RNNs are quite different from the other types of neural network, and have

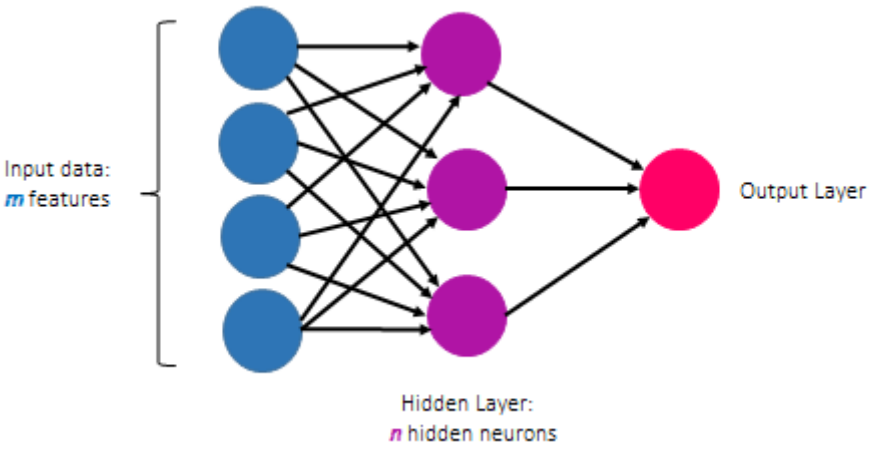


Figure 2.2: A graphical representation of Multi-layer Perceptron model with a single hidden layer

shown unique capabilities in terms of dealing with sequential data [120, 133, 248]. RNNs can capture the sequential nature of variable-length sequences $X = (x_1,x_2,\ldots,x_t)$, by having a recurrent hidden state h_t whose activation at each time, t, is dependent on that of the previous time, $t-1$, (i.e., the output of the hidden layer neurons are used as inputs to these neurons). RNNs can be represented mathematically as follows:

$$h_t = f(x_t, h_{t-1}) \tag{2.10}$$

where x_t is the input, and h_t represents the hidden state at time t, and this makes the output prediction as well. The diagram in Figure 2.3 shows a simple representation of the RNN architecture.

A number of different variants of the RNN architecture have been proposed in the literature to eliminate some of the potential training problems (vanishing/exploding gradient problems, where the gradient becomes too small and prevents the weight from changing its value or becomes too large which results in NaN values) [135]. The Long-Short Term Memory (LSTM) model [135] is a modified RNN structure with a memory cell — here, typically, the hidden layer of a RNN is replaced by a complex block of computing units composed of gates (an input gate, an output gate and a forget gate). A more recent variant, the Gated

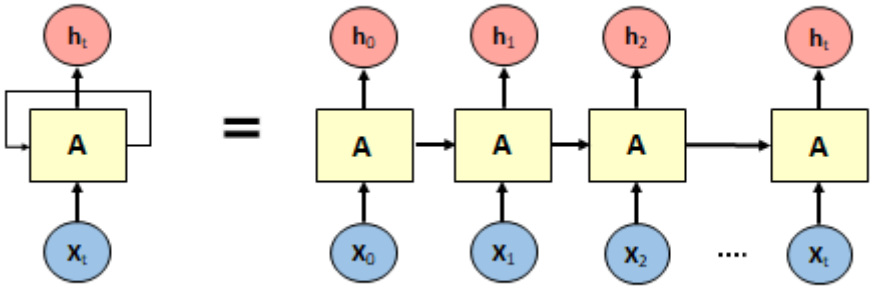


Figure 2.3: A graphical representation of Recurrent Neural Network architecture

Recurrent Unit (GRU), was proposed in 2014 by Cho et al. [58]. The following section presents a detailed description of the GRU concept; GRUs will be used in Chapter 6 for time series forecasting in terms of predicting the availability of car parking spaces. Detailed descriptions of the other architectures of neural networks is beyond the scope of this thesis; the interested reader is referred to [120, 203, 236].

2.7.1 Gated Recurrent Units (GRUs)

The Gated Recurrent Unit, known as the GRU, layer was proposed in order to allow each recurrent unit to capture the dependencies operating over different time scales. Specifically, it consists of two gates, a reset gate and an update gate, that modulate the flow of information inside the unit [58, 61]. The diagram in Figure 2.4 shows a simple representation of a GRU layer.

Given a sequence $X = (x_1,x_2,\ldots x_n)$, the update gate z_t of the GRU at time t decides how much the unit should update its activation or content h_t (i.e., how much information from previous time steps needs to be passed to future ones); this is defined as [58, 61, 162]:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{2.11}$$

where W_z and U_z are the corresponding weights. The reset gate r_t helps the unit to decide how much of the past information to forget; for example, when r_t is close to zero, this makes the unit act as if it is reading the very first symbol of an input sequence, allowing it to forget the previously computed state. The value at the reset gate, r_t, is computed in a similar way to the value at the update gate z_t, with different weights W_r and U_r as follows [58, 61, 162]:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{2.12}$$

The current memory content $\widetilde{h}_t$ uses the reset gate to store the relevant information from the past, it is computed as [58, 61, 162]:

$$\widetilde{h}_t = \tanh(W x_t + r_t U \odot h_{t-1}) \tag{2.13}$$

where W and U are weights, $\odot$ is an element-wise multiplication, and $tanh$ is a nonlinear activation function. Finally, the activation, h_t, of the GRU at time t is calculated using the update gate z_t; this is defined as follows [58, 61, 162]:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \widetilde{h}_t \tag{2.14}$$

GRUs can be considered as a simplified variant of LSTMs, because they both have similar designs. Despite the fact that GRUs are quite new and their potential

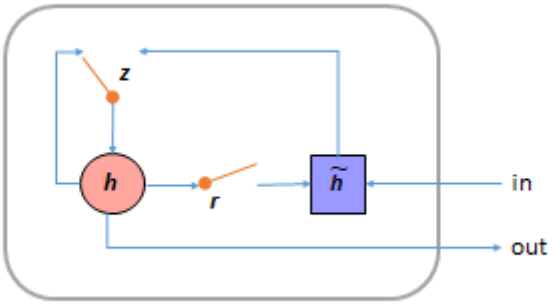


Figure 2.4: A graphical representation of a Gated Recurrent Unit layer

has not been fully explored, they have nevertheless been shown to be faster to train than LSTMs, as they have fewer gates; they also have demonstrated comparable performance [58, 61, 149, 162].

2.8 Chapter Summary

We live in the age of the analytics; there is an explosive growth of data — produced by information digitisation — in volume, velocity and diversity. This has dictated the need for efficient and effective learning techniques and analysis methods which can operate on data streams generated in dynamic environments (i.e., that are changing unexpectedly). Aggregating and analysing data streams from diverse sources has the potential to derive insights from these data streams in order to make smarter and real time decisions. However, mining techniques which operate on conventional data cannot be applied directly on these data streams. This because that in conventional databases, already stored data may be scanned and processed many times. In contrast, data streams can be only processed online, thus, learning models can be trained either incrementally by continuous update or by retraining using recent batches of data [80, 97, 107, 210, 241].

In many real-world applications, data streams are produced in non-stationary environments which change dynamically, this results in the phenomenon of concept drift [280]: the change of underlying data distributions over time. Another difficulty arises when it is assumed that the data is balanced, however in many real-world problems, the class distributions can become severely underrepresented in data streams [178, 273]. The problem of class imbalance becomes more acute in multi-class learning tasks, as the data are dynamically evolving and it is impossible to see the whole picture of data [273].

This chapter presented a brief overview with regard to mining data streams and the methods used in this investigation. It examined the existing literature that relates to learning from data stream which generated by variety of sources

in dynamic environments. Moreover, it described and discussed main challenges that may emerge in terms of learning from data streams that exhibit concept drift and class imbalance. So far, several studies [46, 109, 197, 273] investigating data streams have been carried out on evolving data streams which have skewed distributions, however to the best of our knowledge, these studies focused on binary class problems [137] and there is a distinct lack of research in multi-class scenarios. Undertaking such an advanced study which examines the combined problems of stream processing, concept drift and multi-class imbalanced scenarios, presents a challenge that we aim to address in this thesis.

The development of new methods which can cope with concept drift in evolving data streams and which can deal with imbalanced classes distributions was a main challenge in this work. Another constraint was concerned with evaluating the performance of learning techniques which deal with data streams online and in a dynamic environment. This relates to the choice of validation procedure, and the choice of the evaluation metrics which can measure the performance of learning technique at any time and which can deal with unbalanced data.

A major limitation of this study was the lack of sufficient data streams which exhibit the presence of heterogeneous data streams produced by diverse sources. Therefore, we developed a simulation model of a dynamic environment which produces heterogeneous data streams from set of IoT devices. This was in order to allow an understanding of the streams dynamics in addition to certain phenomena present in these data streams. The set of data streams derived from the simulation model is described in next chapter. Then, we used this dataset, in Chapter 4 and Chapter 5, to establish new mining techniques which can deal with evolving imbalanced data stream. In Chapter 6, another time series dataset was constructed from the simulated data streams in order to explorer prediction techniques with regard to evolving imbalanced time series streams. These datasets, which derived from the simulation model, can serve as a benchmark for analysing data streams from heterogeneous sources.

Chapter 3

Experimental Framework

3.1 Introduction

The enormous proliferation of data streams generated in many real world applications produces new challenges [51]. The dynamic, high speed, incomplete, and noisy data streams generated from multiple heterogeneous sources, make it necessary to develop new data mining techniques in order to derive insights from such data streams [74]. The basic purpose of this thesis is to investigate an aspect of data mining: learning from multiple heterogeneous data streams generated from dynamic systems and thus to predict the behaviour of upcoming streams generated in subsequent time intervals. This chapter presents a broad description of the problem domain. Subsequently, it overviews the abstract formulation of the problem for this research; then it provides an illustrative scenario which can be used to facilitate the process of investigating the problem and exploring its solutions. In order to demonstrate the problem domain, a smart parking system based on the IoT has been adopted as a system model. This smart parking system provides scenario which illustrates a dynamic system requiring reasoning across a variety of streams. The system can be characterised as heterogeneous, dynamic, and unbalanced; this means that the streams are generated by different sources, in a non-stationary environment, and with skewed distributions. The smart parking

system's characteristics facilitated the implementing of the required system properties: there are many parking lots; different types of users; and the data streams are generated from sensors, smart pay stations, and mobile phones applications etc. These characteristics make the smart parking system a suitable problem to use as a basis for our work and the exploration of techniques.

3.2 An Abstract Formulation of the Problem

In this section, we formulate the problem of real time learning from imbalanced evolving streams. We consider the problem in terms of heterogeneous stream reasoning, where data coming from different sources — sensors and devices such as smart pay stations, mobile phones etc. — can be used to extract useful information and provide support to users in making their choices. Specifically, we assume an environment which represents a typical university or another organisation with different types of users that require the use of parking spaces for different periods of time, with different frequency etc. We are seeking to develop methods that may be used as part of an application to assist the different users who are searching for a park space for their vehicles in the situation there are multiple parking lots.

3.2.1 Data Streams from IOT Devices

In the context of this thesis, a data stream S is an unbounded, ordered sequence, implicitly ordered by arrival time or explicitly by time stamp, of data items i_{t_0} ,i_{t_1}, ...,i_{t_n}, ..., which are continuously generated over time. These data items can be simple attribute-value pairs similar to relational database tuples, such as the items in the streams exhibited in this thesis, or they might have more complex structures such as graphs.

Definition 1. *A Data Stream*, S_{Tn}, over a specific time interval, T_n [t_n ,..., t_{n+L-1}], is defined as a time series of data items, i_{t_n}, ...,$i_{t_{n+L-1}}$:

$$S_{T_n} = \{i_{t_n}, i_{t_{n+1}}, \ldots, i_{t_{n+L-1}}\}$$

where

- $i_{t_n} = (t_n, f_1, \ldots, f_a)$; a is the number of features of the data item
- t_n is the beginning of the time interval T_n
- t_{n+L-1} is the end of T_n
- L is the length of T_n

Here, it is assumed that there are different streams from various sources (src): sensors, smart meters, twitter feeds, mobile applications, etc.

Let Str be the set of structured data streams, $\{Str_1, Str_2, \ldots, Str_k\}$, generated by different IoT devices at a given time stamp t_n where:

k	is the number of sources that have generated the data streams
$Str_1(f_1, \ldots, f_x)$	is a stream from the first source src_1
$Str_2(f_1, \ldots, f_y)$	is a stream from the second source src_2
$Str_k(f_1, \ldots, f_z)$	is a stream from the k^{th} source src_k
x, y, z	are the numbers of features in each stream

Hence, in this work we assume that each stream consists of different features. Furthermore, we assume that the joint of these streams (based on specific features) over a specific time interval, T_n [t_n ,..., t_{n+L-1}], (e.g., a single hour) represents the current window-batch $JStr_{T_n}$, and the joint of these streams over the previous

time interval, T_{n-L} $[t_{n-L}$,..., $t_{n-1}]$, (e.g., the previous hour) represents the previous window-batch $JStr_{T_{n-L}}$. As the window-batches are time-based windows, so their sizes vary, each window may consist of any number of data items — in the joint stream.

3.2.2 Titled Time Windows

In the proposed approach, time-window batches are chosen as a means to represent/process the multiple streams from the IoT devices. More specifically, differently defined time-windows (titled-time windows) [113, 124, 214] will be used in order to detect the periodical patterns (the differing seasonal patterns: hourly, daily, weekly, etc.), from the evolving streams over different time intervals (an hour, a day, a week and a term). A time-window defined by its title is automatically self-maintained [103, 113, 124]; whenever it reaches the boundary of its time window granularity, the aggregates stored at its lowest granularity level are summarized and transferred to the upper granularity level. Typically, the natural time-windows [103, 113, 124] are represented in the diagram in the Figure 3.1, this is based on the notion of physical time being split in seconds, minutes, etc. As illustrated in the diagram, the maintenance of windows is straightforward, when four batches of 15-minutes-window are accumulated, they are merged together in order to represent one hour. After 24 hours are accumulated, a day batch can be built, and so on.

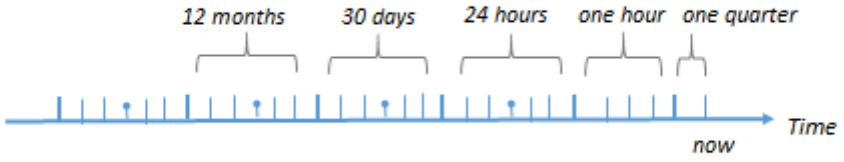


Figure 3.1: The natural titled-time-window structure

The concept of a titled-time window is similar to that of the sliding window concept where the most recent data are given more importance than old data [72, 187, 214]. But unlike the sliding windows which discard the old data items in data stream [214], the titled-time window focuses more on the recent data without completely ignoring the old data items in the stream [187]. The design of the titled-time window system is based on the fact that learning from data streams is time-sensitive. The recent changes at a fine granularity are valuable and can be used for purpose such as fraud detection (bank transactions) or fault diagnosis (monitoring systems) [101, 117]. Although the data items in the stream lose their significance over time [117], monitoring the long term changes at a coarse granularity can be very useful for identifying trends or detecting changes in patterns which exist within streams [101].

Definition 2. *A Titled-Time Window*, W_{T_n}, over a data stream S of a titled length L, is the subset of all data items in the stream S which consists of those items that occur between time stamps t_n and t_{n+L-1}:

$$W_{T_n} = \{i_{t_n}, i_{t_{n+1}}, \ldots, i_{t_{n+L-1}}\}$$

where

- the items $i_{t_n}, i_{t_{n+1}}, \ldots, i_{t_{n+L-1}}$ are data items from the stream S, this can represented as $\{i_{t_n}, i_{t_{n+1}}, \ldots, i_{t_{n+L-1}}\} \subset S$
- $t_n \in T$, is the time stamp at the beginning of the window W_{T_n}, $0 \leq n$
- $t_{n+L-1} \in T$, is the time stamp at the end of the window W_{T_n}, $L > 0$
- L is the length of the window, as identified by the user

The choice of the optimal length L of the window depends on the application domain. In general, the window size should be chosen taking into account the drift speed of a data stream, therefore small windows should be chosen in periods

during which the drift is fast, while larger windows should be decided upon during periods of slow drift [38, 113, 175]. It is important to note that the maintenance of windows does not necessarily follow the natural titled-time windows. For example, eight hours might be accumulated to form one working day, or five week days might form one week.

3.2.3 Car Parking Lots Problem Formulation

This section provides a formalization of the car parking lots problem. As has been mentioned above, a dynamic parking lots setting has been used in this thesis as a system model, where there are heterogeneous data streams generated from a variety of sources: sensors, smart pay machines and a parking mobile app. Based on the data streams described in Section 3.2.1, let Str be the set of structured data streams $\{Str_1, Str_2, Str_3\}$ generated by various IoT devices at a given time stamp t_n where:

$Str_1(f_1, \ldots, f_x)$ is a stream from camera sensors $sn_1 ... sn_m$; m: the number of sensors that monitor the entry of the set of parking lots

$Str_2(f_1, \ldots, f_y)$ is a stream from smart parking pay stations $ps_1 ... ps_l$; l: the number of parking pay stations

$Srt_3(f_1, \ldots, f_z)$ is a stream from parking mobile app

x, y, z are the number of features in each stream

Let the data items in the stream Str_1, which represents the data stream from the camera sensors, consist of the following features:

$$Str_1(sens\_loc, vehicle\_CT, vehicle\_plate, dir)$$

where:

- $sens\_loc$, is the location of the sensor

- *vehicle_CT*, is the car's crossing-time
- *vehicle_plate*, is the car's plate number
- *dir*, a boolean variable that indicates the car's direction, 1: incoming; 0: outgoing

The second stream Str_2, represents the stream from the pay stations; its data items will comprise the following features:

$$Str_2(pay\_loc, vehicle\_plate, park\_start, park\_end, park\_fee, discount)$$

where:

- *pay_loc*, is location of the station
- *vehicle_plate*, is the car's plate number
- *park_start*, is the start of the parking-time
- *park_end*, is the end of the parking-time
- $park\_fee$, is the parking fee paid
- *discount*, a boolean variable that indicates whether a discount was applied to the parking fee, 1: discount applied; 0: discount not applied

Let the last stream Str_3, be the stream from a mobile phone application, where each item consists of the following features:

$$Str_3(loc, vehicle\_plate, park\_start, park\_end, park\_fee, driver)$$

where:

- *loc*, is parking location obtained through the phone GPS.
- *vehicle_plate*, is the car's plate number

- *park_start*, is the parking-time
- *park_end*, is the end of the parking-time
- $park\_fee$, is the parking fee paid
- *driver*, the driver type from the user's profile

Let the joint of the three streams (the join operation being based on the car plate number) over a fixed time interval (e.g., a single hour) be the current window-batch, $JStr_{T_n}$, and the joint of the three streams at the previous time interval (e.g., the previous hour) be the previous window-batch, $JStr_{T_{n-L}}$. As the window-batches are time based, so their sizes vary, each window of the joint stream may consist of any number of data items.

The scenario described by the streams above is a typical scenario in relation to car parking lots, although the number and the structure of streams may vary slightly due to different lots specifications and the available facilities. The main aim of this thesis is to develop new methods: (i) which can learn from evolving streams with skewed distributions and capture the behaviour of different drivers (in Chapter 4), and (ii) discover patterns which represent meaningful situation within such a dynamic environment (in Chapter 5), and then (iii) to predict the future behaviour of items which will be encountered in these multiple streams (in Chapter 6). Particularly, we envisage a situation where perhaps the methods proposed are used as part of an application in order to guide drivers to different car parks, based on the predicted availability and patterns extracted from the heterogeneous data streams. So that such drivers will be able to find out, before they come to the location, how busy the parking lots are and the probability of finding a parking space when they arrive.

3.3 Experimental Framework

There are a large number of research areas where real-world experimentation is difficult or impossible to apply [13, 21, 146, 164], for example, experiments that are conducted in real-world for traffic control can be expensive and time consuming [13, 164]. Thus, many researchers have used simulations instead, in order to not just model a real-world systems and phenomena, but also to run experiments with hypothetical what-if questions that they would not normally be able to experiment with or obtain data from in reality [13, 85]. For instance, companies can use simulations to analyze future market scenarios [171], and policy makers can use simulations to understand the impact of transport policies on individuals and the whole system [13, 128].

Simulation is a tool that helps to visualize systems, investigate how systems will work under specific assumptions, and predict how systems may behave when provided with differing inputs [13, 49, 85, 263]. Researches [13, 85, 171] pointed out to the following advantages for simulations over the real systems:

- Independence from spatial dimensions, this enables investigating processes in extremely large or small systems which would be otherwise difficult to analyse.
- Time compression (or expansion), this enables observing real-world phenomena that consume very long time (or which occur too quickly) in a reasonable time frame.
- Cost-effectiveness in terms of implementation and acquiring data.
- Performing experiments that either are dangerous, or that would be impossible to do them in reality.

The smallest components of the system are referred to as micro [128, 182]. Simulating the micro-level behavior provides insights into the complex systems and finds causes of the macro-level features [182]. Arisona et al. in [13] showed

that micro simulation-based models of urban systems can be used for prediction and scenario analysis especially for transportation and land use.

Definition 3. *A Simulation* is the imitation of the operation of a real-world process over time, by representing and scenario playing through computational models.

In the next sections, we describe the simulation that has been developed as part of the experimental framework to explore problems with learning and prediction from multiple data streams, this includes a detailed description of the dataset which was chosen for this work.

3.3.1 The Simulator

For the proposed problems that emanate from multiple streams of data in domains such as the IoT, it was difficult to find a sufficiently data set generated from a streaming environment similar to the one described above in Section 3.2. We consider a smart parking system with multiple types of users etc. to be an illustration of this streaming environment, where there are a variety of IoT devices that produced heterogeneous data streams, in a dynamic environment and with multiple Imbalanced class distributions. However, such datasets are not generally made publicly available due to privacy concerns. Therefore, we decided that the best method to adopt for this investigation was to simulate an environment that generates such datasets. The aim of the simulation was to simulate drivers' behaviour when they are searching for parking spaces and to explore the patterns which emerge from the behaviour of drivers in parking lots.

There are many computational simulations tools used in the literature such as Netlogo [254, 283], Simulink [192, 250], DynaMIT [22, 23] and SUMO [64, 208]. Accordingly, the simulator NetLogo [282] was selected as a means of implementing the proposed model, involving cars and parking lots. Netlogo [282] is a multi-agent-based modelling tool which provides its own programming language, de-

signed for the creation of models [254, 283]. It was chosen because of its specific advantages [254, 283], viz: first, it is a free and open-source modelling environment for simulating small systems; second, it is a powerful tool that can be easily learned and used; third, and the most important point, it is appropriate for modelling complex systems which develop over time.

3.3.2 The Simulations Requirements

To realise the dynamic model of the parking lots environment, the simulation system must have the following facilities:

- A representation of dynamic behaviour: the simulation's main purpose is to implement the drivers' behaviour when they are searching for a car space in a parking lot, while other drivers are leaving their parking lots. Moreover, some drivers may enter the car park for drop-off or pick-up without reserving a space in the parking lots (taxi drivers commonly do this, of course). Therefore, the system must be able to simulate such dynamic system behaviour.

- Heterogeneity: the simulation must be able to generate a variety of streams which contain different kinds of data items at run-time. This feature is important when simulating the parking lots system to realise real world parking lots environment, where multiple data streams are generated, for example, cameras reading plate numbers, parking stations, entry/exit sensors, mobile app applications through which the users book places etc.

- A continuous time-driven approach: as opposed to an event-driven approach, the simulation must facilitate the process of running continuous time-driven experiments. With this approach, the simulation clock is advanced, continuously, using a fixed increment Δt. Then after each update of the clock, the state variables are updated for that time interval $[t, t + \Delta t]$. This is the most widely used approach when simulating natural systems [160].

- Scalability: the simulation must be capable of being scaled up by increasing the numbers of cars that arrive at the parking lots, without changing drivers' behaviours in relation to the parking lots.

- The simulation should be able to run different hypothetical what-if scenarios.

3.3.3 Data Set Description

As was pointed out in the problem formulation in Section 3.2, this thesis investigates the problem represented by heterogeneous data streams that are generated by different sources, using the specific case where there are multiple parking lots and different types of drivers. Accordingly, the proposed approach was applied to the IoT streams generated by a typical car parking scenario as it would be experienced in universities both in the UK and abroad. Particularly, the data from the University of Essex were used to construct this experimental setting, a map of the University of Essex parking lots [1] is shown in Figure 3.2.

Here, there are six different parking lots, and only three main entrances to all these parking lots, as follows:

[1]The map available on: `http://findyourway.essex.ac.uk/`

Figure 3.2: The map of parking lots at the University of Essex

- The first entrance leads to the North Car Park, which is open to both staff members and students, also it has special parking spaces for drop-off for the university's nursery, and a drop off/collection point for taxis.

- The second entrance leads to the Multi-deck Car Park which is intended for staff members only, and Car Park B which is used by gym members in addition to staff and students. Also, there is a drop-off/collection point for taxis at Car Park B.

- The third entrance leads to Car Park A which is designated for students use, plus the Valley car park which has special parking spaces for visitors and the Constable Building car park which is designated for on-campus hotel staff and hotel customers/visitors.

The Multi-deck Car Park is the only one which is equipped with a camera sensor for number plate recognition. However, with the increasing usage of IoT devices, it is more likely that in the near future a lot more places will be equipped with such sensors, and for the purpose of this work we assume that there is a camera sensor at each entry capable of identifying all the cars that cross the entry point. Here, the sensors were assumed to be camera sensors, due to the increasing use of camera sensors in monitoring and surveillance and their low costs and high accuracy as compared to other sensors.

In this scenario, the drivers who are looking for parking spaces, can be divided into four groups: staff members, students, gym members and visitors; staff members and students get a discounted rate on the parking fee. Moreover, there is a smart pay station located at each parking lot, where the drivers can pay the parking fee, or they can alternatively pay for parking using a (presumed) mobile phone application.

To obtain an understanding of the real-world situation in the parking lots, and in order to develop as realistic a simulation as possible, the following techniques were used:

- We approached and had in-depth discussions regarding the operation of the car parks with the Transport Policy Manager of the University of Essex, Ms. Charlotte Humphries.

- We obtained and examined a report concerning the daily usage of the Multi-deck Car Park over three months period in the summer term of 2016, from 08/03/2016 until 17/06/2016, snapshots of this report are provided in Appendix A.

- Hour-long meetings were arranged with two of the traffic officers at the University of Essex, Mr. Gary Gibbons and Mr. Joe Preston, to obtain a better understanding of the patterns that they had observed as they patrolled the parking lots on a daily basis to ensure that users had paid for parking. The full transcripts of the interviews is provided in Appendix B.

- We accessed the University's web site and investigated the official published information concerning the car parks at Colchester campus, drivers parking information, and parking rules and regulations.

According to this information, there are 1,620 parking spaces at the University; the details of these parking spaces are shown in Table 3.1. The parking charges are applied from 9:00 in the morning until 16:00 in the after noon on weekdays from Monday to Friday; at all other times, it is free to park. Furthermore, if a driver (a staff member or student) registers his/her car with the Estates Management Help Desk, s/he is entitled to pay for their parking at lower rate: 10 pence per hour or 70 pence per day, otherwise s/he must pay the normal visitors' rate. The total number of registered users (both staff members and students) in the year 2015-16 was 2,942 which is double the number of spaces available in the parking lots. Also, the number of users registered as staff members was 1,613 in the year 2015-16, exceeding the number of users registered as students which is 1,329 for the same year.

Table 3.1: The capacity of the parking lots at the University of Essex

Car Park	Capacity (spaces)
Car Park A	102
The Valley Car Park	368
the Constable Building Car Park	91
The Multi-decked Car Park	397
Car Park B	255
The North Car Park	407
Total	**1,620**

From an analysis of the report, it can be seen that that 8:00 and 9:00 in the morning are the peak hours; after these hours, the number of cars arriving decreases slightly; this is reasonable considering the structure of a typical working day, the times that lectures, seminars and other events start. On the other hand, there is a significant increase in the number of cars on Tuesdays and conversely a significant decrease on Fridays.

Furthermore, due to the fact that there is no information concerning the use of the other parking lots which are available, and in order to create a sufficiently realistic simulation model, semi-structured interviews were conducted with the traffic officers working at the University of Essex. It was found from this that there is a slight decrease in the number of cars parked at all of the lots, but especially in Car Park A and the Valley Car Park towards the end of term. While in the summer, Car Park A is nearly empty, the Valley Car Park is not used to its full capacity, conversely there is a noticeable increase in the number of visitors at the same time. However, there is no clear difference between term time and summer as far as other parking lots are concerned where most of the users are staff members. The Figure 3.3 shows how busy the parking lots are in the term time based on the information from the aforementioned interviews and the report.

In addition, the majority of the users of the six parking lots are registered users, which means they are staff members or students. About 25% of the drivers who use Car Park B are non-registered users and according to the traffic officers they are usually gym members. The officers also reported that special days at

		7	8	9	10	11	12	13	14	15	16
A Park	Monday	LO	LO	LP	LP	FO	FO	FO	FO	LO	LO
	Tuesday	LO	LO	HP	HP	FO	FO	FO	FO	LO	LO
	Wednesday	LO	LO	HP	HP	FO	FO	FO	FO	LO	LO
	Thursday	LO	LO	P	P	FO	FO	FO	FO	LO	LO
	Friday	LO	LO	P	P	FO	FO	LO	LO	LO	LO
Valley Park	Monday	LO	P	P	P	FO	FO	FO	FO	LO	LO
	Tuesday	LO	HP	HP	HP	FO	FO	FO	FO	LO	LO
	Wednesday	LO	HP	HP	HP	FO	FO	FO	FO	LO	LO
	Thursday	LO	HP	HP	HP	FO	FO	FO	FO	LO	LO
	Friday	LO	P	P	P	FO	FO	FO	FO	LO	LO
B Park	Monday	LO	P	P	FO	FO	FO	FO	FO	FO	FO
	Tuesday	LO	HP	HP	FO	FO	FO	FO	FO	FO	FO
	Wednesday	LO	HP	HP	FO	FO	FO	FO	FO	FO	FO
	Thursday	LO	HP	HP	FO	FO	FO	FO	FO	FO	FO
	Friday	LO	P	P	FO	FO	FO	FO	FO	LO	LO
Constable Park	Monday	LO	LP	LP	FO	FO	FO	FO	FO	FO	LO
	Tuesday	LO	HP	HP	FO	FO	FO	FO	FO	FO	LO
	Wednesday	LO	HP	HP	FO	FO	FO	FO	FO	FO	LO
	Thursday	LO	HP	HP	FO	FO	FO	FO	FO	FO	LO
	Friday	LO	P	P	FO	FO	FO	FO	FO	FO	LO
The North Park	Monday	LO	P	P	FO	FO	FO	FO	FO	FO	FO
	Tuesday	LO	HP	HP	FO	FO	FO	FO	FO	FO	FO
	Wednesday	LO	HP	HP	FO	FO	FO	FO	FO	FO	FO
	Thursday	LO	HP	HP	FO	FO	FO	FO	FO	FO	FO
	Friday	LO	P	P	FO	FO	FO	FO	FO	LO	LO
Multi-decked Park	Monday	LO	P	P	FO	FO	FO	FO	FO	FO	FO
	Tuesday	LO	HP	FO	FO	FO	FO	FO	FO	FO	FO
	Wednesday	LO	HP	HP	FO	FO	FO	FO	FO	FO	FO
	Thursday	LO	HP	HP	FO	FO	FO	FO	FO	FO	FO
	Friday	LO	P	P	FO	FO	FO	FO	FO	LO	LO

LO : Low Occupancy
LP : Low Peak
P : Peak
HP: High Peak
FO : Full Occupancy

Figure 3.3: The occupancy of the University of Essex parking lots during the term time

the University are usually held in the summer: e.g., graduation days and summer schools. Since Car Park A and the Valley car park have a lot of empty spaces in the summer, they are allocated for visitor use on these special days; temporary parking spaces are opened for visitors as well. A summary of the interviews is shown in Table 3.2.

Table 3.2: A summary of the interviews with the traffic officers of the University of Essex

Car Park	Park A	Valley Park	Park B	Constable Building Park	The North Park
What are the peak hours?	from 9 to 10 in the morning	from 8 to 10 in the morning	from 8 to 9 in the morning	from 8 to 9 in the morning	from 8 to 9 in the morning
Does the number of cars parked change over weekdays?and how?	increased on Tuesdays and Wednesdays	increased on Tuesdays, Wednesdays, and Thursdays	Mondays' morning and Fridays' afternoon are much quieter	decreased on Mondays and increased on Fridays	Fridays' afternoon are much quieter
What is the average parking duration?	3 to 4 hours	students spend 4 hours, staff spend 6 to 7 hours, visitors spend 2 hours	students spend 4 hours, staff spend 6 to 7 hours	staff spend 6 to 7 hours	students spend 4 hours, staff spend 6 to 7 hours
Type of drivers who used the parking lot	mostly students	mix	mix / mostly staff	staff	mix / mostly staff
What is the percentage of non-registered drivers who use the parking lot?	about 10% (most of them students live in campus)	about 10%, there is a special row for visitors and often it does not reach full capacity	around 25%	it depends on hotel's visitors	there are 11 spaces for nursery drop off
Does the number of cars parked change over term time?	pretty full at the beginning of term, then the number of cars decrease	the number of students' cars decrease	there is a slightly decrease (most staff)	no difference	there is a slightly decrease (most staff)
How busy the car park is in the summer?	almost empty	the number of visitors' cars increase and the number of students' cars decrease	no difference	no difference	no difference

3.3.4 Data Set Simulation

In the light of what has been discussed above about the difficulties in obtaining a data set that sufficiently represents a dynamic setting like these parking lots, and based on the data collected via the interviews and the analysis of seasonal and other patterns of behaviour discussed in the previous section, the activity at the parking lots at the University of Essex for the academic year 2015-2016 was simulated using NetLogo [254]. Subsequently, the 11 weeks from Monday 5 October 2015 until Friday 18 December 2015, i.e., the Autumn term of 2015-16, was chosen as the period from which the data set to be used was extracted.

Furthermore, a multinomial distribution was selected as the one to govern to the allocation of cars into parking lots. Due to the lack of information about the precise parking probabilities governing each parking, the parking probabilities were calculated based on the number of registered drivers. As has been reported by the parking officers, the percentage of non-registered drivers who park their cars in the university parking lots at any one time comes to approximately 10% of the total number of parking spaces, and this figure, coupled with the information which was available about the number of registered drivers who were staff and students, outlined in the previous Section 3.3.3, means that the percentage of staff to the total number of drivers is 53%, the comparative percentage of students is 42%, the percentage of gym members is 2% to the total number of drivers and the percentage of visitors to the total number users is 3%. There is an equal probability that a staff member parks his/her in the Multi-decked Car Park, Car Park B, the Valley Car Park or the North Car Park. Moreover, there is an equal probability that a student will park his/her car in Car Park A, Car Park B, Valley Car Park or the North car park. However, the Constable Building Car Park is allocated to a specific group of staff. Consequently, the probability used to allocate a car into a parking space, i.e. the probability that an arriving car will park in a specific parking lot, was calculated as presented in Table 3.3

Table 3.3: The probabilities used to allocate a car into a parking lot in the simulation model

Car Park	Probability
Car Park A	0.12
The Valley Car Park	0.24
Constable Building Car Park	0.03
Multi-decked Car Park	0.13
Car Park B	0.24
The North Car Park	0.24
The Cumulative parking probability	**1**

In brief, for the Autumn term, three data streams were generated and exported as MS-Excel files: the sensors stream, which consisted of 189,260 data items, the combined payment stream, from all the pay stations and also from the mobile phone application, consisting of 114,021 data items. The structures of these streams are presented in Table 3.4. Screen shots of the graphical user interface (GUI) of the simulation are provided in Figure 3.4.

Moreover, the full details of the simulation's steps and its functions are shown in Appendix C. It is worth to review the occupancy patterns at the different

Table 3.4: The structure of data streams generated from the simulation model of the parking lots at the University of Essex

Data stream	The structure of the stream
Camera sensors stream	Date Time Sensor location In or Out Car plate number
Pay station stream	Date Payment time GPS location of the station Car plate number Payment amount End of parking duration Discount (yes, no)
Mobile application stream	Date Payment time Car plate number GPS location (some users enable it, other disable it) Driver type (from user's profile) Payment amount(parking duration derived)

Figure 3.4: Screen shots of the model simulation GUI in NetLogo

parking lots as it appears in the joint of the three simulated streams, and compare it with the occupancy in the real data (previously illustrated in Figure 3.3). In the joint stream, there were 74,105 rows that represent parking information for staff members, 31,453 rows for students, 4,439 for gym members and 1,508 rows for visitors. These numbers show the imbalance in driver types in the simulated data (i.e., multi-class imbalance) . We used Tableau [60] to extract and plot parking patterns from the joint streams. The following diagram in Figure 3.5 shows the patterns appear at each parking lot over different intervals of the Autumn term. This shows that our simulation is able to produce similar data to what has been reported in the interviews, Multi-deck car park report etc.

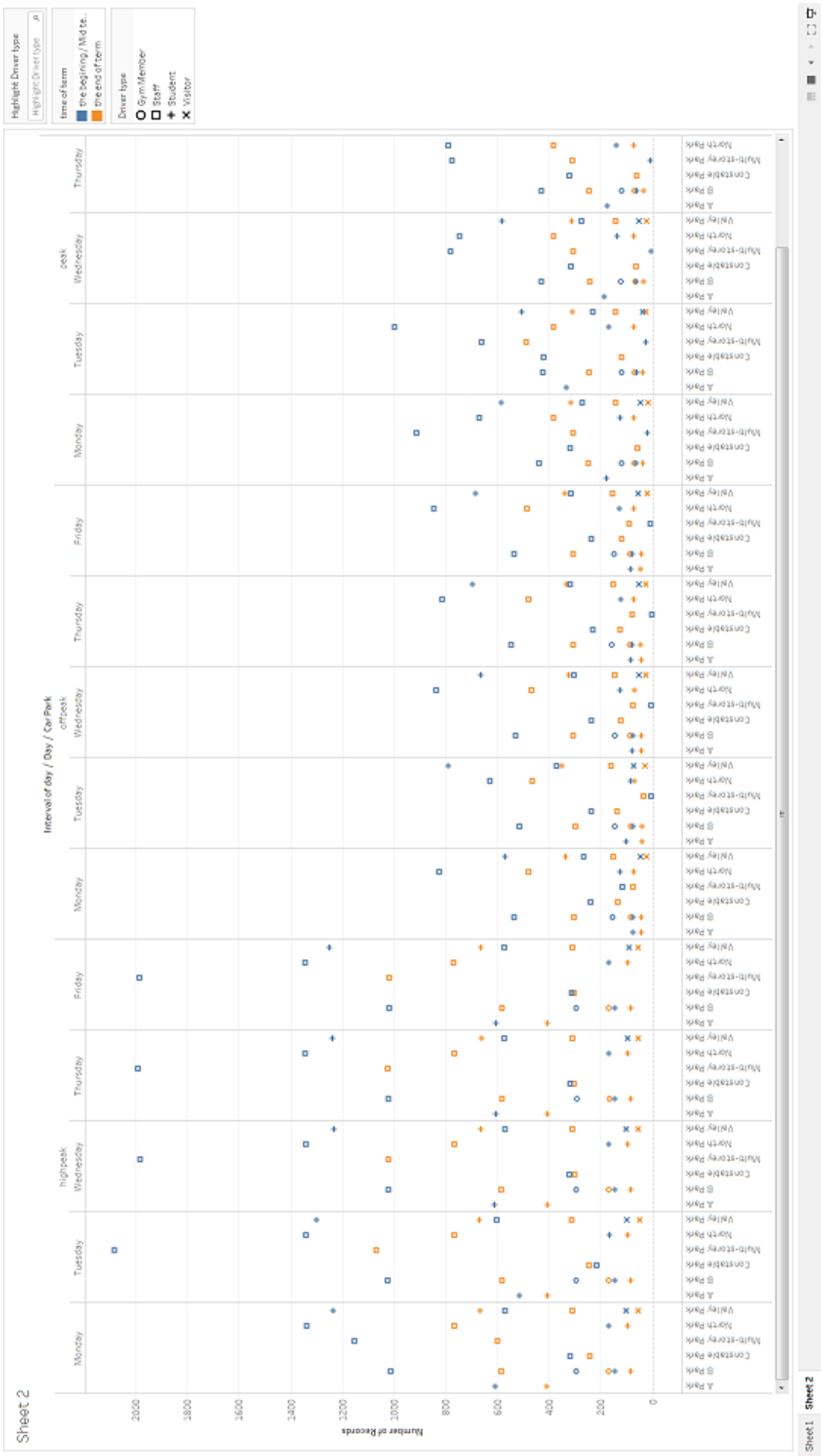


Figure 3.5: The occupancy patterns at the parking lots over the Autumn term in the simulated streams

3.4 Chapter Summary

This chapter describes the experimental framework of the present research. The contribution described in this chapter is two-fold. First, we present an abstract formulation of the problem using an illustrative scenario which relates to a smart car-parking system, where there are a set of parking lots which are all equipped with a number of differing IoT devices. Second, we develop a car parking simulation based on a real world environment to facilitate experimental work. This simulation is used to create a dataset that is close to a realistic scenario and the dataset itself exhibits multi-class imbalance suited for experimental purposes. In detail, the chapter provides a detailed description of the University of Essex parking lot usage information, then it presents a simulation model of the movements of traffic within the parking lots. This simulation was designed to produce the necessary heterogeneous data streams, which change dynamically and which exhibit multi-class imbalanced distributions (i.e., different types of users where most of users are either staff members or students, in addition to the other users: gym members and visitors). These heterogeneous data streams provided the means by which the investigations described in the following chapters were facilitated, particularly, for the development of new learning techniques which can operate on evolving data streams with skewed distributions (i.e., to classify the user type) and which can extract useful patterns from these streams and predict the future behaviour of items which will be encountered in these streams.

Chapter 4

Learning from Imbalanced Evolving Data Streams

4.1 Introduction

Machine learning and data mining models have been widely applied in industry, marketing, finance, web analytic and many other areas [94, 286]. In the past, these models have been built based on data that has been assumed to be sufficient and representative of fixed distributions [137]. However, in many-real world applications, such assumptions are unsupportable. For example, many real-world applications run in a dynamic environment (e.g., weather prediction, financial markets analysis or monitoring systems), and these environments often suffer from the concept drift problem. Furthermore, in many situations, the class distributions are skewed, resulting in the imbalanced classes problem. In data stream scenarios, the variety and velocity of the data have introduced new challenges to the established learning techniques, as these streams must be processed quickly in one or only a few passes.

This chapter investigates the combined challenges posed by evolving streams and multi-class imbalance. The complexity of these problems lies in the intersection of stream processing, concept drift and imbalanced data. The combination of

these difficulties constitutes a challenge not only when building algorithms which learn from non-stationary streams but also when evaluating their performance. Accordingly, a new approach is proposed which extends concept drift adaptation techniques into imbalanced classes scenarios, by developing an adaptive learning algorithm which uses a windows based approach. In addition, in the proposed approach, a window of the minority classes is maintained and then is used to modify the training set, this is in order to cope the imbalanced classes' problem.

The rest of this chapter is structured as follows. The next section discusses the main challenges represented by the task of learning from data streams. Section 4.3 presents a review of the techniques used for mining data streams and which are of potential use in non-stationary environments that may exhibit skewed distributions. Data stream mining algorithms can be grouped into classification, clustering or pattern mining methods [137], the focus of this chapter is solely on the classification techniques. Then, the proposed method will be described in Section 4.4. This will be followed by a presentation of the experimental work in Section 4.5. Finally, the conclusions will be discussed in Section 4.6.

4.2 Challenges in Learning from Data Streams

Learning from data streams is challenging, it is quite a different task from that of learning from conventional data sources. Specifically, the nature of these data streams presents the following difficulties when applying the conventional learning techniques and algorithms on them[31, 123, 167, 215]:

- **Real-time data processing**: data stream items arrive continuously and at high speed, and the learning algorithms must process this data in real time or near real time. Therefore, ideally, the learning algorithms should perform only one-scan on the data (this is unlike the conventional techniques which store and scan data multiple times), reducing the computational cost without compromising accuracy [215].

- **Concept drift**: Many real-world applications generate huge amounts of data in the form of data streams: web logs, sensor data, financial transaction logs, etc. And these data streams can evolve over time, so manifesting in a phenomenon called concept drift [80]. In the classification scenarios, for example, in which there are labelled data for training and then for test, some classes being trained for may change over time and other new classes might emerge. Concept drift affects the performance of any learning technique which must operate over these evolving data streams; previous models may no longer be able to classify the new data instances [166, 237]. Therefore, learning algorithms must adapt to the changes in the data stream quickly and accurately [123, 167].

- **Imbalanced class distributions**: Class imbalance is a common problem in real world applications. Typically, it occurs in classification scenarios where the classes are not all represented equally in the data set [46]. In these cases, some of the classes may be rare, appearing only occasionally; such circumstances arise in relation to applications like fraud detection, spam filtering and fault diagnosis in e.g., computer monitoring systems [48, 273]. Classifiers and the evaluation techniques associated with them tend to be biased towards the majority classes, resulting in high accuracy classificatory performance being reported despite the fact that the minority classes are largely ignored. Multi-class imbalance imposes additional challenges as compared to those encountered in relation to two-class imbalances; the situation may be exacerbated in some data stream domains by the data dynamically evolving in such a way that it is impossible to see the whole picture of data at any one time [46, 47, 273].

Moreover, it should be noted that, in particular because they come from heterogeneous sources, the streams must be pre-processed before learning takes place based on these streams, the pre-processing includes [110, 127, 221]:

- Data cleaning: data streams from sensors typically contain noise which affects the data analysis, so redundant values must be removed, missing values must be filled in, and outliers also removed.
- Data integration, the streams must be joined in order to process them in parallel over the same interval.
- Data transformation, data may have to be converted from one format or structure into another format or structure.
- Data reduction, to obtain the optimal minimum representation, in terms of volume, which produces valid analytical results.

This kind of pre-processing is not only required for the application of the technique proposed in this chapter, but it is a general requirement when one is trying to learn from multiple streams of data that come from heterogeneous sources [110, 127, 221].

4.3 Classification Techniques for Data Streams

Supervised classification is one of the most widely studied techniques in machine learning and data mining. With regards to data streams, classification techniques used on conventional databases may be extended to streaming scenarios (e.g., Bayes techniques, decision trees, or rule-based classifiers) [118], previous studies reported that some classifiers recover faster from sudden changes of concepts than others, particular, incremental learners can adapt to some extent automatically with the drift [225, 262]. Combinations of these techniques may be used to build more robust classifiers known as ensemble techniques [118].

Ensemble learners have been extensively studied in literature, and they have been adapted in many real-world problems to handle problems in data stream such as concept drift or imbalanced classes. One of the first attempts to cope with concept drift in imbalanced data streams was proposed by Wang et al. [271]. Their

work used an ensemble technique which consists of three base classifiers: decision trees, naive bayes and neural networks and analysed the impact of concept drift on the performance of each class. An explicit Drift Detection Method for Online Class Imbalance (called DDM-OCI) then used to monitor the change in the recall of the minority class in order to detect the drift. However, their work has a number of limitations; first, it only dealt with two-classes data streams and did not take multi-class imbalance into account; second, it made an assumption that the minority class is known at the beginning before applying the proposed technique; third, the drift detection method proposed has been affected by the variance of recall on the minority class throughout the learning, which led to false positive drift detections (false alarms). Another algorithm was presented in [46], a new method called Prequential AUC (PAUC) was proposed for calculating the area under the ROC curve (AUR) incrementally in imbalanced data streams. In this work, a number of classifiers were chosen: naive Bayes and Very Fast Decision Tree with naive Bayes leaves as incremental classifiers; and Dynamic Weighted Majority, Online Bagging with an ADWIN drift detector, and Online Accuracy Updated Ensemble as nsemble classifiers. Similar to the previous work [271], PAUC used independent drift detecting techniques, so the online model only retrained when a drift detected. Furthermore, it observed the overall performance rather than the performance of single class. However, a major drawback of this work that it only dealt with two-classes imbalance, and it was not clear how this method could be extended into multi-class imbalance. Later, another ensemble learner named ESOS-ELM [198] was presented. It consists of an ensemble classifier, where the minority class sample is processed by a number of classifiers depending on the imbalance ratio and the majority class sample is processed by a single classifier. Furthermore, a re-sampling technique was used to cope with the class imbalance problem and an accuracy weight votes were used to tackle the concept drift. The main drawback of this work, however, that it was impractical to apply the method proposed in real-world application, as it required the availability of an initial data

set that includes examples from all classes — before the algorithm commences. All the aforementioned methods were designed to handle the concept drift in imbalanced data streams which exhibited the presence of only two classes, but none of them addressed the multiple class imbalance [274].

To the best of our knowledge, the ensemble approach is the only method which has been applied in the literature to solving the problem of concept drift and class imbalance appearing together [137, 274]. More research needs to be undertaken to extend the other concept drift adaptation techniques which has been discussed in Section 2.5.1, specifically, adaptive base learners and learners which modify the training set into the context of class imbalance [137]. Moreover, data streams generated in real-world applications often are non-stationary streams that contain multi-class imbalance. Further solutions are required to handle concept drifts in these multi-class imbalanced data streams, because the existing techniques only work in the two-class imbalance context [46, 198, 271, 274].

4.4 ICE-Stream Technique

Here, a new technique for classifying Imbalanced Classes in Evolving Streams (ICE-Stream) is proposed. This new technique seeks to handle concept drift in imbalanced data streams, particularly, in multi class imbalanced data streams where each class can be represented by a single label. It is beyond the scope of this thesis to examine multi-label classification problems in streaming scenarios. In detail, the proposed ICE-Stream technique attempts to extend concept drift adaptation techniques into imbalanced classes scenarios. Thus, an adaptive learner (naive Bayes) is used to classify multiple streams over a sequence of titled-time windows. Moreover, the training set for the ICE-Stream technique is modified by the addition of examples of falsely classified instances (from the previous windows) in the subsequent training windows in order to discover instances of the minority classes.

4.4.1 Adaptive Learner Model

In the proposed method, naive Bayes is used as the base learning classifier. It is a supervised learning approach that assumes independence between attributes (i.e., changing the value of a feature, does not directly change values of the other features) and uses the probabilities of each attribute belonging to each class to make a prediction of another attribute class. In particular, naive Bayes is based on the Bayes Theorem, which indicates that there is a simple relationship between $p(x \mid y)$ and $p(y \mid x)$, which can be expressed as [202, 231]:

$$p(x \mid y) = \frac{p(y \mid x) * p(x)}{p(y)} \tag{4.1}$$

where the conditional probability, $p(x \mid y)$, can be computed from the conditional probability $p(y \mid x)$ and the unconditional probabilities, $p(x)$ and $p(y)$. Despite that the assumption of features independence may be violated in many real-world applications, naive Bayes has been reported in practice to work efficiently even though its independence assumption is violated [83, 172, 226].

The naive Bayes technique has been reported in the literature as being ideal for stream mining because of its incremental nature and its ability to deal with missing values; incremental learning techniques cope with concept drift faster than other learning techniques [104, 225]. It is easy to update naive model when new data stream items arrive unlike other classification approaches, for example, Decision Tree which involves an exhaustive searching and sorting of all possible answers or reconstructing the tree structure with every minor change, which is inappropriate in real-time situations. It is worth pointing out here that we used Weka [122] to find the best algorithm that fits with the data in our model and applied different mining techniques, and although decision trees (J48) produced sufficient classification results but it required reconstruction with every new stream data arrives, which costs memory and processing time. Furthermore, naive Bayes is the only generative classification method which provides a complete model of

the probabilistic structure of the data whereas other classification approaches are examples of discriminative models [104, 202]. The main difference between a generative and a discriminative model is that the latter one learns the conditional relationship between inputs, x, and the class labels, y, directly from the data. Whereas the generative model provides a complete probabilistic description of the data, by computing the joint probability of the inputs, x, and the class labels, y, and then make predictions using Bayes rules to calculate $p(y \mid x)$, to pick the most likely class label [202].

Therefore, we decided that the naive Bayes technique was the best method to adopt for this investigation, considering its ability to cope with concept drift without extra explicit change detection methods, and where the posterior probabilities could be used to calculate dynamic thresholds to identify the patterns of interest (this will be discussed in details in the following chapter).

Let the joint stream, $JStr$, described in the previous chapter, be represented by a vector $f = (f_1, \ldots, f_x)$ representing x fields (independent variables); this will give each class, C_y the probability:

$$p(C_y \mid f_1, \ldots, f_x)$$

To estimate the parameters of the naive Bayes model to be used in our approach, we utilise the Maximum Likelihood method which can be written as follows:

$$p(C_y \mid f_i) = \frac{p(C_y)\, p(f \mid C_y)}{p(f_i)} \tag{4.2}$$

and can otherwise be understood as:

$$posterior\ probability = \frac{prior * likelihood}{evidence} \tag{4.3}$$

Where:

- *posterior probability*, is the probability of the occurrence of a class ,C_y, given that set of features (i.e., predictors) occurred
- *prior*, class prior probability, is a probability of the occurrence a class ,C_y, in the dataset and which can be calculated by counting all instances that belong to this specific class divided by the overall observed target values
- *likelihood*, is the probability of the occurrence of a feature, f_i, within a class, C_y, this can be calculated by counting all instances that indicate a specific feature value (and ignore other values) divided by the overall observed feature values
- *evidence*, is the probability of the occurrence of a feature, f_i, in the dataset

The naive Bayes classifier is applied over sequential titled-time windows of the joint stream, $JStr$, on the basis of interleaved test then train chunks [29], where each arriving window-batch, $JStr_{T_n}$, of the joint stream, $JStr$, is used first for testing, and then for training the classifier. As mentioned previously in Section 2.4.1, cross validation is not applicable for data streams and so pre-quential evaluation and holdout evaluation are the basic evaluation techniques used for the data streams. Next, for each window-batch, $JStr_{T_n}$, the joint probabilities are calculated. Joint probability, JP, is a measure that calculates the likelihood of two or more items, x and y, occurring together at the same time: $p(x \cap y)$, it can calculated using the conditional probability as [202, 231]:

$$p(x \cap y) = p(y \mid x) * p(x) \tag{4.4}$$

This joint probability can be expressed, in relation to our approach, as:

$$p(C_y) \prod_{i=1}^{x} p(f_i \mid C_y) \tag{4.5}$$

where the probability that specific attributes f_1 and f_2 belong to a specific class C_y is the probability of f_1 and f_2 occurring, given that C_y occurs multiplied by the probability that C_y occurs.

4.4.2 Time-window Based Approach

One of the most widely used methods for adapting classifiers to work over evolving streams is the sliding windows approach (this is also referred to as batch-incremental) [48, 224]. This method has been extensively made use of in the literature to process data streams over sequences of windows in a way that ensures only the most recent data items are used to train the classifier. Typically, classifiers are constantly reconstructed with the arrival of a new batch of streams. In the approach proposed here, a number of different titled-time windows are used in order to detect the periodic patterns (the daily and seasonal patterns) which exist in the evolving streams across different time intervals. Using the titled-time windows approach, the most recent data are given more importance than old data, without discarding the old data items in data stream [72, 187, 214]. The structure of the titled-time window technique was described previously in Section 3.2.2.

Because of the complex situation represented by the intersection between concept drift and imbalanced classes, it is difficult to choose the most appropriate size for the windows. In general, the window size should be adapted according to the drift speed of the data stream. The use of larger window sizes is more efficient (in terms of processing resources) during periods of slow concept drift, but shorter window sizes generate better results during periods of faster drift [38, 113, 175]. Thus, small windows should be chosen during periods of fast drift, and large window sizes must be selected for periods of slow drift. On the other hand, while increasing the size of a window would increas the probability of concept drift occurring in that window [38, 113, 175], this also allows a sufficient number of examples of all classes (i.e., including instances from the minority classes) to appear

in the data stream [28, 289, 295].

4.4.3 Minority Classes Window

The minority classes problem occurs when classes are not represented equally in the data stream. For example (the worst case) when an instance of one of the minority classes appears in the test set, and none of this class' instances has been seen before in the training set. To overcome this problem in the ICE-Stream technique proposed here, the training set will be modified as processing proceeds in order to ensure that the problem of class imbalance is addressed at the dataset level. In particular, every window-batch, $JStr_T$, is used first for testing and then for training, so once the window-batch, $JStr_T$, has been used for testing, it will be taken account of, by the ICE-Stream technique, to maintain a window, W_{min}, of the incorrectly classified instances. This window, W_{min}, only stores a minimum number of falsely classified instances (i.e., if a number of identical attributes are misclassified under the same label, only one instance is stored in the window), so it does not affect other classes distributions. The choice of the number of examples maintained to represent the false classification depends on the domain of the problem, this imposes the need to have an understanding of the classes in the underlying domain. The examples from this false classification window, W_{min}, are included in the training set for subsequent windows. Moreover, W_{min} is updated on the arrival of each new window-batch $JStr_{T+1}$. This will help the system to "be aware" of the exceptional situations in (for instance) the parking lots as it maintains a record of them for future reference.

A simple pseudo code description of the proposed ICE-Stream technique is provided in Algorithm 1.

Algorithm 1: ICE-Stream algorithm for learning from imbalanced evolving streams

ICE-Stream
Input : $IoT - Stream$: joint stream from heterogeneous IoT streams
L: length of time windows specified by the user, $L > 0$
N: length of the stream specified by the user, $N > 0$
Output: $Labels$: the class labels for items in $IoT - Stream$
x= 1
$W_{min} = \emptyset$
$W_{training} = \emptyset$
for each incoming window $JStr_{T_x}$ in $IoT - Stream$ over $T_x[t_x, t_{x+L-1}]$; (x+L) $\leq N$ **do**
  $W_{training} = JStr_{T_x}$
  if $JStr_{T_x}$ is the first window in the stream **then**
    $NB - model$ = build-the-classifier ($W_{training}$, $JStr_{T_x}$)
  else
    $W_{training} = W_{training} + W_{min}$
    $NB - model$ = update-the-classifier ($W_{training}$, $JStr_{T_x}$)
    $W_{min} = W_{min}$ + the falsely classified instances ($NB - model$)
  end
  Ps = prior probabilities from $NB - model$
  CPs = the conditional probabilities from $NB - model$
  PPs = calculate-the-post-probabilities(Ps, CPs)
  $Labels$ = find-classes-labels(PPs, $JStr_{Tx}$)
  $x = x + 1$
end
return $Labels$;

4.4.4 Evaluating the Classifier Performance

There are two common approaches to evaluating the classifiers which work on streaming data:(i) Hold out evaluation, which makes use of hold out set (this includes hold out set gathered at periodical intervals) for test, and (ii) Prequential evaluation — predictive sequential evaluation — which involves the interleaved testing then training of data items (or data chunks) [106–108]. While, for prequential evaluation, each example or window of examples, is first used to test the current model, then used for training, only some instances of the stream are used for testing when using the periodical hold out evaluation method. More specifically, hold out evaluation estimates the accuracy of the classifier only on the most recent data, and it may overestimate the classifier's performance overall

if it is applied during a less volatile period. Therefore, interleaved testing then training of chunks of data [48] has been chosen for the evaluation of the classifier implemented by the proposed ICE-Stream technique.

4.5 Experimental Work

In this section, the implementation of the proposed ICE-Stream technique is shown as applied to the IoT streams generated from the parking lots setting of a typical University. Three data streams generated from a simulation of the University of Essex car parking lots scenario, which has been described in Section 3.3.3 in the previous chapter, have been used to construct the experimental setting. Theses streams come from multiple devices and demonstrate multi-class imbalance problem. Thus, it is possible to demonstrate experimentally the efficiency of extending concept drift adaptation techniques to the multi-class imbalanced evolving streams context.

4.5.1 Data Analysis and Pre-processing

For the purposes of this experiment, the simulated car movement data for the parking lots over the autumn term of year 2015-2016 was chosen. This term consisted of 11 weeks, from Monday 5 October 2015 to Friday 18 December 2015. Only working days within this period were selected for consideration; the working day was considered to start from 07:00 in the morning and to end at 16:00 in the afternoon (cleaners and maintenance staff tend to start their shifts earlier than 09:00 am).

As the streaming environment in this work is based on University parking lots, and the lectures are generally of one hour long, therefore, one-hour was selected as the appropriate time window interval. The proposed algorithm will applied on one-hour windows, these windows will not be discarded after been processed, instead, the summary of these windows will be transferred to the upper window.

When nine one-hour windows are accumulated, a one day window will be formed; similarly, five days, from Monday to Friday, will generate a one week window, and 11 week windows form the one-term window, so the maximum number of windows which the proposed approach may deal with is 51 windows per week (i.e., 45 one-hour window, five days window, and one week window). The structure of the titled-time windows over the Autumn term used in this approach is shown in Figure 4.1.

Before applying the proposed ICE-Stream algorithm, the data streams must be prepared for processing. The streams from the camera sensors, the pay stations and the mobile app are all simulated; hence the data is complete and clean and contain no redundancies. These three simulated streams, Str_1, Str_2, and Str_3 were joined (only including the cars' entry data) over the term window, based on the car registration (plate) numbers using MS-SQL. After this, the rows that contained null values were removed, and the parking durations were derived from the payment information. Moreover, as the naive Bayes had been chosen as the base classifier for ICE-Stream algorithm, and in order to prepare the joint data stream for this next step, the entry time values were converted into three intervals — peak, off-peak and high-peak. In addition, the parking duration values were converted into categorical features; it is common practice when using the naive Bayes techniques to discretise all numeric features so that all features are categori-

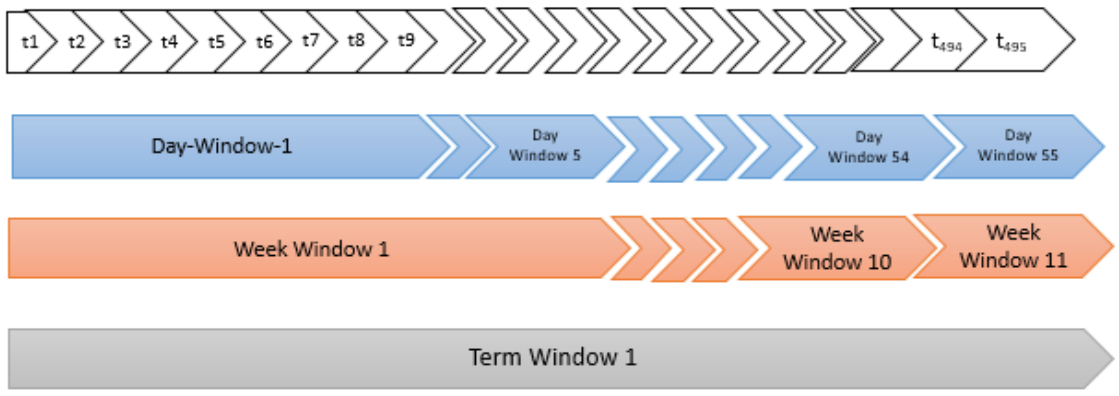


Figure 4.1: the structure of the titled-time windows in the proposed approach

Table 4.1: The details of the classes in the parking lots data set

Class	Number of instances	% of total	Class type
Class 1 (staff member)	74,105	66.45%	majority class
Class 2 (student)	31,453	28.20%	majority class
Class 3 (gym member)	4,439	4%	minority class
Class 4 (visitor)	1,508	1.35%	minority class

cal [178]. Furthermore, as we noted in Section 4.4.1, the naive Bayes assumes that given a class variable, the other features of this class are independent variables (features independence), this can be observed in the parking lots data set (i.e., given the class: type of driver; the other features: parking duration, parking location, and the camera sensor location are independent). Overall, the total number of instances to be classified in the data set was 111,505, over two majority classes and two minority classes. Table 4.1 provides the details of the classes represented in the data set.

4.5.2 Building the Adaptive Learner Model

After joining the data streams, a classification of the data items representing the cars arriving at the parking lots is sought. The type of driver must be determined, based on the collected parking information (parking location and parking duration). Accordingly, the base learner, the naive Bayes, is applied to the joint stream, $JStr$. The algorithm was implemented using the e1071 package [195] written in R [145, 253], an open source language and environment for statistical computing and graphics, and the MOA software framework [29] — which contains implementations of several state of the art classifiers and evaluation methods. Furthermore, an interleaved process of testing then training chunks of data from one-hour windows was used to evaluate the classifier, whereby each window, $JStr_T$, was used first for testing, then with the arrival of a new window, it is used for training the classifier.

4.5.3 Comparative Assessment

The main aim of this comparison is to assess the proposed method and investigate the effectiveness of applying adaptive learning technique to imbalanced evolving streams scenarios. Thus, the performance of the proposed adaptive learning technique is compared with the performance of a number of benchmark classification approaches: four of these are state-of-the-art dynamic ensemble learning methods which are designed specifically for data stream learning. Namely, the naive Bayes using interleaved chunks sourced from windows of one-hour each, is compared with the following methods: the Accuracy Updated Ensemble classifier [45], Accuracy Weighted Ensemble classifier [267], Adaptive Random Forest algorithm for evolving data stream [119], Adaptive Random Forest Hoeffding Tree [27]. These algorithms were described in details in Section 2.5.1. In addition, it compared with the K-nearest Neighbours classifier [67] which is a simple and powerful classification algorithm that uses voting from nearest neighbors and which has been used the literature for concept drif [137], the Rule classifier, the Majority Class classifier and the No-change classifier. All these algorithms are available within MOA framework [29, 32].

Table 4.2 shows the results obtained from the performance evaluation of the aforementioned algorithms; the results can be compared using Figure 4.2. Based on mean accuracy and Kappa metrics [65], κ, which allows better estimations for datasets with imbalanced classes [119], the naive Bayes was able to achieve a high learning performance, it is ranked fourth among the other approaches, with high mean accuracy and Kappa (κ) values [1] : 99.88% and 99.76% respectively. It is apparent that there is no observed difference in the performance between the naive Bayes and the ensemble classifiers; for instance, the Accuracy Updates Ensemble classifier was ranked first with 99.98% and 99.96% mean accuracy and Kappa respectively which means that it was able to classify the evolving data stream

[1]Kappa, κ, values in this chapter are calculated using the MOA framework and are shown in percentage [29]

Table 4.2: The results of the comparative performance evaluation between the classifiers

Classifier	Accuracy (%)		Kappa (%)		Time (sec)	
	last chunk	mean	last chunk	mean	last chunk	mean
The Adaptive learning classifier (naive Bayes)	99.98	99.88	99.95	99.76	0.14	0.08
Accuracy Updated Ensemble classifier	100	99.98	99.99	99.96	1.39	0.76
Accuracy Weighted Ensemble classifier	99.99	99.97	99.98	99.93	5.88	3.02
Adaptive Random Forest classifier	99.99	99.95	99.98	99.90	2.71	1.41
Adaptive Random Forest Hoeffding Tree	98.77	97.49	97.41	94.63	0.19	0.11
K-nearest Neighbours classifier	98.60	98.39	97.04	96.59	6.10	3.16
Rule classifier	84.27	74.29	62.18	29.15	5.90	2.09
Majority Class classifier	66.44	66.27	32.25	31.88	0.12	0.08
No-change classifier	52.63	52.63	4.37	4.32	0.09	0.06

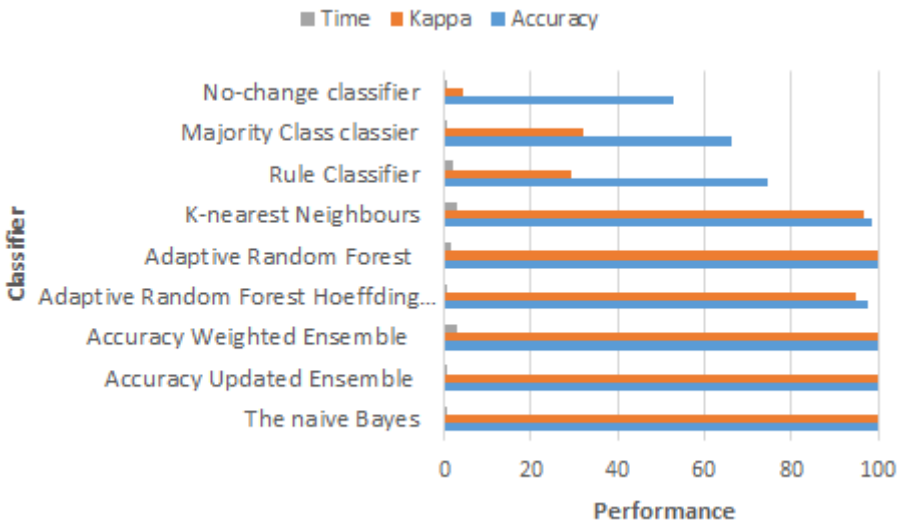


Figure 4.2: A comparison of the performance evaluation results between the classifiers

and takes the class imbalance in account. Moreover, the naive Bayes classifier surpasses the ensemble classifiers in respect of average processing time.

As the accuracy might be misleading in relation to the unbalanced data, the output of the naive Bayes is compared manually with the ground truth (the actual classes labels in the simulated stream), by comparing the predicted target values from the classifier with the actual target values. Accordingly, out of 495 one-hour time windows which were tested, misclassification occurred in 59 windows,

Table 4.3: An analysis of the naive Bayes errors over the Autumn term 2015-2016 interval (11 weeks)

Observed		**misclassified**	
windows	instances	windows	instances
495	111,505	59	147

only two misclassification cases were repeatedly observed (i.e., the two cases were repeated in 147 instances). Table 4.3 [2] summarises the analysis of the errors made by the adaptive learning technique (the naive Bayes) over the 11 weeks interval, the Autumn term 2015-2016, and a detailed confusion matrix is presented in Table 4.4. What is interesting to see from these data is the ability of the proposed adaptive learner to detect the minority classes. However, the adaptive learning technique alone is not adequate for the task of classifying all instances of the minority classes properly, i.e., 5.4% of class 4 instances are classified incorrectly. Furthermore, this adaptive learning technique also fails to discover the drift in the minority classes. The reason for this is that the adaptive learner was applied over time windows, this approach can detect/adapt with the concept drift in balanced data, but it is difficult to recognise the change in the minority classes over time, where often there are none or only few instances of the minority classes appear in the window. The next section shows experimentally how could modifying the training set affect the adaptive learning technique's performance. This experiment was attempted in order to try to improve the classification reliability.

4.5.4 Improved Detection of Minority Classes

For the purposes of improving the reliability of the implemented adaptive learning technique described in Section 4.5.2, construction of the training set was modified as follows: once a new window was received for use in testing (the naive Bayes), the misclassified instances are stored in a window, W_{min}. This window keeps the

[2] A misclassified window is which contains at least one misclassified instance, and all misclassified instances belong to the misclassified windows

Table 4.4: The confusion matrix for the naive Bayes classifier over the Autumn term 2015-2016 interval (11 weeks)

Predicted class	**Actual class**			
	class 1	class 2	class 3	class 4
class 1	74,105	65	0	0
class 2	0	31,388	0	82
class 3	0	0	4,439	0
class 4	0	0	0	1,426

minimum necessary number of misclassified instances. In this experiment, related to the University scenario, the window W_{min} keeps just a single instance for each misclassified case. Then, these instances (one for each misclassification) are all included in the next training window. After this, the window W_{min} is updated with the data from the processing of the new batch of the joint stream. in fact, out of the 495 one-hour time windows which were tested, misclassification only occurred in two windows. Table 4.5 summarises the analysis of the proposed ICE-Stream technique (the adaptive learner + W_{min} window) errors over the 11 weeks — the Autumn term 2015-2016 interval. Also, the confusion matrix for the ICE-Stream algorithm's results was calculated in and is shown in Table 4.6. As can be seen from this table, very few instances were classified inaccurately by the proposed ICE-Stream algorithm — 5 out of 111,505 instances.

To evaluate the effects of the information from the proposed window W_{min} on the implemented adaptive learning technique's performance, we compared the performance of the two variants: the adaptive learning method without the modification of the training set and the ICE-Stream method. It is apparent from the

Table 4.5: An analysis of the proposed ICE-Stream algorithm's errors over the Autumn term 2015-2016 interval (11 weeks)

Observed		**misclassified**		W_{min}	
windows	instances	windows	instances	windows	instances
495	111,505	2	5	1	2

Table 4.6: The confusion matrix for the proposed ICE-stream algorithm over the Autumn term 2015-2016 interval (11 weeks)

Predicted class	**Actual class**			
	class 1	class 2	class 3	class 4
class 1	74,105	4	0	0
class 2	0	31,449	0	1
class 3	0	0	4,439	0
class 4	0	0	0	1,507

information shown in Table 4.4 and Table 4.6, that there are only a few instances which are classified incorrectly by the adaptive learner only algorithm; however, this number is reduced further by the use of the proposed ICE-Stream method. For example, in the first week of the Autumn term, on Tuesday between 11:00 and 12:00, when some students parked in the Multi-deck Car Park, both classifiers failed to classify four instances of class 1 (student) correctly and instead classified them as class 2 (staff). The reason is that the Multi-deck Car Park is designated for staff member use, so students rarely park there: i.e., only when the other parking lots are full. However, when this situation occurred again in the same week, after nine window batches, on Wednesday between 10:00 and 11:00, the proposed ICE-Stream algorithm was able to detect the fact that students parked in the Multi-deck Car Park, and classified these drivers correctly, whereas the adaptive learner only algorithm classified them inaccurately again.

Figure 4.3 shows the comparison between the performances of the two classifiers. Typically, techniques that learn from an evolving stream are focused on the most recent data (the latest window), as it is time consuming to train the learner on all the data items which have already been seen [118, 225, 262]. As far as we know, this is the first method proposed which has shown experimentally that it is possible to learn from imbalanced evolving streams in a straightforward manner and detect the instances of minority classes within these streams by applying adaptive learning approaches and balancing the current training set by adding minority instances from previous blocks/windows. Different from other approaches

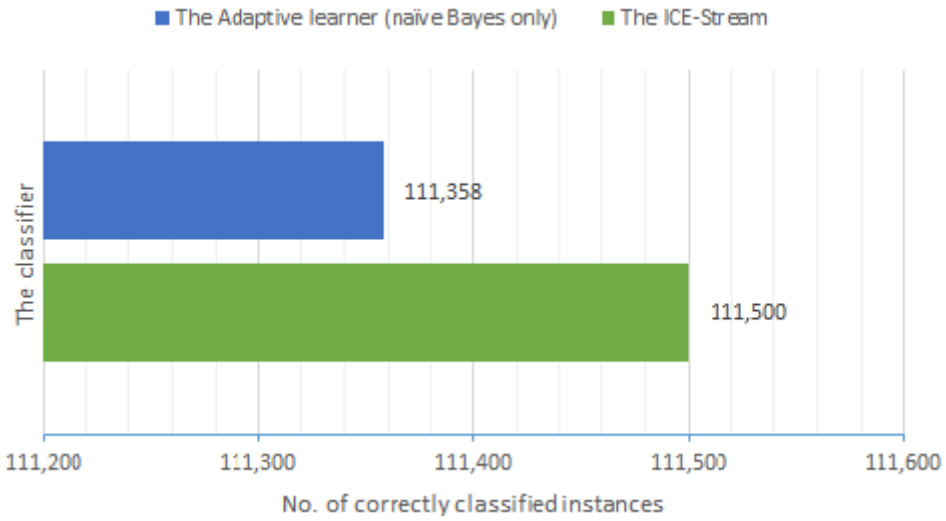


Figure 4.3: A comparison between the performance of the implemented adaptive learner and the proposed ICE-Stream algorithm

proposed in the literature which only use ensemble classifiers to cope with concept drifts in imbalanced data streams (discussed previously in Section 4.3), this study has demonstrated experimentally, for the first time, that other concept drift adaptation approaches: adaptive learning and modifying the training set can be extended in imbalance classes scenarios.

4.5.5 Analysis of Real-world Data Sets

The main goal of this experimental evaluation was to study the behaviour of the proposed technique in terms of performance and model complexity (measured in terms of learning times). In order to perform this evaluation, at first, we looked for real datasets which could be used to evaluate the ICE-Stream method. However, it proved difficult to find large-scale real-world data sets for use as benchmarks — wherein the distributions underlying the data change over time, and which exhibit the presence of multiple classes. The UCI machine learning repository [86] contains a number of real-world data sets which can be used for the purpose of evaluating machine learning techniques, and two of these real-world data sets were chosen in order to test the efficiency of the ICE-Stream algorithm: the UCI

Table 4.7: The specifications of the benchmark data sets used

Data set	No. of instances	No. of attributes	No. of Classes
The Electricity	45,312	9	2
Connect-4	67,557	43	3

Electricity data set [3] [129], and the UCI Connect-4 data set [4]. These data sets both contain at least two classes: at least one majority class and one minority class. In the following, a brief description of these data sets is presented, and a summary of their specifications is shown in Table 4.7.

- **The UCI Electricity data set:** is a popular benchmark in relation to testing adaptive classifiers [30]; it was originally acquired from the Australian New South Wales Electricity Market, and spans the two years dated from 7 May 1996 to 5 December 1998. In total, the Electricity data set contains 45,312 instances; these record electricity prices at 30 minute intervals (i.e., there are two observations per hour). There are nine attributes involved, and the class label (of each instance) makes a prediction of the next change in the electricity price (up or down) related to a moving average calculated over the previous 24 hours. The electricity prices do not remain stationary; this is due to changing consumption habits, unexpected events and also to seasonality. This data set is associated with only two classes, Up and Down, which can be considered as one majority class (Down: 26,075 instances) and one minority class (Up: 19,237 instances).

- **UCI Connect-4 data set:** this is a popular benchmark data-set used, in particular, for testing systems which deal with concept drift [94], it contains 67,557 instances and these, together, represent all the legal 8-ply positions of the game of connect-4, in which neither player has yet won and in which the next move is not forced. In all, there are 43 attributes involved, representing

[3] Available on: `https://www.openml.org/d/151`
[4] Available on: `https://www.openml.org/d/40668`

the board positions (each one being blank, occupied by the first player or occupied by the second player) on a 6x6 board. The outcome class represents the game-theoretical value in relation to the first player (2: win, 1: loss, 0: draw). Thus, the data set yields three classes, one majority class and two minority classes, Table 4.8 provides the details of these classes.

We applied the proposed ICE-Stream method to the UCI Electricity data set. The method was tested on this data set using various different window sizes, because the speed of the concept drift within the data set is unknown (the changes in the electricity prices respond to market supply and demand). Thus, three different time window sizes were selected: 5-day windows (each consists of 10 instances), 50-day windows (each consists of 100 instances) and 250-day windows (each consists of 500 instances). The adaptive learner was applied 'raw' (i.e., without the modification of the training set performed via the ICE-Stream), and then its performance was compared with that of other state-of—the-art classifiers, using the same settings. Particulary, it compared with the Accuracy Updated Ensemble classifier [45], Accuracy Weighted Ensemble classifier [267], Adaptive Random Forest algorithm for evolving data stream [119], and the Majority Class classifier. As explained earlier, all these algorithms have been implemented in R [145, 253] and within the MOA framework [29]. Table 4.9 presents the experimental settings and the results obtained from applying the proposed adaptive learner on the Electricity data set (the performance measures represent the mean value per window).

Table 4.8: The details of the classes yielded by the UCI Connect-4 data set

Class	**Number of instances**	**Class type**
Class 1 (Win)	44,473	majority class
Class 2 (Loss)	16,635	minority class
Class 3 (Draw)	6,449	minority class

Table 4.9: The results of the performance evaluation carried out using the UCI Electricity data set

Classifier	**Window length**	**Accuracy**(%)	**Kappa** (%)	**Time**(sec)
The adaptive learner (naive Bayes)	5-days	76.25	49.50	0.19
	50-days	76.18	49.27	0.15
	250-days	76.11	49.12	0.15
Accuracy Update Ensemble Classifier	5-days	81.25	60.12	1.85
	50-days	80.45	59.75	1.66
	250-days	76.41	51.63	1.57
Accuracy Weighted Ensemble Classifier	5-days	79.40	56.40	2.70
	50-days	78.60	56.29	2.25
	250-days	68.43	36.79	1.77
Adaptive Random Forest Classifier	5-days	88.83	77.02	15.25
	50-days	88.83	77.01	11.73
	250-days	88.81	76.96	4.98
Majority Class Classifier	5-days	57.46	0.09	0.09
	50-days	57.44	0.08	0.09
	250-days	57.44	0.07	0.07

Subsequently, the ICE-Stream method was applied in order to modify the training set used and then the performance of the method as a whole (including the adaptive learner) was compared with that of the adaptive learner alone. In order to understand how the window set-up proposed for the ICE-Stream method may affect classifier performance, we compared the performance of the adaptive learning technique only (with no additional process for modifying the training set) with the ICE-Stream technique itself (which includes the same adaptive learning technique plus a process for modifying the training set), in terms of the ground truth. Table 4.10 presents a comparison between the results yielded by applying these two methods on the UCI Electricity data set.

In a similar way, we applied the ICE-Stream method on the UCI Connect-4 dataset, using four different window sizes. This dataset does not contain any time-based information; it simply represents all the different possibilities in terms of board positions; so we decided to select windows based on the number of instances per window as follows: 10 instances, 100 instances, 500 instances, and 1,000 instances. As in the previous experiment, the adaptive learner was applied

Table 4.10: A comparison between the performance of the implemented adaptive learner only and the full ICE-Stream algorithm (the UCI Electricity dataset)

Window length	The adaptive learner		ICE-Stream	
	misclassified instances	time (sec)	misclassified instances	time (sec)
5-days	14,105	35.59	13,258	45.35
50-days	12,664	9.73	12,591	11.23
250-days	11,023	8.06	14,637	8.24

on its own first, and then its results were compared with other state-of-the-art classifiers using the same settings. As the previous experiment, the adaptive learner was compared with the Accuracy Updated Ensemble classifier [45], Accuracy Weighted Ensemble classifier [267], Adaptive Random Forest algorithm for evolving data stream [119], and the Majority Class classifier. Table 4.11 presents the experimental settings and the results obtained from applying the proposed adaptive learner on the UCI Connect-4 dataset (the performance measures represent the mean value per window). Moreover, the full ICE-Stream method (including training set modification) was applied and then its results compared with that of the adaptive learner alone. Then, the performance of the adaptive learning technique only (without modification of the training set) was compared with the full ICE-Stream technique (adaptive learning technique + modification of the training set), in terms of the ground truth. A comparison between the results from the two methods is provided in Table 4.12.

Table 4.11: The results of the performance evaluation on the UCI Connect-4 dataset

Classifier	**Window size**	**Accuracy**(%)	**Kappa**(%)	**Time**(sec)
The adaptive learner (naive Bayes)	10	71.38	32.51	2.17
	100	71.39	32.54	0.43
	500	71.38	32.60	0.30
	1000	71.41	32.63	0.27
Accuracy Update Ensemble Classifier	10	53.60	19.02	7.67
	100	70.73	31.00	3.31
	500	69.97	28.79	3.66
	1000	64.95	23.34	4.17
Accuracy Weighted Ensemble Classifier	10	42.11	13.30	59.51
	100	58.15	24.81	9.54
	500	57.04	23.91	5.04
	1000	56.94	22.93	3.95
Adaptive Random Forest Classifier	10	77.20	41.80	6.57
	100	78.12	44.06	72.06
	500	77.18	41.78	7.74
	1000	77.20	41.80	6.88
Majority Class Classifier	10	69.57	0.02	1.71
	100	69.56	0.01	0.28
	500	69.51	2.15	0.20
	1000	69.49	2.11	0.19

Table 4.12: A comparison between the performance of the implemented adaptive learner and the ICE-Stream algorithm on the UCI Connect-4 dataset

Window size	The adaptive learner		ICE-Stream	
	misclassified instances	time (sec)	misclassified instances	time (sec)
10	20,146	312.01	18,020	498.09
100	20,558	72.02	19,103	90.03
500	23,207	51.58	21,382	55.75
1000	24,176	47.88	21,928	50.54

4.5.6 Results and Discussion

The experiments presented in this chapter were conducted on three datasets: data streams generated from a simulation of the University of Essex parking lots, the UCI Electricity dataset, and the UCI Connect-4 dataset. The results obtained from applying the proposed ICE-Stream technique on the real (UCI) datasets matched those observed from applying the ICE-Stream technique on the het-

erogeneous data streams generated from the simulation model of the University parking lots (as discussed above).

It is apparent, from the comparisons shown in Table 4.9 and Table 4.11, that it is effective to apply the adaptive learner using a window based approach. When the ICE-Stream technique is applied on the UCI Electricity dataset, the naive Bayes is ranked fourth among the other approaches (using a 5-day window). This is comparable to the results from the first experiment on the parking lots dataset; there, the naive Bayes yielded acceptable mean accuracy and Kappa values of 76.25% and 49.50% respectively. This is close to the performance of the Accuracy Weighted Ensemble Classifier which is ranked third with mean accuracy and Kappa values of 79.40% and 56.40% respectively.

In addition, applying the ICE-Stream technique on the UCI Connect-4 dataset, made the adaptive learner (naive Bayes) the second best classifier (using a 500-instance window) with mean accuracy and Kappa values of 71.39% and 32.50% respectively. The performance of the naive Bayes exceeded the performance of two other ensemble classifiers and came second after the Adaptive Random Forest Classifier (also an ensemble classifier) which had mean accuracy and Kappa values of 77.18% and 41.78%, respectively, using the same window length. It is important to note, here, that the Majority Classifier may seem to have yielded a high mean accuracy, 59.61%, but this came with a very low Kappa value, 2.15%, which means that this classifier has poor performance as regards class imbalance.

In addition, it is clear from comparing the results in relation to both datasets, that changing the size of the window does affect the performance of the ensemble classifiers; however, the naive Bayes has a stable performance regardless of the size of the window. Moreover, this adaptive learner (the naive Bayes) is much faster, in terms of computation, than the other algorithms, where all experiments in this chapter were performed using Windows 7 Enterprise 64 Bit operating system, on i5 3.30GHz processor, and 8GB 1600Mhz DDR3 RAM memory. For example, the average execution time for the naive Bayes, on a 5-days window extracted from

the Electricity dataset and on a 500-instance window from the Connect-4 dataset, was $0.19s$ and $0.30s$ respectively; this should be compared with $1.85s$ and $2.70s$ (for the Accuracy Update Ensemble Classifier); $15.25s$ and $3.66s$ (for the Accuracy Weighted Ensemble Classifier); and $5.04s$ and $7.74s$ (for the Adaptive Random Forest Classifier). These are average execution times relating to the 5-day window and the 500-instance window, respectively.

Even though the naive Bayes classifier did not exceed the performance of the Adaptive Random Forest Classifier, this is mitigated by the fact that an ensemble technique deploys many different classifiers in order to perform its task and so every time an ensemble method is used for classifying data stream, all its classifiers must be trained and then tested, which results in much greater execution times than are achievable with just naive Bayes.

Moreover, by comparing Table 4.10 and Table 4.12, where the relationships between the classification results of the ICE-Stream with the use of the proposed minority window W_{min} and without the use of the minority window W_{min} can be seen in terms of the ground truth, it is clear that the number of instances (among the differing windows extracted from both real datasets) which were classified incorrectly was reduced by the maintenance of W_{min} and the of use this window to modify the training set for the following windows. The only exception whereby the use of W_{min} increased the number of misclassified instances, was over a relatively long window, a 250-day window from the Electricity dataset. One possible explanation for this was that it was due to the choice of the length of the time window. It seems that the drifts in the Electricity dataset generally occur over a short space of time; therefore, selecting large windows for the adaptive classifier will result in the W_{min} retaining old concepts which may will change over the subsequent intervals. This consideration is reasonable in relation to the energy consumption domain: there is a direct relationship between the season of the year and energy consumption (a season lasts between 90-120 days). These results suggest that understanding of the seasonality or domain can lead to better results

as the windows can be optimally chosen. Furthermore, these results provide further support for the hypothesis that using the proposed minority window W_{min} improved the performance of the ICE-Stream.

Furthermore, there is a noticeable difference between the performance of the proposed ICE-Stream method on the data streams produced by the simulation model, and its performance on the real datasets. The ICE-Stream method achieved high performance in relation to the parking lots dataset, both as regards Kappa and accuracy, and achieved good/moderate performance in relation to the other datasets. These differences can be explained as follows: in terms of the parking lots dataset, there was a clear understanding of the domain and the relation between the input variables and the target class, as well as of the speed of the drift; all these factors assisted in the design of a good classifier. In contrast, there was a lack of knowledge about the domain as regards the real datasets; all the attributes were used for building the classification, regardless of their relevance, and there was insufficient information about the nature and the speed of the drift (which, critically, affects the choice of window size for the adaptive learner). Another possible explanation for these differences is that, in the parking lots dataset, the data streams were generated by a simulation model, so it was to be expected that the proposed ICE-Stream method would yield better performance in relation to these streams, than in relation to real datasets; this is a reasonable proviso since all the other ensemble classifiers — used in the performance evaluation above — also achieved better performance on the parking lots dataset.

4.6 Chapter Summary

A large and growing body of literature investigates data mining techniques which operate on data streams that exhibit concept drift and/or class imbalance [46, 198, 271, 274]. Despite this, little progress has been made in relation to the multi-class imbalance problem as this may emerge in non-stationary streams. In addition,

two of the main methods used for dealing with concept drift, the adaptive learning model and modification of the training set, have not been examined yet in relation to the multi-class imbalance problem in streaming data.

In this chapter, a novel technique is presented; this is one which is capable of classifying non stationary data streams, detecting and adapting to changes in the underlying concepts being represented in these streams, and dealing with class distributions resulting in imbalance. The contribution described in this chapter is two-fold. First, we extend the adaptation techniques focused on concept drift into imbalanced stream scenarios; specifically, we developed an adaptive classifier which is capable of performing classification across multiple streams over a sequence of titled-time windows. We chose naive Bayes as the base classifier, due to its incremental nature and its ability to adapt to drift without external drift detectors [104, 225]. As the adaptive learner works using a window-based approach, the classifier will be re-trained on the most recent window of the data stream. Second, our proposed system maintains a holding window containing incorrectly classified instances; this window is then used to modify the training set to be used by the adaptive learner over the subsequent windows, in order to enhance the classifier's performance (i.e., in relation to the ability to classify instances from the minority classes correctly).

The empirical findings of this investigation clearly show the feasibility of applying the ICE-Stream method, efficiently and effectively, to the classifying of evolving streams with skewed distributions. This clearly demonstrates the usability of applying adaptive learner approaches which use modification of the training set to cope with concept drift encountered in in the multi-class imbalance scenarios. While ensemble techniques are the most common approach used for coping with multi-class imbalance problems in evolving streams, this work is the first, to our knowledge, which has shown experimentally the efficiency of extending other concept drift adaption techniques, namely adaptive learning technique and modifying the training set technique, to the multi-class imbalanced evolving stream

context. Moreover, to the best of our knowledge, ICE-Stream is the first proposal which has been developed to tackle the multi -class imbalance problem (involving more than two classes) in non-stationary streams.

Although this investigation was conducted, primarily, in relation to a car parking lots environment, the proposed ICE-Stream method can be adapted to any such real-world application, regardless of domains, where heterogeneous data streams are produced by dynamic environments. It should be said, however, that the generalisation of the results obtained in this investigation is subject to certain limitations. For instance, the performance of the adaptive learner is affected by the choice of window size. It might be difficult to find a compromise choice between small windows which adapt quickly with concept drifts, and large windows which lead to a good generalisation, in cases where there is a lack of domain knowledge. Another limitation is related to how the minority window in the ICE-Stream method is maintained, specifically, in terms of deciding when items in the window should be discarded. Minority classes may evolve over time and accordingly the instances stored in the minority window (i.e., instances which belong to the old concept) may affect the classifier performance (i.e., by including instances of old concepts in the training). Furthermore, for the purpose of evaluating the performance of the proposed ICE-Stream, we used Kappa statistics [65] which take class imbalance into account, calculated the accuracy, and compared the results to the ground truth. Further investigations to find measures for evaluating not only single aspects of stream mining algorithms, but also ways of combining several aspects (i.e., concept drift, imbalanced distributions) into the evaluation procedure, will need to be undertaken [168]. More performance evaluation metrics for data stream are required which are sensitive to multi-class imbalance [31] and which can detect change in more than one minority class.

Chapter 5

Pattern Discovery from Heterogeneous Data Streams

5.1 Introduction

In the field of data mining, pattern discovery has become a powerful tool for extracting valuable information from mass data sources [90, 116, 125]. In the literature, research into pattern mining has focused mostly on the mining of frequent item sets [125]. Apriori [6] was the first algorithm proposed for the purpose of mining frequent patterns; it was designed for supermarket basket analysis. Subsequently, many other algorithms have been proposed which do not use candidate generation, such as the FP-Growth algorithm [123]. The rare pattern dilemma [179] has been a major issue in these studies [113, 123, 156, 179]; single or multiple fixed thresholds were used to identify the frequent patterns over the entire data set, as a result, patterns that appeared in only a few data items, i.e. rare patterns, were not captured [179]. In many real-world applications, the rare patterns encountered can convey a great deal of information of interest. For example, the detection of rare items can help to prevent fraud in financial transactions [277], identify diseases in medical domains, or detect faults via monitoring systems [264]. Pattern mining is more challenging in streaming scenarios, where algorithms may

only make a single pass over the data, and where, often, only some of the data stream items are available when the pattern mining process commences. The situation is aggravated when the environment is non-stationary and the streams have skewed distributions; the frequency counts of item sets may change significantly over time, resulting in the appearance of new patterns or in changes to existing ones [132, 141, 142].

This chapter presents a new method for learning Frequent Patterns from imbalanced Evolving Streams (FP-EStream). A dynamic support threshold is proposed in order to facilitate the discovery of frequent patterns within such streams. The proposed algorithm is capable of detecting rare patterns, as well as of handling any concept drift in the emerging patterns efficiently. The reader should bear in mind that we do not intend to investigate the frequent items from the various streams in any detail; instead, the main aim of this investigation is to develop methods for discovering patterns, found in imbalanced evolving streams, which represent meaningful real-world situations.

The rest of this chapter is structured as follows. The following section presents the background of pattern mining as it is relevant to this chapter. Section 5.3 describes the details of the proposed FP-EStream method. Then the experimental work and results will be discussed in Section 5.4. Finally, the chapter concludes by providing a summary of the work undertaken in Section 5.5.

5.2 Basic Concepts

It is necessary here to clarify and make more precise some terms that are used in this chapter before moving on to present the proposed FP-EStream method.

Definition 4. *A Pattern*, P, from the data stream, $JStr$, over the time interval, T_n, is defined as the set of data items, $pi_1, pi_2, \ldots, pi_k$ that represent a particular information element extracted from the most recent window, RW, of the data

stream, S:

$$P = \{pi_1, pi_2, \ldots, pi_k\}$$

where

- k is the number of data items in P: i.e., P can be termed a $k - itemset$, $k \geq 1$;
- $|RW|$ is the size of the window, RW: i.e., the number of recent data stream items (tuples) considered for processing;
- $Count_{RW}(P)$ is the number of stream data items (tuples) that were found to contain P in the recent window RW.

Then, the minimum support threshold is a measure of the user's interest in item sets whereas the support of pattern, $Sup_{RW}(P)$, is defined by the ratio of the number of stream items in RW that contains P to the total number of data items in RW.

$$Sup_{RW}(P) = \frac{Count_{RW}(P)}{|RW|} \tag{5.1}$$

Definition 5. A pattern, P, is considered to be a *Frequent Pattern* if its support, $Sup_{RW}(P)$, is equal or exceeds a predefined support threshold, $Threshold$.

$$Sup_{RW}(P) \geq Threshold \tag{5.2}$$

Definition 6. A pattern, P, is considered to be a *Rare Pattern*, if its support, $Sup_{RW}(P)$, is less than a predefined support threshold $Threshold$.

$$Sup_{RW}(P) \leq Threshold \tag{5.3}$$

The rare pattern dilemma is often encountered when mining for patterns in real-world application data. Generally, if we pick a high value for the minimum support threshold, this will reduce the number of patterns extracted: i.e., the mining technique will not be able to detect rare patterns. In contrast, selecting a low value for the minimum support threshold will generate a large number of insignificant patterns — leading to high processing costs[155, 179].

5.3 FP-EStream Technique

Extracting the interesting patterns from evolving data streams is difficult and time consuming since a huge number of patterns can emerge from such data; therefore, patterns must be identified according to their level of interest to the user. For the purposes of this thesis, the FP-EStream algorithm has been proposed. This method is based on an integration of two mining techniques; a classification technique which identifies the patterns of interest (described in the previous chapter), and a pattern mining technique based on a modified version of the FP-Growth algorithm, which uses a dynamically calculated support. An abstract model of this approach to pattern discovery from evolving IoT data streams is shown in Figure 5.1.

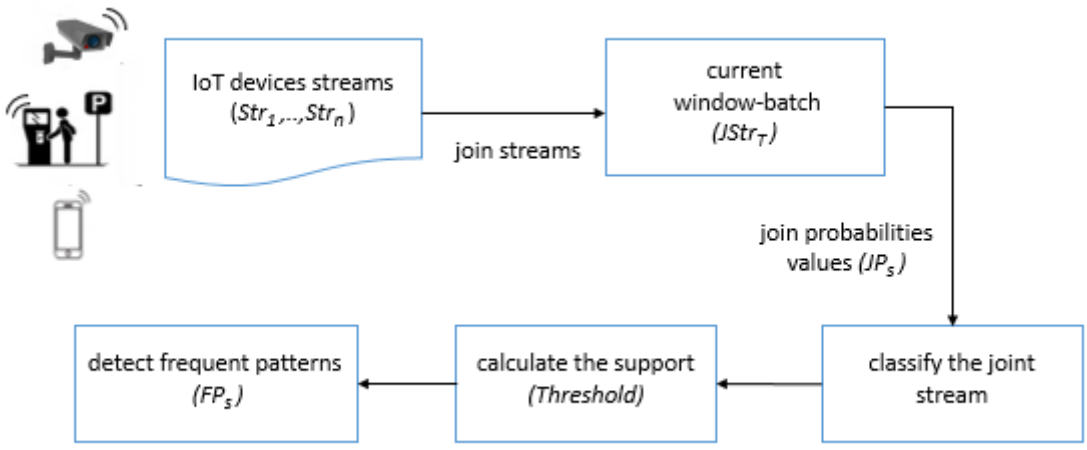


Figure 5.1: Pattern Discovery Abstract Model

5.3.1 Dynamic Support Threshold

Defining the support threshold value is a significant element in the proposed pattern discovery approach; a dynamic support threshold has been chosen as a means to detect the patterns of interest. It is typical of evolving, skewed streams that some items appear very frequently in the data set while others only rarely appear, and some of the initially frequent items may become less frequent over time. Therefore mining patterns from these streams using a fixed support threshold is insufficient; as stated before, choosing a low threshold value will identify a large number of meaningless patterns, whereas a high threshold value may omit some significant patterns.

Based on the definition of the concept of a support value given in Equation 5.1, and the definition of joint probability, JP, given in Equation 4.5, which indicates the occurrence of items together, the minimum joint probability indicated by the adaptive learner (implemented in the previous chapter) is selected to be the dynamic support threshold value for FP-EStream algorithm, as follows:

$$Threshold = min(JPs_{RW}) \tag{5.4}$$

Accordingly, the minimum joint probability of the recent window batch, RW, which is calculated by the naive Bayes classifier, is chosen to be the support threshold value of this window Sup_{RW}; this value will thus be updated for each window batch, $JStr_T$. This method of calculating the support dynamically can be used with real-time or near real-time data since these calculations depend only on the current and previous time windows (the previous time window is used to train the naive Bayes classifier).

5.3.2 Extracting Patterns from Imbalanced Evolving Streams

The proposed FP-EStream technique is based on the FP-Growth algorithm [123], which is discussed previously in Section 2.5.3 and which is used widely for static

data, and in order to make it applicable to imbalanced evolving streams, two main modifications were made:

1. The use of a dynamic support threshold value, $Threshold$, instead of the fixed threshold value used in FP-Growth.

2. The use of the naive Bayes technique to construct and update the FP-tree. This represents a significant difference in relation to the original FP-Growth algorithm (which scans the data set twice, first to get the frequencies then to find the frequent patterns). The joint probabilities, which are calculated by the naive Bayes classifier, are used to indicate the frequencies and so to construct the frequent items list, FIS. Accordingly, the frequent item list is constructed based on the classifier fields $(f_1, \ldots, f_x, class)$ in a descending order, this helps to ensure the tree maintains as much prefix sharing as possible. Subsequently, the items in the recent window are compressed into a tree of frequent patterns $FP-tree$, derived from the frequent items list, FIS. Each path in the tree of frequent patterns, $FP-tree$, represents a pattern. It is important to note here that, the frequent item list, FIS, and the tree of frequent patterns $FP-tree$, are constructed for the initial window, RW. Then they are updated with the arrival of each window-batch $JStr_{Ti}$; this reduces the cost in terms of processing time and data-storage needed to scan the data set.

Here, frequent pattern sets can be maintained for each titled time window, an example of the structure of the FP-EStream proposed is shown in Figure 5.2. Using this approach will help to identify frequent patterns over the period T (i.e., time window), also it can be used to find periods which within specific patterns became frequent or changed (i.e., become less frequent). Furthermore, a description of our proposed FP-EStream algorithm is given in Algorithm 2.

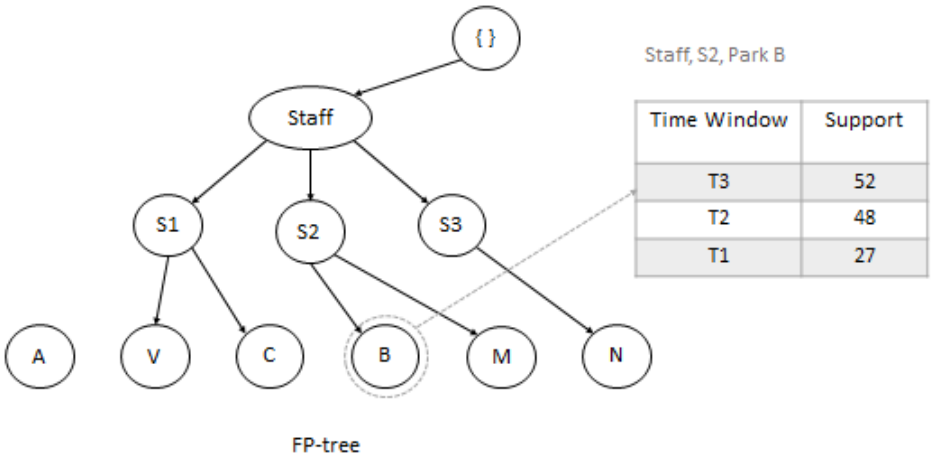


Figure 5.2: An example of the structure of the FP-EStream proposed

Algorithm 2: FP-EStream algorithm for pattern discovery from imbalanced evolving streams

FP-EStream
Input : $IoT - Stream$: joint stream of heterogeneous IoT streams
L: length of time windows specified by the user, $L > 0$
N: length of the stream specified by the user, $N > 0$
Output: FPs : Frequent patterns from $IoT - Stream$
x= 1
for each $JStr_{T_x}$ in $IoT - stream$ over $T_x[t_x, t_{x+L-1}]$; (x+L) $\leq N$) **do**
  apply $NB - model$ on $(JStr_{T_x})$
  Ps = prior probabilities from $NB - model$ $(JStr_{Tx})$
  CPs = the conditional probabilities from $NB - model$ $(JStr_{Tx})$
  PPs = calculate-the-post-probabilities(Ps, CPs)
  JPs = calculate-the-joint-probabilities(Ps, CP)
  $Threshold$ = find-minimum-joint-probability(JPs)
  if $FIL = \emptyset$ **then**
    FIL = build-frequent-item-list(JPs)
    Build $FP - tree$ $(JStr_{T_x})$
  else
    Update FIL
    Update $FP - tree$
  end
  FPs = mining-the-frequent-patterns ($Threshold$, $FP - tree$)
  $x = x + 1$
end
return FPs;

5.4 Experimental Work

In this section, the experimental work started in Chapter 4 is continued, using the simulated streams based on the University of Essex parking lots setting. The naive Bayes classifier has already been applied and evaluated via the ICE-Stream algorithm, as shown in the previous chapter. In brief, an adaptive learner had been applied to one-hour-window batches, and based on this naive Bayes model, the joint probabilities have been calculated for each window. Table 5.1 presents the characteristics of the parking lots data set.

5.4.1 Pattern Discovery Model

The FP-EStream algorithm was implemented using the R [253] and Java programming language. In particular, the rJava package [260] was used to integrate Java code with R code (for the ICE-Stream model). Moreover, the modified FP-Growth technique was implemented based on the implementation of the FP-Growth algorithm found in the SPMF library, the Java open-source pattern mining library [95].

Once the most recent window has arrived, the joint stream is classified by the ICE-Stream technique and the joint probabilities are calculated. Then, the minimum value of the joint probabilities is selected to be the support threshold for extracting the frequent patterns of this window. After that, for the first scanned window only, the frequent items list, FIS, is built using the joint probabilities. The items are ordered based on the classifier attributes (sensor, parking location, driver type) in descending order. Then, the tree of frequent patterns $FP-tree$ is constructed based on the classified window — in accordance with

Table 5.1: The characteristics of the University parking lots data set

Number of windows	Number of tubles	Number of items
495	111,505	21

Table 5.2: Details of the patterns which emerge from over the University parking lots data set

Threshold	Number of patterns	Number of rare patterns
$0.001 \leq threshold \leq 0.01$	12	5

the frequent items list FIS. For the subsequent windows, the frequent items list FIS is updated according to the joint probabilities then the tree of frequent patterns, $FP - tree$, is updated as well.

A number of different patterns were extracted from the 495 one-hour windows. Table 5.2 shows the details of the patterns extracted over these intervals. Overall, the number of patterns extracted ranges between 8 and 11 patterns per one-hour window — where the dynamic threshold varies from 0.001 to 0.01. There are 12 different patterns altogether; two of them are rare patterns, and there are another three frequent patterns that become rare over some windows.

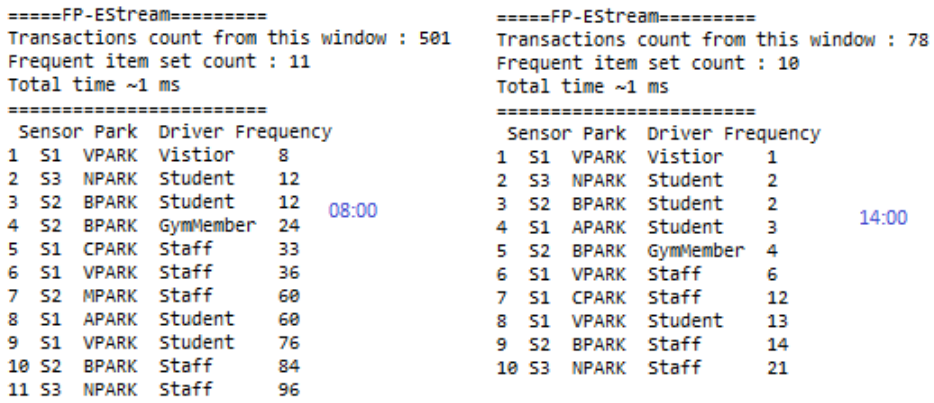
In addition, screen-shot in Figure 5.3 shows the emerging patterns encountered on Monday morning, between 09:00 and 10:00 during the first week of October in the Autumn term 2015-2016.

In relation to this same Monday, Figure 5.4 shows the difference between the patterns which emerge at the beginning of this working weekday, from 08:00 to 09:00 in the morning, and those that emerge at the end of it, from 14:00 to 15:00

Figure 5.3: The emerging patterns over a one-hour window, in the first week of the Autumn term 2015-2016

```
=====FP-EStream=========
Transactions count from this window : 503
Frequent item set count : 11
Total time ~1 ms
=======================
 Sensor Park  Driver Frequency
1  S2  BPARK  Student    9
2  S1  VPARK  Vistior    10
3  S3  NPARK  Student    12
4  S1  CPARK  Staff      14
5  S2  BPARK  GymMember  18
6  S1  APARK  Student    28
7  S1  VPARK  Staff      45
8  S2  BPARK  Staff      62
9  S3  NPARK  Staff      96
10 S1  VPARK  Student    97
11 S2  MPARK  Staff      112
```

Figure 5.4: The changes in the emerging patterns over a one day interval of the Autumn term 2015-2016

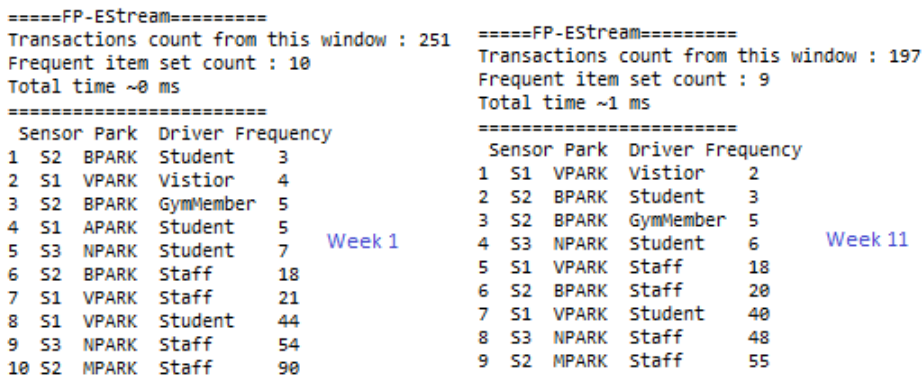


in the afternoon. Furthermore, there is a notable change in the patterns which emerge at the beginning of a term and those that emerge at the end of the same term; for example, Figure 5.5 shows the patterns which emerged on two Thursday mornings from the same interval from 10:00 to 11:00 on each day. These Thursdays are the first and last week of the Autumn term, 2015-2016. By investigating the patterns obtained over different intervals, daily, weekly and at different points over the Autumn term, it becomes apparent that these patterns are consistent with those in the real data (as understood from interviews and reports). For example, as Figure 5.4 shows, there is a significant decrease in the car arrival rate in the afternoon. In addition, as shown in Figure 5.5, there is a considerable drop in the car arrival rate at the end of the term, particularly in Car Park A, the student's car park.

5.4.2 Comparative Evaluation

To validate the effectiveness of the dynamically calculated support threshold used in the proposed FP-EStream method, we compared the number of patterns detected from the joint IoT stream $JStr$ using a low fixed threshold value, a high fixed threshold value, and the proposed dynamically calculated threshold value — over different window-batches. A number of windows from the investigated

Figure 5.5: The changes in the emerging patterns over the Autumn term 2015-2016



dataset were selected and were processed using these different threshold values; the patterns which were detected by these procedures were then compared. For this comparison, the nine consecutive windows from 07:00 in the morning until 16:00 in the afternoon, on Wednesday of the first week of the Autumn term, from the joint stream, $JStr$, were chosen. It is important to note here, that windows were selected from the beginning of the term (the third day out of 55 days), this to ensure that FP-EStream has been trained on some windows (but not a significant number of examples), and in order to evaluate how the FP-EStream proposed works when new patterns emerge/change. We attempted the detection of the frequent patterns in these window-batches using the following three techniques:

1. FP-Growth algorithm with a low fixed threshold ($Threshold = 0$)
2. FP-Growth algorithm with a high fixed threshold ($Threshold = 0.05$)
3. The proposed technique FP-EStream, where the support threshold is dynamically calculated ($Threshold \in [0.001, \ldots, 0.01]$)

The FP-Growth algorithm was applied on the joint stream, $JStr$, using the SPMF library [95], and in order to obtain comparable results, the inputs provided to FP-Growth were the same inputs which were provided to the classifier used in the proposed FP-EStream technique. A comparison between the three methods is shown in Figure 5.6. It can be seen clearly that the number of patterns extracted

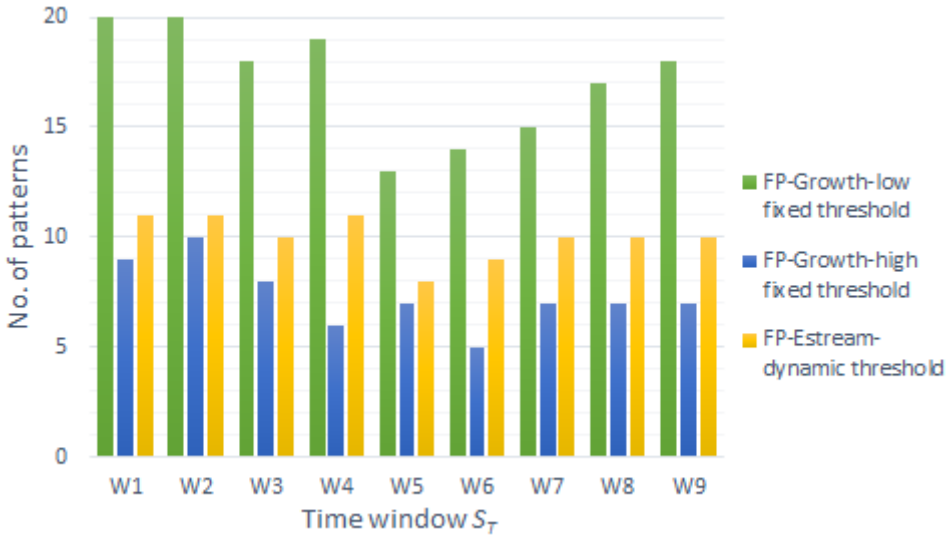


Figure 5.6: A comparison between the number of patterns extracted using FP-EStream and FP-Growth on the parking lots data set

by the low threshold ($Threshold = 0$) was nearly double the number of patterns which were extracted using the dynamic threshold. However, some of the patterns which were found using threshold ($Threshold = 0$) were insignificant (e.g., pattern consists of one item) and so would dictate the use of an additional pruning step. By contrast, the number of patterns extracted using the high threshold ($Threshold = 0$) was less than the number of patterns extracted using the dynamic threshold: the high threshold failed to detect any pattern relating to visitors, gym-members or Car Park A. In general, choosing a fixed low threshold increases the number of detected patterns; however, such an expanded pattern set would require an additional filtering step. On the other hand, choosing a high threshold not only decreased the number of detected patterns, but also failed to detect any of the rare patterns. It is clear from the graph, that the number of patterns discovered by the proposed dynamic threshold is in between the numbers discovered by the other two; the use of a dynamic threshold avoids the returning of too many insignificant patterns and also enables the detection of rare patterns. Interestingly, the use of this threshold identified all the patterns which were detected by the use of the lower threshold (of 0), including the rare patterns (but no filter needed).

Speed is the main performance measure which must be monitored in relation to the mining of frequent patterns. In this regard, all experiments in this chapter were performed using Windows 7 Enterprise 64 Bit operating system, on i5 3.30GHz processor, and 8GB 1600Mhz DDR3 RAM memory. The classifier used in our method, as opposed to those used in the multiple conditional tree generation and test approaches, narrows the number of derived patterns down to only the ones which may be of interest — without the need for repeated scans. This reduces the cost of the stream scanning process, and the pruning search, it also speeds up the mining process as well. The average time required to discover patterns from a one-hour window derived from the first week of the Autumn term using the FP-EStram is $5ms$, faster than the average time required for the same interval by the FP-Growth algorithm, $11ms$ ($Threshold = 0$). Moreover, when applied to the whole term, our algorithm took $7.162s$ while the FP-Growth algorithm took $14.268s$. It is quite likely that this will scale up directly (i.e., as window size increase, the difference of the speed between the two algorithms increases too), so that these efficiency improvements (resulting from the application of the FP-EStram algorithm) will pertain to its application to larger data sets in real-time.

Moreover, we use the same interval above (the nine windows) to compare the performance of the FP-EStream technique proposed here with two state-of-art algorithms: the CFP-Growth++ algorithm [156] which use multiple (fixed) minimum support thresholds to mining frequent itemsets, and the RP-Growth [258], an adaptation of the FP-Growth which presented for discovering rare item sets in transaction databases (using two threshold values: Minsup and Minraresup). These two algorithms were discussed in detail previously in Section 2.5.3, furthermore, they were implemented for this experiment using the SPMF library [95], and the settings used as follows:

- For the CFP-Growth++ algorithm, in order to build the list of minimum support threshold, it was not clear how the minimum value for each item should be selected. Therefore, the minimum values were chosen to be nearly

equal to the frequency value of the least occurring element, for instance, the appearance value of "visitor" item varies from zero to six per window, so the minimum support values were chosen to be five. Furthermore, the items in each tuple were sorted by lexicographical order (algorithm requirement that items are assumed to be sorted in the transaction by lexicographical order), and the minimum confidence was chosen to be %50.

- For the RP-Growth algorithm, the items in each tuple were also sorted by lexicographical order, then the minimum support value (upper boundary of what is considered rare) and the minimum rare support value (the lower boundary of what is considered rare) were chosen to be 0.05, and 0.001 respectively. The choice of the minimum rare support value was based on the observed value in our proposed algorithm, and the choice of the minimum support value was based on the high threshold used in FP-Growth experiment (we tried the highest minimum threshold 0.01 but it gave zero patterns), important to note here that using the maximum value of the dynamic threshold as the minimum support value in the RP-Growth results in zero rare patterns in this experiment.

The result of the comparison between these methods and the proposed FP-EStream technique is shown in Figure 5.7. It is clear from the bar chart that the number of patterns which are obtained using our proposed FP-EStream Algorithm is less than the number of patterns obtained by the other two methods. It is apparent from investigating the patterns obtained (using the three methods over nine hour-window interval), that the patterns obtained from the RP-Growth algorithm and the CFP-Growth++ algorithm contain large number of trivial patterns. What is interesting in this comparison, is that the number of all patterns extracted using our proposed method, over nine hour-window interval, is less than the number of patterns obtained using the RP-Growth algorithm over the same windows (the RP-Growth algorithm is designed for discovering the rare patterns

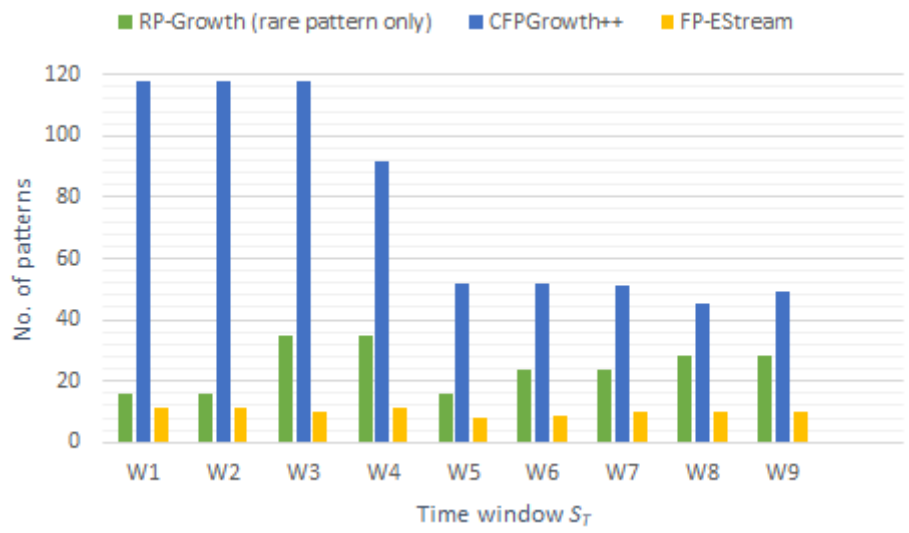


Figure 5.7: A comparison between the number of patterns extracted using FP-EStream, CFP-Growth++, and RP-Growth techniques on the parking lots data set

only). This result may be explained by the fact that both the CFP-Growth++ algorithm and the RP-Growth algorithm used fixed support values: the CFP-Growth++ algorithm used a list of predefined minimum support threshold; and the RP-Growth algorithm used two fixed values as the minimum support and the minimum rare support; thus, as the support value decreases, the number of frequent item sets increases exponentially. It is important to note that windows used here were chosen across different intervals (in the morning in the afternoon), these windows have different patterns, including rare patterns. Therefore, it is likely that results will apply across different windows as the *Threshold* is computed dynamically for each window.

5.4.3 Analysis of a Real-world Data Set

In order to evaluate the FP-EStream method proposed here within more challenging settings, we applied FP-EStream on a real-world benchmark data set, the UCI Connect-4 data set, which was described earlier in Chapter 4. In the last chapter, four different window lengths (10, 100, 500, and 1000 instances) were used during the classification of this Connect-4 dataset. The instances of this dataset consist

of 43 attributes and applying FP-Growth on this data set (over any significant window size) generates very large numbers of patterns. Thus, we decided to test the FP-EStream only on five consecutive windows (each window consists of 10 instances); this was to simplify the comparison of the results with those of the other methods. Accordingly, the first five windows were selected and then the proposed FP-EStream was tested using these windows and then its performance was compared with that of both the FP-Growth technique, and the RP-Growth technique as follows:

- In the FP-EStream technique, the $FP-list$ is built for the first window, based on the class attributes (class, G6, G5,..., A2, A1) — generated by the ICE-Stream classifier in the previous chapter. As explained previously there, these attributes represent the board positions (each one being blank, occupied by the first player or occupied by the second player) on a 6x6 board, whereas class represents the game-theoretical value in relation to the first player (2: win, 1: loss, 0: draw). The $FP-list$ was then used with the dynamically calculated support, the $Threshold$, to build the $FP-tree$, which then could be mined, instead of the whole data set, in order to discover patterns in the window. With the arrival of each new window, the $FP-list$ and the $FP-tree$ were updated, as well as the minimum joint probability values which were calculated for each window (using the priors and the conditional probabilities from the naive Bayes classifier). These latter were then employed to identify the $Threshold$ for the current window. In fact, over this interval, the $Threshold$ value varied from 0.05 to 0.1.

- In terms of the original FP-Growth technique, we selected the minimum support value based on the observed $Threshold$ values by our proposed algorithm. Accordingly, we tested the minimum support value, 0.1, which is the highest $Threshold$ value returned by our proposed algorithm. however, it has been clearly observed that applying the FP-Growth algorithm takes

a significant amount of computational time, for instance, by selecting 0.1 as the minimum support value, the FP-Growth algorithm needed 18.29 minutes to extract 53,194 patterns from one window (the first window W1). Therefore, we sought for testing a higher values (minimum support values) in order to reduce the computational time (i.e., increasing the minimum support value will reduce the scanning time, as items that have supports under the minimum support value will be ignored). Specifically, the following support values (0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8) were tested for this exercise in order to identify the most appropriate level. We observed that when we increased the minimum support value, the number of extracted patterns and the required time were decreased. For instance, using the minimum support value, 0.8, the FP-Growth algorithm required 4.3 minutes to extract 36,513 patterns from the same window, W1. As a result, we tested a higher support value: 0.85; using this threshold, the FP-Growth algorithm required between $16ms$ and $64ms$ (i.e., this is comparable to the time required by our proposed algorithm) to extract number of patterns ranges from 30 to 78 patterns per window. The choice of threshold 0.85 as the minimum support value allowed for the extraction of large number of patterns in shorter time than the other tested values (i.e., lower support value would increase the number obtained patterns to unsupportable levels and reduce the algorithm's performance).

- In terms of the RP-Growth technique, the items in each tuple were sorted by lexicographical ordering, and then the minimum support and the minimum rare support values were selected as 0.1, and 0.05 respectively. The choice of these values was based on the observed values yielded by our proposed algorithm: the highest and the lowest $Threshold$ values returned.

The comparison between the numbers of patterns obtained using the three techniques, with the above specification, over five consecutive windows is provided

by the bar chart in Figure 5.8, this will be discussed in detail in the following section.

5.4.4 Results Discussion

The experiments presented in this chapter were conducted on two data sets: data streams generated from a simulation of the University of Essex parking lots, and the ICU Connect-4 data set. The results obtained from applying the proposed FP-EStream technique on the real data set (the Connect-4 data set) support the findings obtained from applying the FP-EStream on the heterogeneous data streams generated from the simulation model of the University parking lots (discussed above).

It is apparent, from Figure 5.8, that the number of patterns extracted by our proposed algorithm was less than the number extracted using the FP-Growth algorithm. Moreover, by investigating these patterns, we found that the ones our method extracted each consisted of eight or nine items, whereas the length of the patterns extracted using FP-Growth varies between two to 11 items. This can be explained by the circumstance that each instance in the UCI Connect-4 data

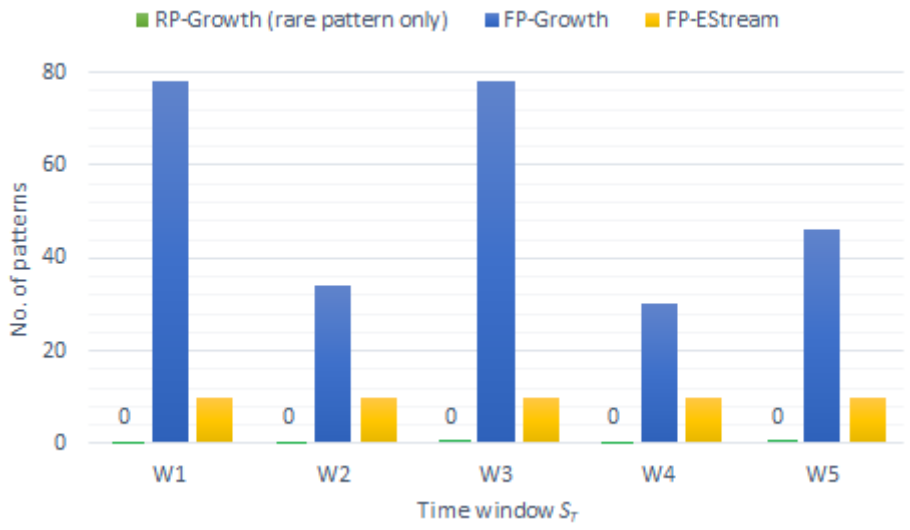


Figure 5.8: A comparison between the number of patterns extracted using the FP-EStream, the RP-Growth, and the FP-Growth techniques on the UCI Connect-4 data set

set can exhibit up to 43 attributes, one representing the class and each one of the remaining 42 attributes taking one of three values (blank, first player, second player); so although our proposed algorithm used a low value for the *Threshold*, it only extracted the positions which exhibit both player and class; for instance, the simplest case of such a position is where the two players choose to make parallel lines as in the image [1] in Figure 5.9.

In contrast, it was somewhat surprising to discover that the RP-Growth algorithm — using the same threshold values that were derived by our algorithm — did not discover any pattern. As explained in Section 2.5.3, the RP-Growth algorithm builds the RP-tree based on item sets which have support values in the range [Minraresup, Minsup], therefore, we tried to extend this range by increasing the minimum support value, Minsup, in order to discover patterns (i.e., increase the upper boundary of the rare items would allow more items to be included in the RP-tree). Accordingly, we changed the minimum support value to be 0.2 instead of 0.1, and then, surprisingly again, it extracted a huge number of patterns (between 4,608 and 7,680). Furthermore, the total execution time for our proposed

[1] Play Connect-4 online on: `https://www.mathsisfun.com/games/connect4.html`

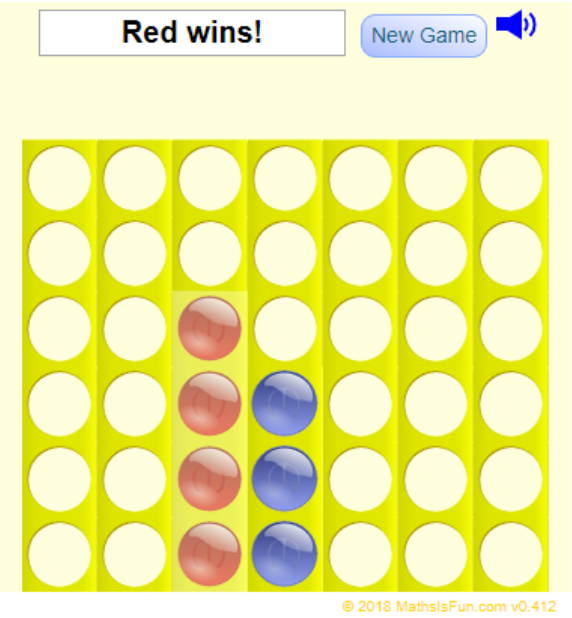


Figure 5.9: An example of a position involving players of the Connect-4 game

algorithm was between $15ms$ and $36ms$; this was lower than the FP-Growth execution time, of between $16ms$ and $64ms$, but exceeded the execution time of the RP-Growth algorithm (using threshold 0.1) of between $8ms$ and $20ms$. However, this latter time was used by RP-Growth without any pattern being extracted; this time expanded using a higher threshold, 0.2, RP-Growth took between $76ms$ and $126ms$ to extract up to 7,680 patterns but most of these patterns were then found to be trivial (e.g., pattern which represents one position).

These findings, in addition to the above discussed results from applying the FP-EStream algorithm to the parking lots data set, further support the idea of using a dynamically calculated threshold to discover pattern from evolving data stream with skewed distributions. The results show the efficiency of the proposed algorithm for overcoming the rare patterns problem; specifically, the low $Threshold$ values must help to extract the rare patterns encountered in the data set, and at the same time, these low values do not cause an explosion in the number of patterns discovered — as such low thresholds do when used with other methods. Reducing the number of patterns extracted results in a decrease in the memory costs.

5.5 Chapter Summary

Discovering patterns from dynamic streams which have skewed distributions is challenging in terms of complexity and execution time. The FP-Growth mining technique is based on a compact frequency-descending FP-tree structure [123], and a number of different variants of this technique have been applied widely on conventional data sets [140, 156, 258]. However, it was difficult to extend these techniques into the evolving streams scenarios which also exhibited skewed distributions. There are two main reasons why these algorithms are restricted to conventional data sets:

- These FP-Growth based techniques require multiple scans of the data set.

With data streams, often only some of the items are available when the pattern mining process commences. Moreover, algorithms which operate on data streams should only make a single or a few passes over the data before the data are discarded — due to the arrival of new data.

- These techniques use predefined single/ or multiple threshold(s) to enable them to discover patterns from data sets. And in fact, choosing a high threshold value causes the rare pattern problem (patterns which appear infrequently in the data stream are not captured) whereas selecting a low threshold value (in order to discover the rare patterns) is considered to be costly and inefficient, as it can produce a huge number of meaningless patterns.

This chapter introduced a novel technique, the FP-EStream technique, for pattern discovery, which operate over evolving data streams with skewed distributions. The contribution of this investigation is two-fold: first, a new pattern discovery method was presented which is capable of discovering both frequent and rare patterns efficiently. Second, the proposed FP-EStream technique has the ability to capture the changing dynamics of patterns emerging from heterogeneous streams efficiently. Specifically, this technique used an adaptive learner technique, naive Bayes, over a sequence of titled time window, to build the frequent item list which could then be used to construct the FP-tree. Classifying the streams reduced the number of patterns which were detected from the heterogeneous streams. Furthermore, a dynamically calculated threshold was used to identify patterns contained within these heterogeneous streams; this threshold was computed for each window based on the minimum joint probability obtained from the naive Bayes classifier. The results, from applying the proposed algorithm to both the data streams generated from the simulation model and to the real data set, show that the technique produced can efficiently extract both frequent and rare patterns; it can also handle the issue of concept drift as encountered in rela-

tion to patterns which emerge from such streams. Furthermore, the results support the merit of using the proposed dynamic support threshold to overcome the rare patterns problem in non-stationary streams exhibiting multi-class imbalance. This technique does not need to generate candidates or scan the database many times, unlike other pattern discovery approaches (e.g., FP-growth). Although the proposed technique was tested primarily on a car parking lots environment in this study, it could be adapted to be used in many differing domains: e.g., it could be applied to the discovery of patterns in a wide variety of IoT streams (tube fault diagnosis, or water leakages monitoring).

Chapter 6

Deep Learning for Non-stationary Multivariate Time Series Forecasting

6.1 Introduction

Forecasting, the process of making predictions about future events based on the past and on current data, plays a major role in decision making across a broad range of domains: e.g., weather, energy, transportation, business, and entertainment [14, 144, 200]. Time series data derived from dynamic environments capture the dynamic behaviours and provide the means by which to monitor and predict changes in these environments [144]. While time series forecasting has been extensively studied by statisticians, the rapid growth in the complexity and size of time series data streams, generated by many real-world applications, pose significant challenges for the existing statistical techniques [40, 41, 144]. Thus, there is an increasing necessity to develop time series modelling and forecasting techniques that deal with large volumes of data generated at high speed (i.e., from dynamic environments) which are governed by multiple variables, complex seasonality and produced over multiple time steps.

This chapter presents an empirical exploration of deep learning techniques used for forecasting from non-stationary multivariate time series. The main purpose of this investigation is to determine if we can develop accurate parking availability forecasts in real time, via Gated Recurrent Units (GRUs) and the multivariate time series collected from the IoT devices in parking lots. The prediction of parking availability mainly depends on the multiple seasonal patterns exhibited in the parking lots and on the real-time occupancy data. Therefore, we seek to build a deep learning model using multiple data streams, generated from a limited number of existing IoT devices, to provide predictions which are as accurate as possible across a variety of parking lots. This is unlike most of the solutions already proposed in the literature, many of which are based on the installation of additional hardware, e.g., cameras or sensors at each parking space, for the achievement of their aims [112, 115, 233, 293]. Such approaches imply extra installation costs and ongoing maintenance expenses.

The rest of this chapter is structured as follows. The next section presents a brief discussion of related works. Section 6.3 will provide the problem scenario and the framework used in this exploration, this will be followed by the experimental work in Section 6.4, where the results will be discussed. Finally, Section 6.5 provides the the chapter summary and conclusions.

6.2 Related Work

In the United Kingdom, drivers spend 91 hours, on average, every year (nearly four days) looking for parking spaces, according to a study [15] produced by the British Parking Association (BPA) in 2016. Furthermore, a driver can often take about eight minutes to find a parking space in London, whereas five minutes of searching is more likely in the East of England and the East Midlands [15].

Predicting parking availability has received significant attention in the literature [112, 115, 209, 228, 233, 265, 293]. However, most studies have focused only

on predicting parking availability in the situation where additional hardware and sensors for monitoring parking lot occupancy have been deployed in the parking lots in question for precisely this purpose. For example, automated clustering and anomaly detection techniques were applied to parking datasets collected from 8,200 sensors in San Francisco — in order to identify the trends and events over a two week period [293]. In another example, a number of different Multi-Layer Perceptron (MLP) neural networks were applied to a number of different parking lots, using the data collected from 253 sensors installed across Santander city in Spain for the purpose of predicting parking occupancy [265]. More recent work [228] has applied Long-Short Term Memory (LSTM) networks [135] to situations where there were fewer parking sensors and a variety of data sets — involving meteorological data, events, map mobility trace data and navigation data — to predict parking availability.

6.3 Deep Learning for Time Series Forecasting

The goal here is to design a generic time series forecasting model that is accurate and applicable to non-stationary multivariate time series. To achieve this, we examined the recently introduced Gated Recurrent Units (GRUs) [58] in relation to the task of forecasting time series. This section presents an abstract framework for the forecasting of non-stationary multivariate time series before discussing the proposed forecasting model.

6.3.1 Problem Scenario

The proposed forecasting model seeks to use the various seasonal patterns in the usage of each parking lot and its nearby parking lots, in addition to the real-time occupancy data, to forecast the availability of free parking spaces at each parking lot, across the following time intervals.

Let O be the set of n multivariate time series that represent the occupancy of the various parking lots P:

$$O = \{O_1 \dots O_n\}$$

As mentioned in Chapter 2, a multivariate time series has more than one time-dependent variable, where each variable depends not only on its past values but also has some dependency on other variables [42, 144, 240]. The occupancy, O_j, of a parking lot P_j, where j ranges from 1 to n, at time t can be represented as:

$$O_j = \frac{number\ of\ the\ parked\ cars\ in\ P_j}{total\ number\ of\ parking\ spaces\ in\ P_j} \tag{6.1}$$

Where P is the set consisting of n parking lots, $\{P_1 \dots P_n\}$, monitored by k IoT devices (i.e., smart pay machines, camera sensors, and/or mobile applications), and where the IoT devices generate k data streams as described earlier in Chapter 3. Typically, the parking availability changes continuously over time, and the past parking patterns of each parking lot and its nearby locations may affect the current parking availability; this can be quite effectively handled by the Recurrent Neural Network (RNN) model which has been proven to be a powerful technique for dealing with sequential data [120, 133, 248].

Typically, one would expect that the occupancy of parking lots is likely to be time dependent. The date-time attributes in the time series may imply a great deal of information, and it can be difficult for a forecasting model to exploit all of this information if the date-time attribute is in the standard format (e.g., DD-MM-YY, HH:MM:SS). Thus, the date-time data-item must be decomposed into its constituent parts, and this may allow the forecasting models to more easily discover the various seasonal patterns in the dataset. The process of extracting more information from the existing data is known in the literature as Feature Engineering (FE) [84, 151]. As the efficacy of a machine learning algorithm depends on the input features [84], FE can have a remarkable impact on prediction perfor-

mance, for example, authors in [19] analysed an individual card holders behaviour for card fraud detection problem, and found that feature engineering increases the performance of the learning algorithm.

Let TS be the set of m multivariate time series that contain the time data associated with the above set of parking lot occupancy time series, O. The choice of the number, m, depends on the number of regular seasonal patterns in the time series data set. That is, the time series data set might have any combination of yearly, quarterly, monthly, weekly, daily or sub-daily seasonal patterns. If a time series is relatively short, it can be expected that fewer seasonal components are present.

In this scenario, let X be defined as the multivariate time series inputs, where $X = TS \cup O$, the number of input time series equals $m + n$, and the recent sequence over the interval a is represented as: $X_a = (x_{t-a-1}, x_{t-a-2}, \ldots, x_t)$. Whereas Y is defined as the target multivariate time series (i.e., the outputs), $Y = O$, and the number of outputs is n time series. Furthermore, as the forecasting model aims to predict different following intervals, the number of the time-steps to forecast is defined as NS. The forecasting model can be represented as follows:

$$Y_t = f(X_{t-1}, \ldots, X_{t-NS}) \tag{6.2}$$

In words, we consider the inputs as multiple time series that composed of multiple variables/measurements and which represent the occupancy of multiple parking lots at multiple temporal granularities, together with additional variables that encode temporal instants (i.e., stating the hour, day of the week, month, etc.) and we consider the outputs as multiple time series which represent the occupancy of multiple parking lots for following intervals (i.e., a given number of the time-steps).

6.3.2 Time Series Forecasting Model

In the proposed method, the GRU has been chosen as the deep learning model for forecasting the non-stationary multivariate time series, derived from the heterogeneous data streams. As mentioned in Section 2.7.1, GRUs are special variants of LSTMs [135] and they consist of two gates: reset gate and update gate that control the flow of information through the model and refine outputs. These models can retain information over a period of time have both been shown to perform well in tasks that require capturing long-term dependencies [61, 149, 248]. Furthermore, it is possible to integrate any type of feature into the GRUs model (e.g., numerical or categorical, time-dependent or series-dependent) [58, 61]. Despite the fact that deep LSTMs have been extensively studied in the literature, and have been shown to have better performance in terms of modelling time series than statistical models [173, 228], to our knowledge, GRUs have not yet been explored in relation to the task of forecasting time series.

Accordingly, the heterogeneous data streams from the IoT devices are used, first, to generate the required set of multiple time series, O, which represent the occupancy of the parking lots. Then in order to capture the different seasonal patterns in the time series, the TS time series set is extracted from the IoT data streams as well. In particular, the time of day, the day of the week and the month are included as input features to the GRUs model. By including these features explicitly, we attempt to make the forecasting model sufficiently flexible that it can capture the various different seasonality correlations in the multivariate time series. The input, X_a, and the target time series, Y_a, are selected as described in the previous section; then the target data are shifted by the number of time-steps required to make the forecast NS. Furthermore, x_i is fed into the GRUs model, which then outputs a hidden state, h_i, and the sequence of hidden states over the interval, a, are concatenated as $\widetilde{h}_a$; these states are then fed through a fully connected layer h_a, the outputs of which are then fed through another fully-

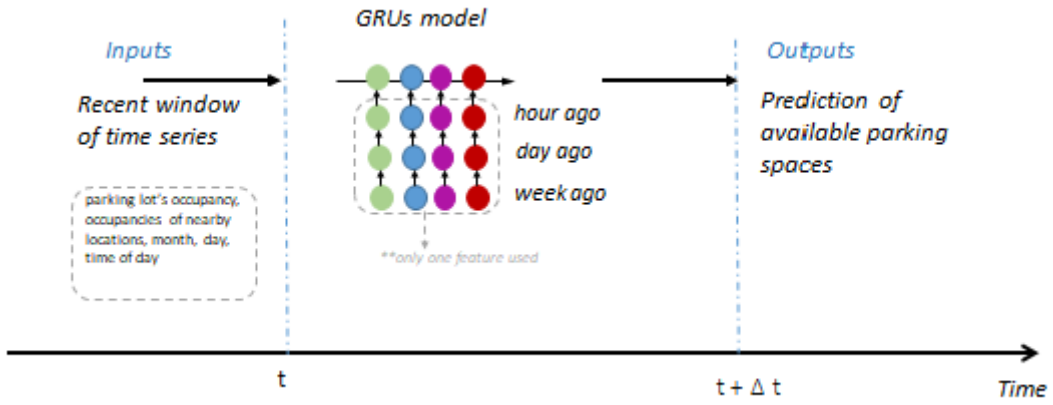


Figure 6.1: An abstract framework for the proposed forecasting model

connected layer and the output of this is then the parking availability, y_i, at time interval t, according to the Equations 2.11, 2.12 , 2.13, and 2.14. The diagram in Figure 6.1 shows an abstract of the forecasting model.

Obviously, time series are time-dependent and past observations are used to predict future observations. Therefore, it was not possible to split the dataset into training and testing sets using the standard cross-validation method; because the standard cross-validation method splits the data randomly and does not preserve the time ordering. Therefore, we trained the data using a walk-forward cross-validation. The walk-forward method is a well-known technique used in time series analysis where the first part of the series is utilised as training data and the last part is utilised as testing data [144, 157]. The training set will be updated only with the arrival of a new batch (sliding window) of data. The diagram in Figure 6.2 illustrates how walk-forward cross-validation works.

6.4 Experimental Work

The experiments discussed in this chapter were conducted on data derived from the parking lots scenario at the University of Essex. In paticular, the simulated data streams described in Chapter 3 were used to construct the multivariate time

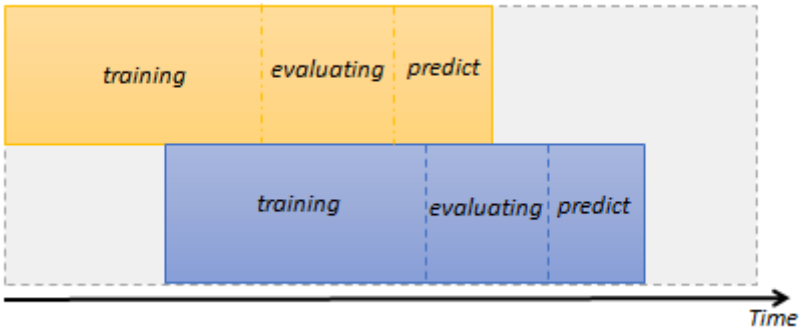


Figure 6.2: Walk-forward cross-validation for time series

series that represent the occupancy of the parking lots. Furthermore, in order to get a detailed analysis of the architectures and methods proposed in this chapter, we run the proposed models on different datasets. Particulary, 10 different iterations of the simulator generated in Chapter 3 were used to generate 10 different datasets for the same Autumn term 2015-16. First, this section will show how the occupancy time series were constructed from the simulated IoT streams. This will be followed by a detailed description of the multivariate time series dataset that represents the occupancy of the parking lots at the University of Essex. Then, the GRUs model will be implemented. Specifically, Keras [59] with a TensorFlow backend [252] has been used to build the neural network models in Python [121]. Keras is a high-level neural networks Application Programming Interface (API), written in Python and capable of running on top of TensorFlow, or Theano [201], and TensorFlow is an open source library for developing and training machine learning models. Finally, forecasting results will be compared with those resulting from the use of LSTMs which used for this type of problems in literature [173, 228], shallow MLP models and from the use of the benchmark statistical methods such as TBATS [76].

6.4.1 The Occupancy of the Parking Lots Time Series

As has been described in Chapter 3 in relation to the experimental framework, three data streams were simulated for the Autumn term 2015-16: the combination of the payment streams from both the pay stations, the stream from the mobile application, and the sensors stream from the three camera sensors. So in order to estimate an approximation of the occupancy in the parking lots, a counter is associated with each parking lot; these counters are increased in line with the arrival of new items in the payment streams (i.e., a counter is increased when a driver pays a parking fee). And conversely, the counters are decreased in line with the arrival of new items in the sensors stream that indicate that cars, linked with payment information, have left the parking lot. It is important to note here that counters are increased based on the payment stream, not on the sensors stream. Although there is generally a short gap in time between finding a parking space and paying parking fee, the payment streams give a better indicator of the real situation than the sensor streams. Subsequently, the three streams were joined using MS-SQL — ordered by time of payment and time of leaving the parking lot. The parking lots are monitored only for part of the day, from 07:00 in the morning until 16:00 in the afternoon, and the counters were initialized (zeroed) with the arrival of each new day-window (starting at 7.00 am). These processes resulted in a time series of 184,323 observations in total [1], for the 11 weeks of the Autumn term, from 5 October 2015 until 18 December 2015. For simplicity, the size of this time series has been reduced to 5,940 observations (for each dataset) by keeping only the information regarding the occupancy of the parking lots which is available at the end of each 5 minute period, from 07:00 in the morning to 16:00 in the afternoon of the week days of the Autumn term 2015-2016 (108 observations per day, 540 observations per week). As mentioned in Chapter 3, there are six parking lots: parking lot A for students; Valley parking lot for staff members,

[1]These observations collected in the first dataset, observations collected from the other 9 datasets range between 184,265 and 184,473

students, and visitors; the Constable Building parking lot for staff members and the hotel visitors; parking lot B for students, staff members, and Gym members; the Multi-deck parking lot for staff members; and the North parking lot for staff members, students, and the university's nursery. It is worth exploring what is known about the (simulated) occupancy of these parking lots before applying the forecasting model. The diagram in Figure 6.3 shows the occupancy of the parking lots over the Autumn term 2015. In more detail, the diagrams in Figure 6.4, Figure 6.5, Figure 6.6, Figure 6.7, Figure 6.8, and Figure 6.9 provide an overview of the occupancy patterns for a period of a single day (the first day of the Autumn term 2015-16), and the occupancy over the whole term (11 weeks) for each parking lot separately [2]. Clearly, there are direct correlations between the occupancies of the various different parking lots; therefore, the occupancies of a parking lot and its nearby parking lots are extracted to represent the occupancy features.

[2]these diagrams represent the occupancy patterns for the first dataset, diagrams of the occupancy patterns for the other 9 datasets are provided in Appendix D

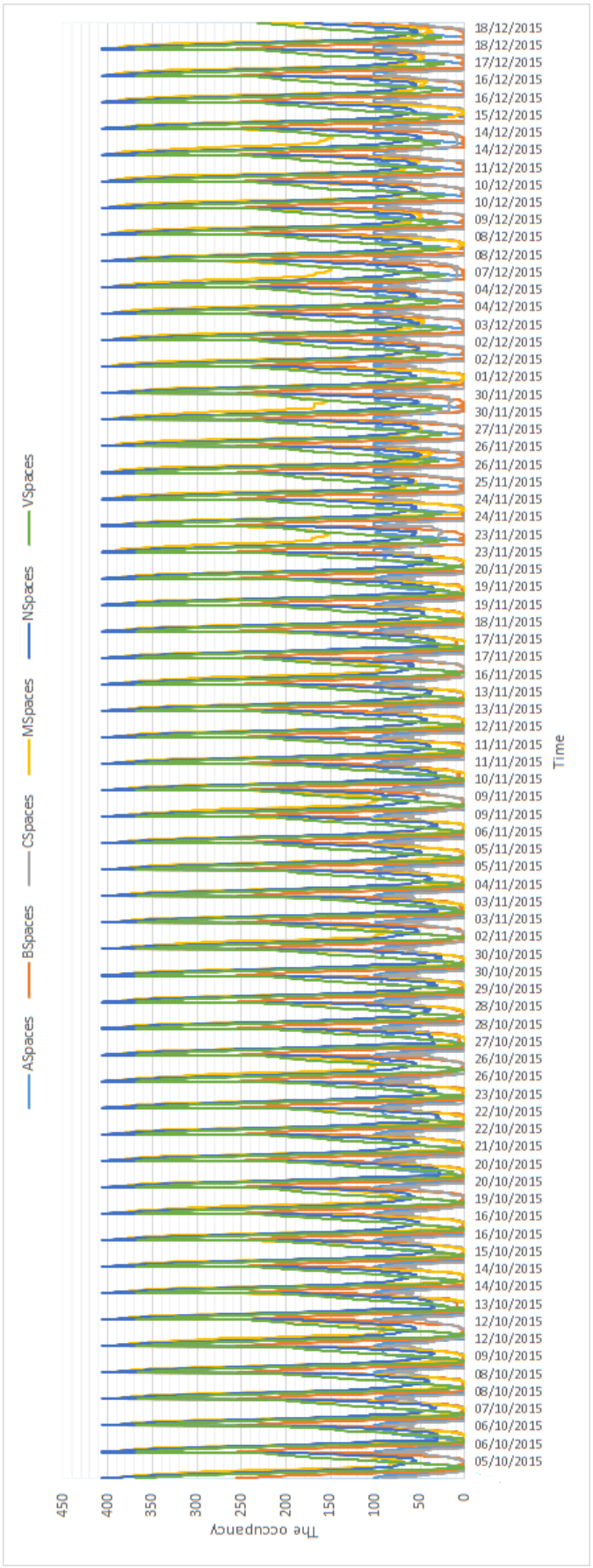
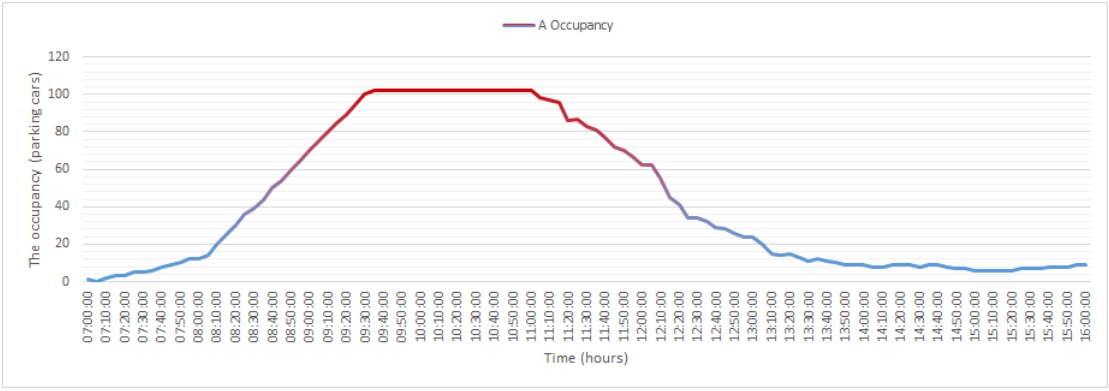
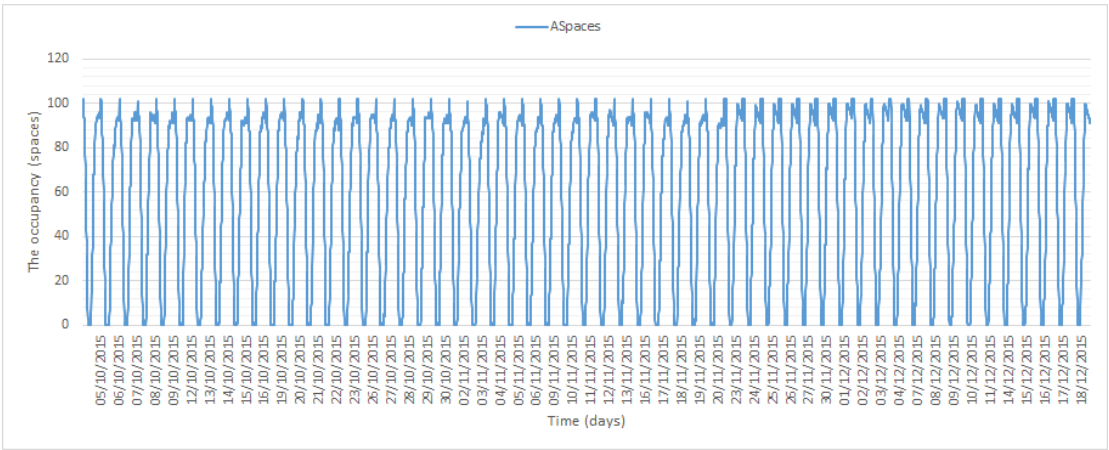


Figure 6.3: The occupancy of the parking lots over the Autumn term 2015

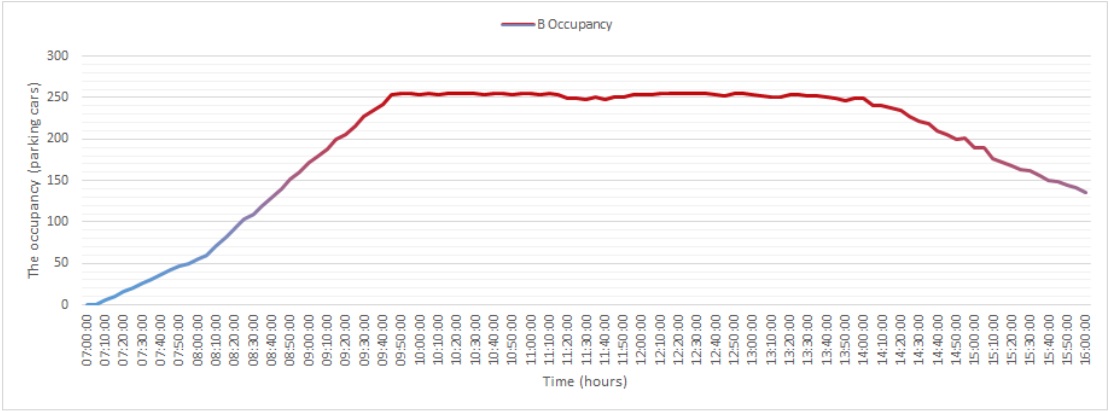


(a) Over the first day of the Autumn term 2015-16 (one day)

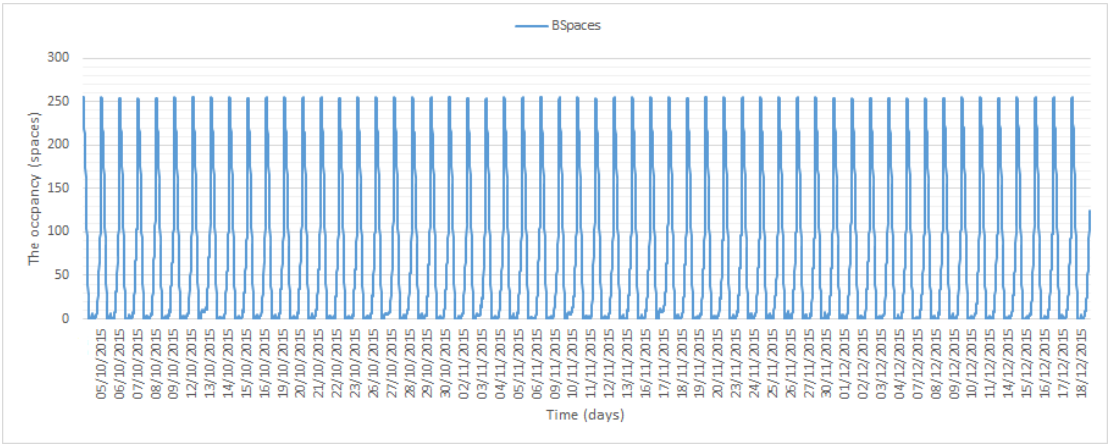


(b) Over the Autumn term 2015-2016 (11 weeks)

Figure 6.4: The occupancy of parking lot A

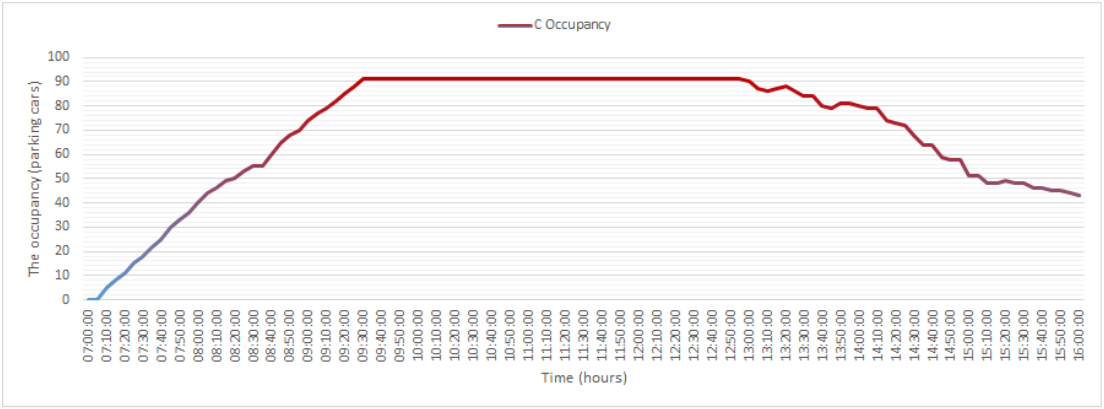


(a) Over the first day of the Autumn term 2015-16 (one day)

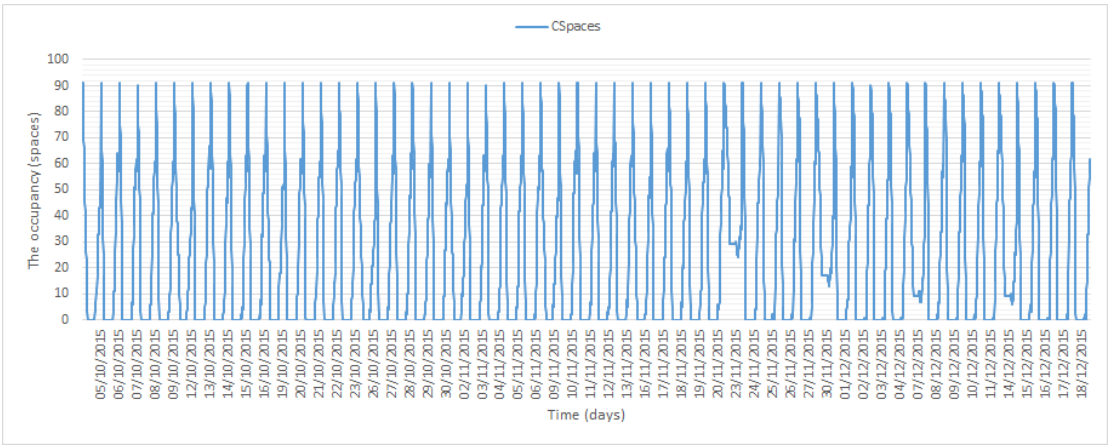


(b) Over the Autumn term 2015-2016 (11 weeks)

Figure 6.5: The occupancy of the parking lot B

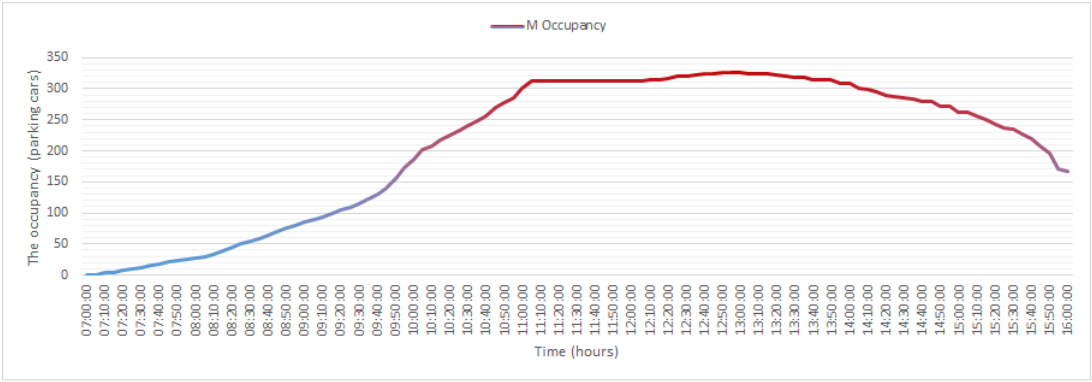


(a) Over the first day of the Autumn term 2015-16 (one day)

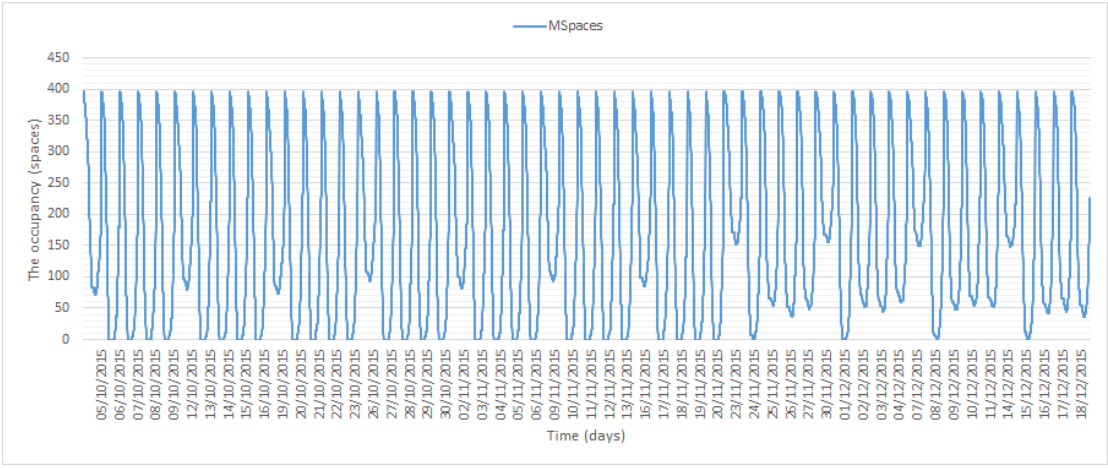


(b) Over the Autumn term 2015-2016 (11 weeks)

Figure 6.6: The occupancy of the Constable building parking lot

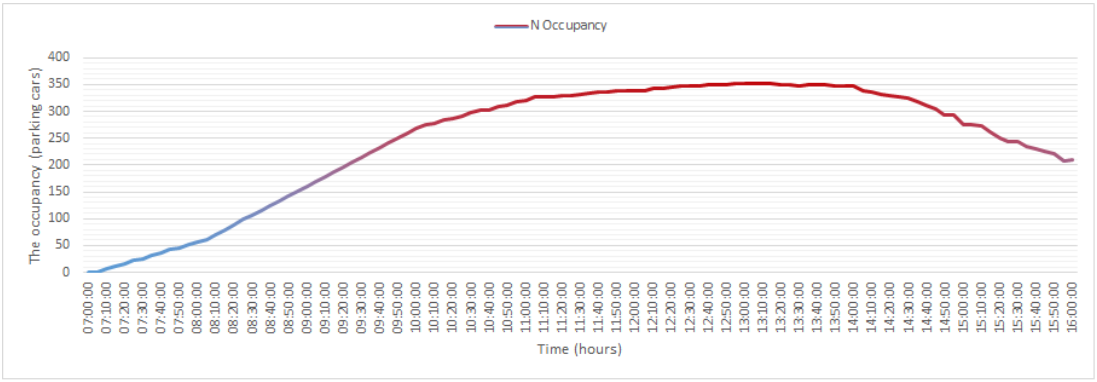


(a) Over the first day of the Autumn term 2015-16 (one day)

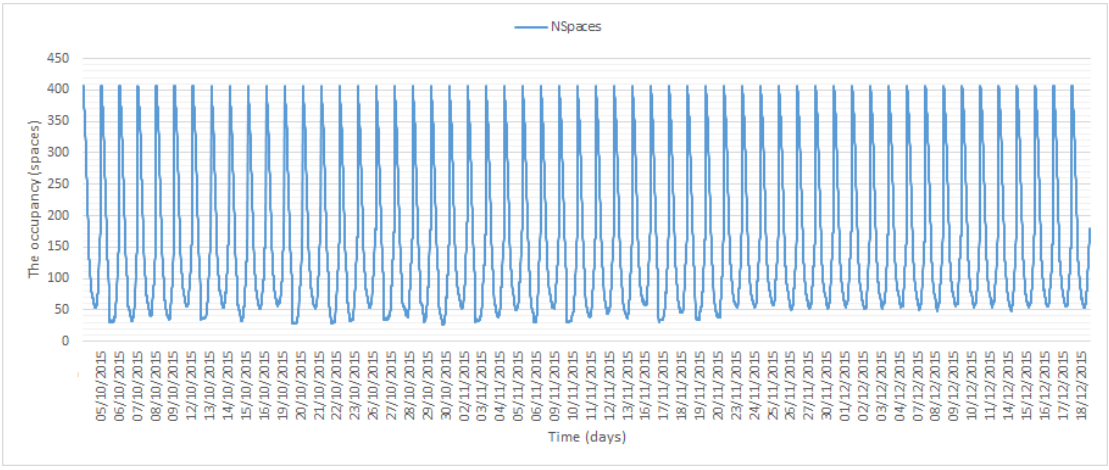


(b) Over the Autumn term 2015-2016 (11 weeks)

Figure 6.7: The occupancy of the Multi decked parking lot

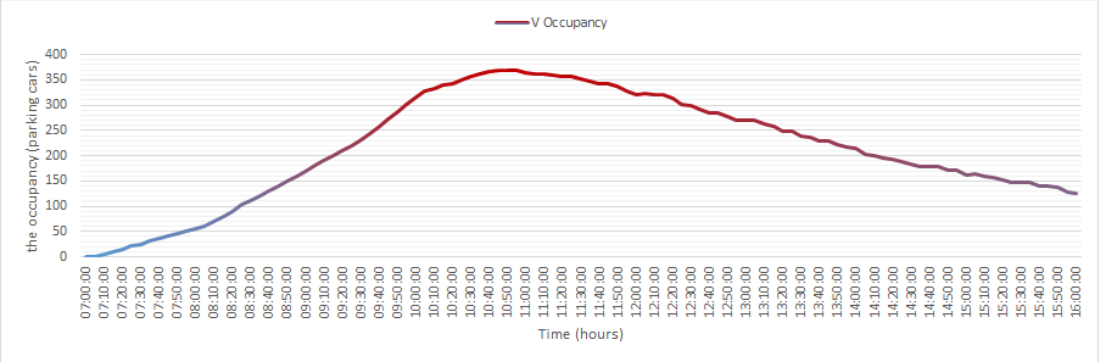


(a) Over the first day of the Autumn term 2015-16 (one day)

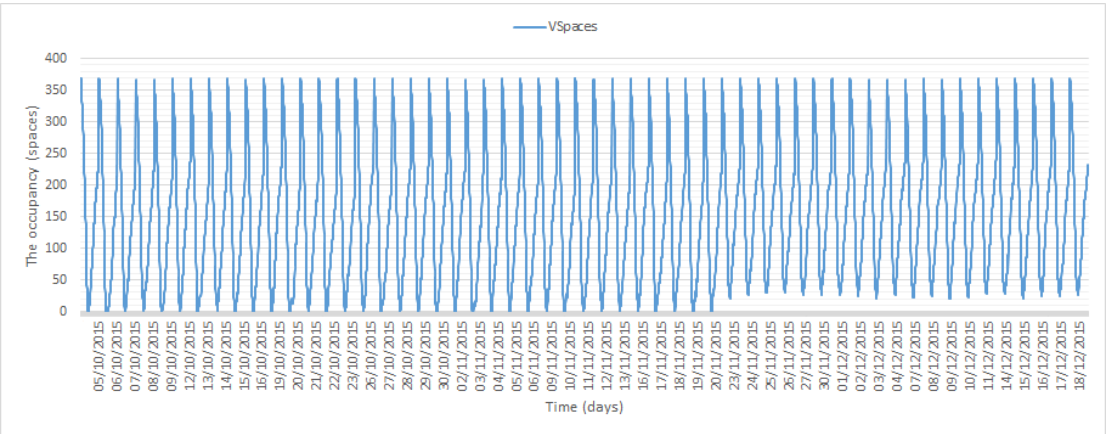


(b) Over the Autumn term 2015-2016 (11 weeks)

Figure 6.8: The occupancy of the North parking lot



(a) Over the first day of the Autumn term 2015-16 (one day)



(b) Over the Autumn term 2015-2016 (11 weeks)

Figure 6.9: The occupancy of the Valley parking lot

6.4.2 Time Features Extraction

Obviously, the occupancy of the University parking lots is affected by the time of day, the day of the week, etc. Although the data relating to off-peak periods and to the holidays have already been removed from the datasets, the diagrams of the parking lots occupancy, presented in the previous section, show multiple seasonal patterns. In particular, there is clear daily and weekly seasonality in the datasets. Moreover, there is a term seasonality, but since the time series datasets are only one term long, this appears just as a decreasing trend. Although each dataset contains date and time information for each observation, this is only used as an index for ordering the data, therefore, a number of time related data items (attributes) have been added to the occupancy time series in order to help the proposed model in making predictions. Namely, five time items have been added: month, week of the academic year, day of the week, and time of day (hour and minute); then different combinations of these inputs have been tested and compared.

6.4.3 Building the Prediction Model

In this section, it is shown how a number of different Gated Recurrent Units (GRUs) models were applied to the datasets using the month of the year, the day of the week, the time of day, and the occupancy of the parking lots as inputs; the target outputs were the predicted occupancy of the six parking lots. Table 6.1 shows the different inputs/outputs variables that constitute the input multivariate time series and the output multivariate time series. Specifically, five different models were tested, as follows:

1. Forecast the parking availability in the next hour
2. Forecast the parking availability in the next two hours
3. Forecast the parking availability in the next three hours

Table 6.1: The detail of variables in the input multivariate time series and the output multivariate time series

Inputs	Outputs
Month	
Day	Occupancy of Park A
Time	Occupancy of Park B
Occupancy of Park A	Occupancy of the Constable Park
Occupancy of Park B	Occupancy of Valley Park
Occupancy of the Constable Park	Occupancy of the Multi-deck Park
Occupancy of Valley Park	Occupancy of the North Park
Occupancy of the Multi-deck Park	
Occupancy of the North Park	

4. Forecast the parking availability for the next day

5. Forecast the parking availability for the next week

For each model, we shifted the target-data according to the number of time-steps which were to be forecast. As has been said, the data set has observations for every five minute period (i.e., one time-step corresponds to five minutes), so 12 time-steps corresponds to one hour, and 12 x 9 x 5 time-steps corresponds to one week. To predict the occupancy in the next hour, the data must be shifted 12 time steps, similarly if want to predict the occupancy for the next day, the data must be shifted 9 * 12 time-steps, the number of shift-steps for each dataset can be defined as:

$$\text{shift steps (hours)} = \text{hours to predict} * 12$$

$$\text{shift steps (days)} = \text{days to predict} * 9 * 12$$

$$\text{shift steps (weeks)} = \text{weeks to predict} * 5 * 9 * 12$$

Then each dataset was splitted into three sub sets: 70% of the dataset for training, 15% for validation to monitor the model's performance, and 15% for testing. As the input features are on different scale, we need to normalise the features, accordingly, the data was then scaled using scikit-learn (Min Max scalar)

[66] before the GRU model was applied; this was done because the neural networks used work best on values between -1 and 1, approximately. There are 5,940 observations in each dataset, and the model took 10 inputs and produced 6 outputs for each observation, but training the GRU models over the complete sequence of observations resulted in poor performance, so instead of using the whole training set, we created batches of shorter sub-sequences; these sequences were picked randomly from the training-data. Two different batches constructed in this way have been tested: the first batch consisted of 108 sequences each with a sequence-length of 540, which meant that each random sequence contained observations for a five day period; and the second batch consisted of 540 sequences each with a sequence length of 108, which meant that each random sequence contained observations for the previous day. The GRUs models were implemented using three layers (two hidden layers and one output layer [3]); the first layer mapped the inputs from the dataset onto 64 or 128 outputs. The choice of using either 64 or 128 such outputs was based on trying a variety of different values to see which ones fitted with the data set. Then a fully-connected (dense) layer was added (also 128 or 64 units wide), then another fully-connected layer was added which mapped (128 or 64) values down to only 6, using the hard-sigmoid function — this is faster to compute than the standard sigmoid function [207]. Either of these functions can be used to map the data to outputs between 0 and 1; the standard sigmoid function can be calculated as follows [207]:

$$f(x) = \frac{1}{1 + \exp^{-x}} \tag{6.3}$$

whereas the hard-sigmoid can be written as [207]:

$$f(x) = max\left(0, min\left(1, \frac{(x+1)}{2}\right)\right) \tag{6.4}$$

[3] note that the input layer is not counted in the number of layers when calculating the depth of network [93, 291]

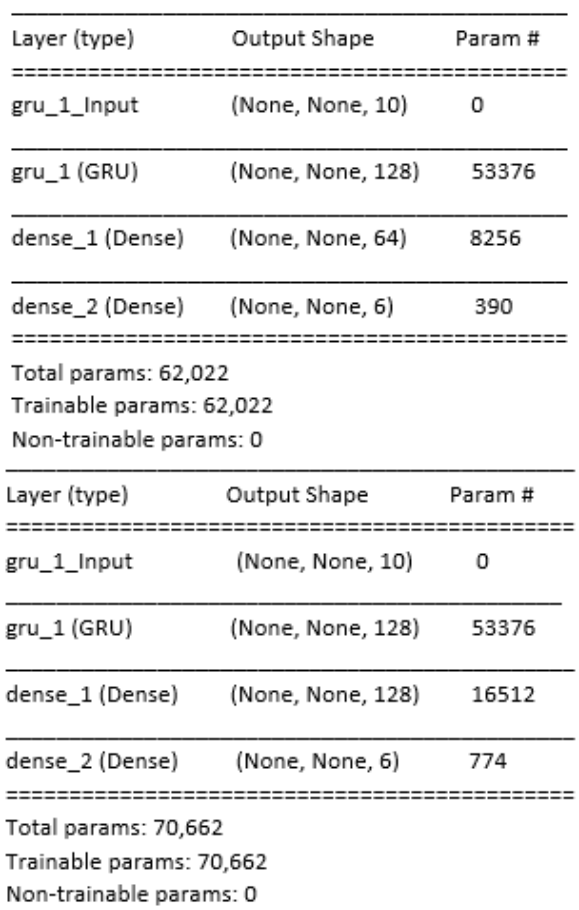


Figure 6.10: Details of layers in the proposed GRUs models

Details of the layers are shown in Figure 6.10. Moreover, the Mean Squared Error (MSE) which has been described previously in Equation 2.7, was used to calculate the loss function. We used a warmup period of 12 time-steps at the beginning of the training, the results from which were discarded for the purposes of calculating the loss function; this was in order to improve the accuracy of the calculation in relation to the results as a whole. Moreover, we used RMSprop [1], with an initial learning rate of $1e-3$, as the model optimizer. Furthermore, for each generated batch, we used 20 epochs with 10 steps for each epoch. Table 6.2 illustrates the specifications of each model then the forecasting performance from these GRUs models for the University parking lots scenario (i.e., on the 10 simulated datasets) are summarized in Figure 6.11.

Table 6.2: The specifications of the GRUs models used

Model	GRUs Model (neurons)	No. steps to forecast	Sequences in the batch
1	10,128,128,6	1 hour	108
2	10,128,128,6	1 hour	540
3	10,128,64,6	1 hour	108
4	10,128,64,6	1 hour	540
5	10,128,128,6	2 hours	108
6	10,128,128,6	2 hours	540
7	10,128,64,6	2 hours	108
8	10,128,64,6	2 hours	540
9	10,128,128,6	3 hours	108
10	10,128,128,6	3 hours	540
11	10,128,64,6	3 hours	108
12	10,128,64,6	3 hours	540
13	10,128,128,6	1 day	108
14	10,128,64,6	1 day	108
15	10,128,128,6	1 week	108
16	10,128,64,6	1 week	108

Overall, the results show that the GRUs models achieved a good performance with low values of error measures: MSE and MAE. It is apparent from the results that using a bigger batch size (540 instead of 108) slightly increased the performance of the GRUs models, however, this may mean that the model was over-fitted, it is easy that such a high-information capacity model became over-fitting on short time series. The Mean Absolute Error (MAE) values of these models varied between 0.043829 and 0.090664, whereas the Mean Squared Error (MSE) values varied between 0.0041 and 0.01655. Moreover, for each dataset in Figure 6.11, the reported performance values are the mean of the performance values over all outputs (i.e., the reported value is the sum of the measure values divided by the number of outputs). For example, the reported MSE value for each observation in Figure 6.11 is the mean of six MSE values (over the six outputs). Moreover, the details of MSEs of the proposed models on all datasets are presented in Appendix E. Furthermore, diagrams in Figure 6.12 show the loss function demonstrated by these models.

Model	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7	Dataset8	Dataset9	Dataset10	Mean Loss
1	0.00848	0.00825	0.00904	0.00891	0.01322	0.00922	0.01143	0.0111	0.01023	0.0099682	0.009985826
2	0.00804	0.00861	0.00784	0.00765	0.00788	0.00882	0.01398	0.00888	0.00963	0.0089998	0.009033006
3	0.00846	0.00935	0.01119	0.0091	0.01079	0.00974	0.01012	0.00959	0.012	0.0130425	0.010337212
4	0.00768	0.00854	0.01105	0.00904	0.01087	0.00912	0.00869	0.00905	0.00917	0.0109429	0.009414954
5	0.00896	0.01281	0.01119	0.01126	0.01094	0.01189	0.01199	0.01223	0.01128	0.0154351	0.011799443
6	0.00865	0.00854	0.00743	0.01174	0.01039	0.00989	0.01004	0.00888	0.00999	0.0097588	0.009529006
7	0.00853	0.01093	0.01216	0.00929	0.01404	0.01243	0.01199	0.01315	0.01124	0.011008	0.011475895
8	0.00986	0.00793	0.00908	0.00929	0.0124	0.01016	0.01131	0.00917	0.01338	0.014515	0.010709552
9	0.00875	0.00849	0.00943	0.01057	0.01019	0.01127	0.01078	0.01141	0.01213	0.0114752	0.010449079
10	0.00859	0.01371	0.00856	0.00814	0.0095	0.00926	0.01548	0.01594	0.01048	0.0094678	0.010912584
11	0.01039	0.00921	0.0104	0.01035	0.01199	0.00976	0.01015	0.01208	0.01198	0.01061	0.010691519
12	0.00826	0.00961	0.00891	0.00946	0.00937	0.00924	0.00957	0.01137	0.00995	0.0097392	0.009548203
13	0.00691	0.00689	0.00563	0.00566	0.00638	0.00687	0.00669	0.01022	0.00633	0.0068828	0.006847277
14	0.00691	0.00636	0.0068	0.00679	0.00658	0.00742	0.00708	0.0078	0.00695	0.0089496	0.007165666
15	0.00406	0.00518	0.00499	0.00535	0.00599	0.00519	0.0054	0.00557	0.0057	0.0056744	0.005309576
16	0.00504	0.00478	0.00595	0.00571	0.00693	0.00636	0.00705	0.00635	0.00622	0.0057698	0.00601519

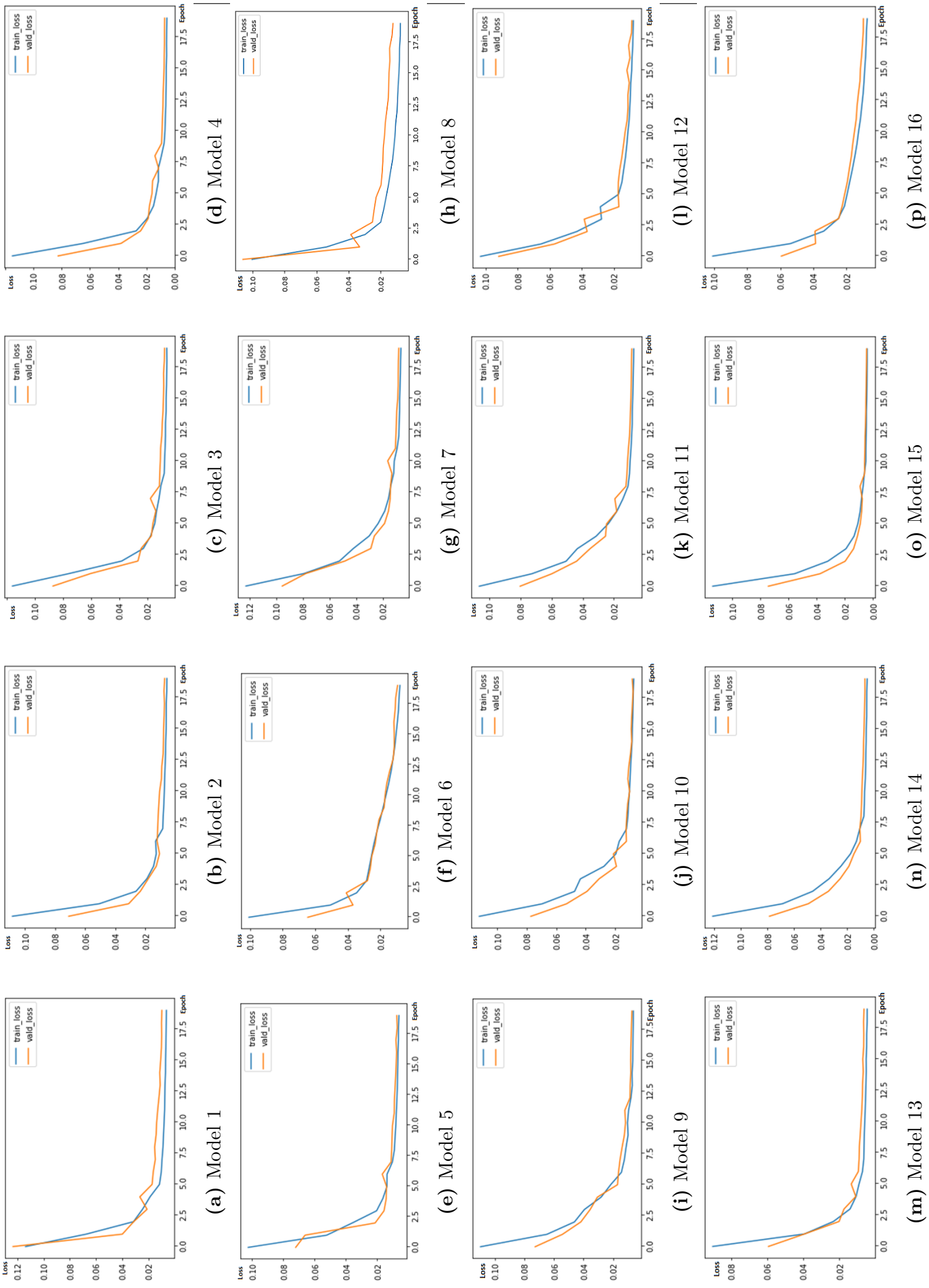
Figure 6.11: Forecasting performance evaluation of the GRUs models on the test sets (loss)

Model	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7	Dataset8	Dataset9	Dataset10	Mean MSE
1	0.010082	0.010311	0.009247	0.009137	0.013789	0.009336	0.011639	0.011755	0.010767	0.0102542	0.010631587
2	0.008652	0.009341	0.00794	0.008261	0.008331	0.009235	0.014266	0.009099	0.009722	0.0095377	0.009438478
3	0.009807	0.011436	0.011481	0.009525	0.011011	0.010067	0.010499	0.010125	0.012071	0.0134793	0.010950147
4	0.008235	0.00882	0.011126	0.009428	0.011314	0.009507	0.009094	0.009343	0.009519	0.0114167	0.009780181
5	0.011285	0.015827	0.01169	0.011952	0.011047	0.012151	0.012755	0.013493	0.011993	0.0162975	0.012849102
6	0.009505	0.013665	0.007482	0.012523	0.010446	0.010085	0.010363	0.009099	0.010299	0.0098069	0.010327378
7	0.010839	0.012032	0.012585	0.009688	0.014468	0.012758	0.012755	0.013576	0.011608	0.0116796	0.012198898
8	0.010798	0.009536	0.009368	0.009688	0.012856	0.010327	0.011768	0.009466	0.013734	0.0151474	0.01126872
9	0.010831	0.010943	0.010079	0.011993	0.010767	0.01186	0.011132	0.012055	0.013143	0.012473	0.011527563
10	0.00946	0.015754	0.008828	0.008282	0.009797	0.009365	0.016211	0.01655	0.010757	0.0097751	0.011477872
11	0.010933	0.009742	0.011085	0.010744	0.012619	0.010179	0.010666	0.012785	0.012528	0.0111627	0.011244445
12	0.008394	0.009826	0.009308	0.009938	0.009692	0.009529	0.009686	0.011773	0.010283	0.0102027	0.009863006
13	0.007757	0.00742	0.00614	0.006138	0.006786	0.007227	0.006885	0.011443	0.006981	0.0075476	0.007432247
14	0.007757	0.006782	0.007114	0.007587	0.006741	0.007744	0.007566	0.00825	0.007255	0.0092775	0.007607394
15	0.0041	0.005604	0.005013	0.005586	0.006198	0.005246	0.005449	0.005925	0.005867	0.0057616	0.00547494
16	0.005091	0.005153	0.006298	0.005932	0.00738	0.006472	0.00763	0.006585	0.006546	0.0060242	0.006311281

Figure 6.11: Forecasting performance evaluation of the GRUs models on the test sets (MSE)(cont)

Model	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7	Dataset8	Dataset9	Dataset10	Mean MAE
1	0.064451	0.065583	0.065026	0.064644	0.086029	0.065686	0.077587	0.069764	0.067894	0.0682953	0.069495992
2	0.063808	0.065774	0.060391	0.060459	0.059949	0.064289	0.083481	0.064899	0.066834	0.0627061	0.065259054
3	0.065454	0.070107	0.073734	0.065406	0.070243	0.068305	0.069757	0.065736	0.077131	0.0778421	0.070371492
4	0.061682	0.063432	0.072435	0.064397	0.073764	0.068461	0.066828	0.06563	0.064467	0.0736205	0.067471766
5	0.06899	0.083826	0.07365	0.070306	0.067775	0.076471	0.076872	0.07549	0.071355	0.0840732	0.074880679
6	0.063819	0.075078	0.057517	0.075593	0.070014	0.066612	0.067576	0.064899	0.066463	0.0663118	0.067388386
7	0.066983	0.075644	0.075875	0.066804	0.081949	0.081292	0.076872	0.076701	0.070674	0.0691911	0.074198411
8	0.072529	0.067583	0.06676	0.066804	0.076476	0.06918	0.077696	0.066136	0.079872	0.0859771	0.072901326
9	0.068472	0.06836	0.064957	0.070989	0.066926	0.073734	0.070319	0.070933	0.073006	0.0700665	0.069776424
10	0.064138	0.084664	0.061784	0.060635	0.064979	0.065601	0.087155	0.090664	0.066762	0.0622222	0.070860197
11	0.067377	0.06464	0.069563	0.068049	0.075229	0.06694	0.07007	0.077294	0.073174	0.0684206	0.070075588
12	0.060166	0.065954	0.064857	0.066325	0.065131	0.063851	0.066831	0.073313	0.066773	0.0641712	0.065737065
13	0.059242	0.056436	0.0523	0.051402	0.05423	0.053465	0.056673	0.072563	0.052722	0.0549883	0.056401939
14	0.059242	0.055614	0.056069	0.058379	0.055458	0.059296	0.057859	0.06039	0.05539	0.0653512	0.058304796
15	0.043829	0.05112	0.049304	0.050344	0.053846	0.049197	0.051532	0.050975	0.050939	0.051534	0.050262065
16	0.050586	0.049481	0.05424	0.054579	0.059499	0.056908	0.058554	0.055484	0.054016	0.0523473	0.054569412

Figure 6.11: Forecasting performance evaluation of the GRUs models on the test sets (MAE)(cont)



(a) Model 1 (b) Model 2 (c) Model 3 (d) Model 4

(e) Model 5 (f) Model 6 (g) Model 7 (h) Model 8

(i) Model 9 (j) Model 10 (k) Model 11 (l) Model 12

(m) Model 13 (n) Model 14 (o) Model 15 (p) Model 16

Figure 6.12: The loss function of the GRUs models on the University parking lots

As explained earlier in the experimental work section, the reason that the proposed models were run on multiple datasets: instead of comparing the performance of the proposed models based on single value (e.g., single MSE value), we compared the performance of these models based on multiple values from multiple datasets. It is clear from Figure 6.11, the mean MSE values over the multiple datasets range from 0.00547 and 0.01285, where the mean MAE values range from 0.05026 and 0.07488. Therefore, we decided to compare the performance results statically to see if there is a significant difference in the performance of the proposed models. Accordingly, we chose repeated measures one-way Analysis of Variance (ANOVA) test [114], which commonly referred to as ANOVA for correlated samples, to compare the proposed models. One-way ANOVA [230] is a statistical technique to test for differences among two or more groups. It is an extension of the t-test which is used to determine if there is a significant difference between the means of two groups. Simply, the one-way ANOVA calculates $F - statistic$; the ratio of the variance calculated among the means to the variance within the groups, in order to infer whether there are significant differences between the means of these groups. Repeated measures ANOVA is the extension of the dependent t-test, it compares means across one or more variables that are based on repeated observations [24, 114].

Before moving to the test, it will be necessary to explain what does the significance test mean. Simply, the null hypothesis H_0 states that the means are equal:

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_g \tag{6.5}$$

where μ is the population mean and g is the number of related groups. The alternative hypothesis H_a states that at least two means are significantly different [114, 230]. Moreover, the $p - value$ is the probability of obtaining the observed

results of a test, assuming that the null hypothesis is true. So if:

$$p - value \leq \alpha \tag{6.6}$$

where the significant level of 0.05 is the most commonly used value for α in literature [24, 278], then we reject the null hypothesis, H_0, and accept the alternative hypothesis, H_a [278].

Accordingly, we used repeated measures one-way ANOVA test to compare the 16 GRU models in R [253]. Figure 6.13 shows the result of the test, where F-value indicates the obtained $F - statistic$, Pr specifies the p-value. Moreover, $critical - F - value$ is calculated using the probability level $\alpha = 0.05$, the numerator degrees of freedom 1, $df_1 = 15$, and the denominator degrees of freedom 2, $df_2 = 135$. It appears from the result that $F - statistic$ is greater than $critical - F - value$: $25.4 > 1.74106594$. This indicates that there is a significant difference between models, but ANOVA test does not indicate which models were different. Therefore, we performed a multiple pairwise paired t-tests between models, in order to determine which models had a significant difference. Moreover, p-values were adjusted using the Bonferroni multiple testing correction method [36]. Bonferroni correction is a simple method which is used commonly to solve the multiple testing problem (i.e., when the number of tests (null hypothesis) increased, the problem of getting a significant difference due to chance is increased too). Simbly, the Bonferroni correction sets the significance by dividing α by the number of tests being performed [36]. Pairwise comparisons with a Bonferroni adjustment revealed that not all the pairwise differences between models were statistically significantly different (i.e., not all $p - values <= \alpha$). Specifically, model 16 and model 15, where the number of steps to forecast is 1 week, were significantly different from models (1 to 12) which predicted three hours ahead or less. Model 13 and model 14 predicted one day ahead and they were significantly different from most models (1 to 12) that predicted 3 hours ahead or less, except

```
================================================================
Repeated measures ANOVA
================================================================
Error: data
          Df    Sum Sq   Mean Sq F value Pr(>F)
Residuals  9 6.533e-05 7.259e-06

Error: data:model
           Df    Sum Sq   Mean Sq F value Pr(>F)
model      15 0.0007337 4.892e-05    25.4 <2e-16 ***
Residuals 135 0.0002600 1.930e-06
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 6.13: The results of repeated measures ANOVA test on GRU models

models that used a bigger batch size (e.g, model 14 is significantly different from models 2, 4, 6, and 12). The results from pairwise comparisons is provided in Appendix E. It appears from the results above that models that predicted one day ahead or one week ahead had a better performance than models which predicted up to three hours ahead using small batches, and there is no significant difference between the performance that predicted one day ahead and those which predicted one week ahead.

6.4.4 Results and Discussion

To evaluate the performance of the GRU models, we applied different forecasting models on the same datsets to forecast one day ahead, the choice of number of steps to forecast was in order to test the performance of the propsed models under a complex seasonality. Accordingly, we compare GRU models to a deep LSTMs model, shallow MLP models and TBATS model.

6.4.4.1 Deep LSTMs Model

Firstly, a deep LSTM model was applied on the same ten datasets. As mentioned in Section 2.7, the RNNs have been used successfully in literature for sequence learning tasks [120, 236], LSTMs allow RNN to learn temporal dependency and have been used for forecasting problems [228, 292]. Accordingly, we applied LSTMs model with the same inputs and the target outputs of the GRU

models in the previous experiment. Each dataset was divided into three sub-sets: 70% training set; 15% validation set and 15% for testing. The experiment was run in a similar way to the setting of the GRU models; we generated random batches from the training set, each consisted of 108 sequences, instead of using the entire training set. The LSTM models on the datasets were implemented using three layers (two hidden layers and one output layer); the first layer mapped the inputs from the dataset onto 128 outputs, this followed by a fully-connected (dense) layer of 128 neurons, then another fully-connected layer was added which mapped 128 values down to only 6, using the hard-sigmoid function. The details of the layers are shown in Figure 6.14.

In addition, the loss function was calculated for the as the previous GRUs models and the performance measures are shown in Figure 6.15, and we explained in the previous section, each measure value is the mean of measure values on the six outputs. By comparing these results with results from applying GRU model 14 which has same number of layers and number of neurons in each layer, and provides forecasting for the same number of steps (i.e., one day ahead); the mean MSE and mean MAE for LSTM model are 0.00882 and 0.06243 respectively, slightly higher than the mean MSE and mean MAE for GRU model are 0.00761 and 0.05830 respectively, these results will be discussed further later in this section.

Layer (type)	Output Shape	Param #
'lstm_1_Input'	(None, None, 10)	0
lstm_1 (LSTM)	(None, None, 128)	71168
dense_1 (Dense)	(None, None, 128)	16512
dense_2 (Dense)	(None, None, 6)	774

Total params: 88,454
Trainable params: 88,454
Non-trainable params: 0

Figure 6.14: Details of layers in LSTM model

Model	Metric	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7	Dataset8	Dataset9	Dataset10	mean
LSTM	Loss	0.00865	0.007817	0.007479	0.006471	0.008476	0.009554	0.008054	0.008781	0.010328	0.009155	0.008476486
	MAE	0.063376	0.060537	0.058672	0.05587	0.061585	0.068622	0.061782	0.063429	0.066756	0.063657	0.062428605
	MSE	0.009188	0.00806	0.007665	0.006937	0.008754	0.009859	0.008349	0.009098	0.010886	0.009369	0.008816672

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
LSTM model	Dataset1	0.009118	0.00369	0.018498	0.014081	0.004999	0.004742	0.009188025
	Dataset2	0.006857	0.003933	0.015377	0.010774	0.005094	0.006327	0.008060479
	Dataset3	0.007997	0.004312	0.014176	0.010005	0.004661	0.00484	0.007665102
	Dataset4	0.005936	0.003552	0.014115	0.008845	0.004496	0.00468	0.006937188
	Dataset5	0.00778	0.00348	0.01878	0.012756	0.004345	0.00538	0.008753574
	Dataset6	0.007546	0.004918	0.017463	0.017	0.00521	0.007019	0.009859298
	Dataset7	0.006172	0.004862	0.016388	0.012641	0.004759	0.005275	0.00834938
	Dataset8	0.008272	0.004076	0.018581	0.012901	0.004327	0.006432	0.009098474
	Dataset9	0.009989	0.004292	0.025866	0.014007	0.004528	0.006636	0.010886417
	Dataset10	0.008128	0.00349	0.022058	0.012837	0.003888	0.005813	0.009368784

Figure 6.15: The performance evaluation of the LSTM model on the test sets

6.4.4.2 Shallow MLP Models

Secondly, shallow Multi-Layer Perceptron (MLP) models were applied to the same datasets using three different settings, the Multi-Layer Perceptron (MLP) models have been used widely in literature for time series forecasting [16, 169, 265, 275]. In particular, three sets of inputs were used for this testing, as follows:

1. the month, the day of the week, time of day, and the occupancy of the parking lots (the same attributes used in the above GRUs models).

2. the week of the academic year, the day of the week, the time of day, and the occupancy of the parking lots.

3. the time of day, and the occupancy of the parking lots.

Where the target outputs for all these sets were the occupancy of the parking lots. Furthermore, similar to the GRUs models in the previous section, each dataset was divided into three sub-sets: 70% training set; 15% validation set, to monitor the model's performance; and 15% for testing. After this, the data was scaled as in the previous experiments. We applied the MLP on the same ten datasets for this experiment. There were 5,940 observations in each dataset, and there were 10 inputs and 6 outputs (9 inputs in the third experiment) to each model. The testing proceeded in a similar way to the testing of the GRUs models; we generated random batches from the training set instead of using the entire training set for training the MLP models — with 25 epochs, and 6 steps for each epoch. The MLP models were implemented using two layers (one hidden layer and one output layer), the first layer mapped the inputs to 512 outputs, and the choice of this number was based on trying different values to see which one fitted best with the data set; then we add a fully-connected (dense) layer which mapped the 512 values down to only 6 values, using the hard-sigmoid function. The details of the layers are shown in Figure 6.16.

Layer (type)	Output Shape	Param #
dense_1_Input	(None, None, 10)	0
dense_1 (Dense)	(None, None, 512)	5632
dense_2 (Dense)	(None, None, 6)	3078

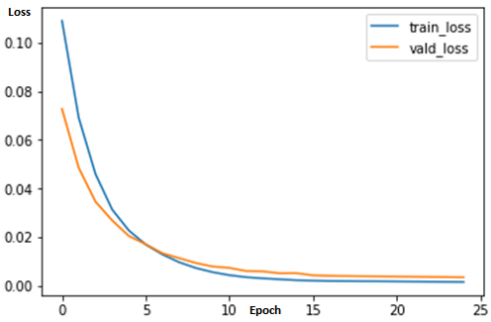
Total params: 8,710
Trainable params: 8,710
Non-trainable params: 0

Layer (type)	Output Shape	Param #
dense_1_Input	(None, None, 9)	0
dense_1 (Dense)	(None, None, 512)	5120
dense_2 (Dense)	(None, None, 6)	3078

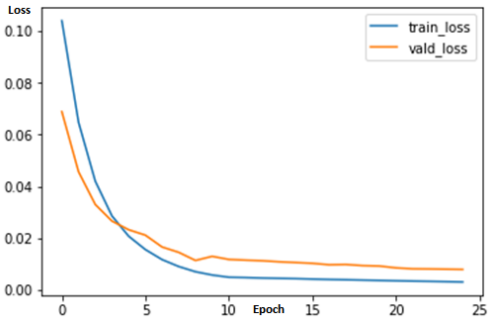
Total params: 8,198
Trainable params: 8,198
Non-trainable params: 0

Figure 6.16: Details of layers in MLP models

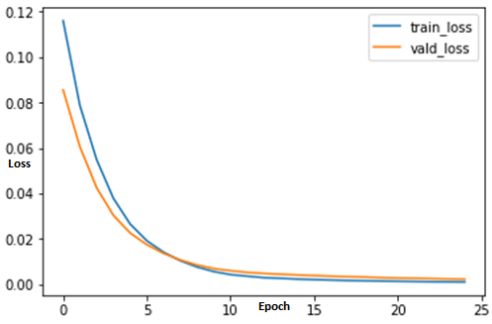
Moreover, Mean Squared Error (MSE) was used to calculate the loss function, and RMSprop with an initial learning rate of $1e-3$ was used as the model optimizer. The diagrams in Figure 6.17 show the loss function (for Dataset 1 only) demonstrated by the three experiments. The various results from applying the three MLP models on the test sets are shown in Figure 6.18. It appears that the first model, which use same inputs and outputs of GRU models, has the least mean MSE and mean MAE values: 0.00340 and 0.04268, where the third model which used only the time of the day and the occupancy of parking lots as inputs, has the highest mean MSE and mean MAE values: 0.0095 and 0.0713. These results will be discussed in detail later in this section, and the details of MSEs of the MLP models on all datasets are presented in Appendix E.



(a) the 1st experiment: month, day, time, and occupancy inputs



(b) the 2nd experiment: week, day, time, and occupancy inputs



(c) the 3rd experiment: time, and occupancy inputs only

Figure 6.17: The loss of the MLP models on the University parking lots

Model	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7	Dataset8	Dataset9	Dataset10	Mean Loss
1	0.004697	0.003622	0.001862	0.003481	0.003767	0.003923	0.002831	0.003025	0.004388	0.0024803	0.0034076
2	0.007833	0.010313	0.003876	0.005754	0.004024	0.009558	0.009491	0.011864	0.006838	0.0095241	0.00790763
3	0.010689	0.010429	0.010507	0.011355	0.00992	0.003526	0.005202	0.008758	0.009004	0.0162857	0.00956764

Model	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7	Dataset8	Dataset9	Dataset10	Mean MSE
1	0.004691	0.003618	0.001866	0.003478	0.003759	0.003913	0.002827	0.003019	0.004375	0.0024779	0.00340244
2	0.007814	0.010269	0.003867	0.005733	0.004013	0.009522	0.009447	0.011806	0.00681	0.0094825	0.00787631
3	0.010637	0.010376	0.010458	0.011318	0.009874	0.003513	0.00518	0.008718	0.008961	0.016239	0.00952751

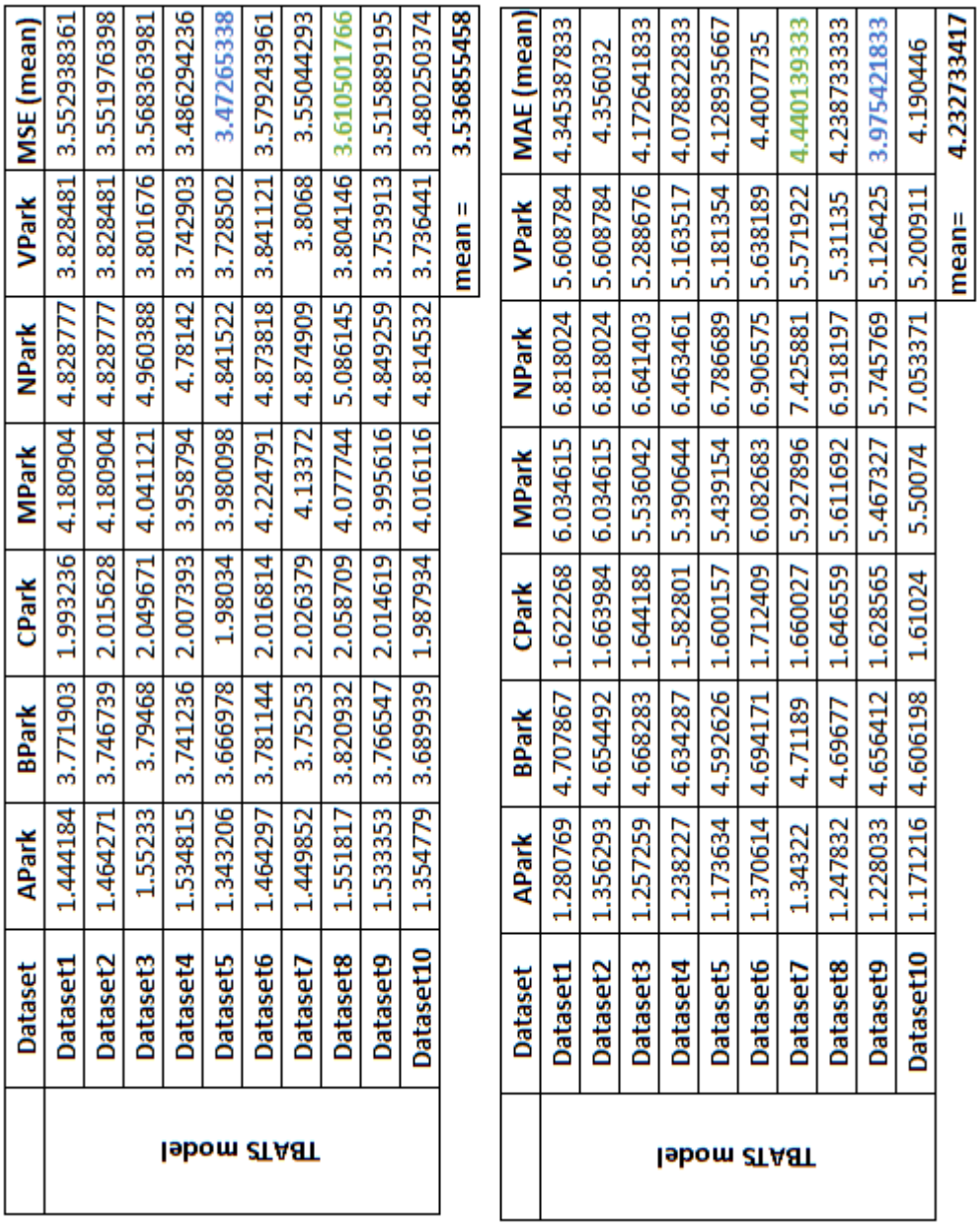
Model	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7	Dataset8	Dataset9	Dataset10	Mean MAE
1	0.050971	0.044929	0.030431	0.043991	0.045455	0.046676	0.038691	0.040351	0.04923	0.0360408	0.0426765
2	0.064761	0.075704	0.047356	0.057055	0.046741	0.070412	0.07176	0.081207	0.061139	0.0728624	0.06489966
3	0.076424	0.075669	0.076061	0.079152	0.073515	0.043356	0.053627	0.069577	0.070281	0.0953237	0.07129854

Figure 6.18: The performance evaluation of the MLP models on the test sets

6.4.4.3 TBATS Model

Moreover, we decided to compare the GRU results with those that could be obtained from purely statistical methods. We sought to find a statistical method that deals with multivariate time series and complex seasonality at the same time. So, for this experiment, we settled on TBATS [76] as the method to be used, it is a statistical forecasting technique that deals with complex seasonality in automated way with no seasonality constraints, and it was described previously in Section 2.6. This method can deal efficiently with complex seasonality exhibited by isolated time series (each parking lot alone), each dataset was splitted as the previous experiments into 70% and 15% for training and validation sets and 15% for testing sets. Figure 6.19 shows the performance metrics of the forecasting model in on the test data, these metrics were aggregated based on the mean value (i.e., for each dataset, the total sum of performance measure values on multiple parking lots was divided by the number of parking lots). The mean of performance values are higher than those for GRU models with MSE and mean MAE values: 3.5369 and 4.2327 respectively. Moreover, screen-shots of predictions from the proposed models in this chapter are provided in Appendix F.

It is clear from comparing the results in Figure 6.11, Figure 6.15, Figure 6.18, and Figure 6.19 that the GRU models, LSTM model and MLP models perform better than the TBATS statistical model, the performance measures of GRU models are always below one. For the same number of steps to forecast (one day ahead) on the same ten datasets, the MSE values of GRUs model 14 range between 0.00674 and 0.00928, and MAE values range between 0.05539 and 0.06039. Whereas MSE values of the LSTM model range between 0.006937 and 0.01886 and MAE values range between 0.05587 and 0.068622. The MSE values of the MLP model 1 range between 0.00187 and 0.004 and MAE values range between 0.03043 and 0.05971. The MSE values of the TBATS model range between 3.4726 and 3.6105 and MAE values range between 3.9754 and 4.4401. In addition, it is



	Dataset	APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
TBATS model	Dataset1	1.444184	3.771903	1.993236	4.180904	4.828777	3.828481	3.552938361
	Dataset2	1.464271	3.746739	2.015628	4.180904	4.828777	3.828481	3.551976398
	Dataset3	1.55233	3.79468	2.049671	4.041121	4.960388	3.801676	3.568363981
	Dataset4	1.534815	3.741236	2.007393	3.958794	4.78142	3.742903	3.486294236
	Dataset5	1.343206	3.666978	1.98034	3.980098	4.841522	3.728502	3.47265338
	Dataset6	1.464297	3.781144	2.016814	4.224791	4.873818	3.841121	3.579243961
	Dataset7	1.449852	3.75253	2.026379	4.13372	4.874909	3.8068	3.55044293
	Dataset8	1.551817	3.820932	2.058709	4.077744	5.086145	3.804146	3.610501766
	Dataset9	1.533353	3.766547	2.014619	3.995616	4.849259	3.753913	3.515889195
	Dataset10	1.354779	3.689939	1.987934	4.016116	4.814532	3.736441	3.480250374
							mean =	3.536855458

	Dataset	APark	BPark	CPark	MPark	NPark	VPark	MAE (mean)
TBATS model	Dataset1	1.280769	4.707867	1.622268	6.034615	6.818024	5.608784	4.345387833
	Dataset2	1.356293	4.654492	1.663984	6.034615	6.818024	5.608784	4.356032
	Dataset3	1.257259	4.668283	1.644188	5.536042	6.641403	5.288676	4.172641833
	Dataset4	1.238227	4.634287	1.582801	5.390644	6.463461	5.163517	4.078822833
	Dataset5	1.173634	4.592626	1.600157	5.439154	6.786689	5.181354	4.128935667
	Dataset6	1.370614	4.694171	1.712409	6.082683	6.906575	5.638189	4.4007735
	Dataset7	1.34322	4.71189	1.660027	5.927896	7.425881	5.571922	4.440139333
	Dataset8	1.247832	4.69677	1.646559	5.611692	6.918197	5.31135	4.238733333
	Dataset9	1.228033	4.656412	1.628565	5.467327	5.745769	5.126425	3.975421833
	Dataset10	1.171216	4.606198	1.61024	5.50074	7.053371	5.200911	4.190446
							mean=	4.232733417

Figure 6.19: The performance evaluation of the TBATS model on the test sets

apparent from comparing the first MLP model, which uses the same inputs of the GRU models that the simple MLP model has a lower performance values than the GRUs model, with mean MSE value 0.00340 less than the mean MSE value obtained of the GRUs model 14: 0.00760, and with mean MAE values: 0.04268 for the MLP model, and 0.05830 for the GRUs model. A possible explanation for this might be that GRUs need more training data (e.g., one year), so they can model the different seasonality in the time series. In order to find if the differences in the performance of these models were significant, we run repeated measures one-way ANOVA test which followed by a pairwise comparisons with a Bonferroni adjustment (similar to the test in Section 6.4.3). The test result in Figure 6.20 shows that these models were significantly different, $F - statistic$ is greater than $critical - F - value$: $58343 > 2.96035135$, however, the pairwise t-tests indicate that the difference between GRU, LSTM, and MLP is not significant; because all $p - values$ between each pair of these models were greater than α, so we could not reject the null hypothesis. The only significant difference was between these models and the TBATS model, $p - values$ between each pair of these models were less than α. These results would seem to suggest that the deep GRU models would outperform the statistical forecasting techniques which operate on multivariate time series data streams. However, GRU models might not be the best approach for forecasting multivariate time series from dynamic streams, as the simple MLP models achieved similar performance.

6.5 Chapter Summary

There are few real-time parking availability systems due to the high cost of the technologies required for building and maintaining the necessary real-time parking information. The increasing proliferation of IoT technologies across many real-world applications, where heterogeneous data streams are produced by multiple sources rapidly, provides the means to develop applications which can provide

```
============================================================
 Repeated measures ANOVA
============================================================
Error: dataset
          Df   Sum Sq   Mean Sq F value Pr(>F)
Residuals  9 0.004812 0.0005347

Error: dataset:modelname
          Df Sum Sq Mean Sq F value Pr(>F)
modelname  3  93.47  31.157   58343 <2e-16 ***
Residuals 27   0.01   0.001
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


============================================================
 Pairwise t-tests comparison
============================================================

data:  valueOferror and themodel

     GRU    LSTM   MLP
LSTM 1      -      -
MLP  1      1      -
TBAT <2e-16 <2e-16 <2e-16

P value adjustment method: bonferroni
```

Figure 6.20: The results of repeated measures ANOVA test on the four models: GRU, LSTM, MLP, and TBATS

real or near-real time forecasting from these time series. However, the accelerated growth of time series data streams, in terms of their size and complexity, has become a major obstacle to the application of existing statistical techniques to these data streams.

This chapter presents a system which sought to produce near-real time predictions in a car-parking lots dynamic environment equipped with a number of different IoT devices. Furthermore, it provided an empirical exploration of the application of Gated Recurrent Units (GRUs) models, which are designed to handle sequence encoding and decoding. This exploration attempted to determine the efficiency of applying these deep learning models to produce near-real time predictions in dynamic environments, over multiple time steps, and to explore their (these models') ability to handle the multiple variables, and complex seasonality which are encountered in such data streams. The contribution of this chapter is two-fold. First, it shows the development of a model which provides an as-accurate-as-possible near-real- time parking availability forecast, across a

variety of parking lots, by using the GRUs technique and the multivariate time series collected from the IoT devices in the parking lots. This is unlike most of the solutions which have already been proposed in the literature, many of which are based on the installation of additional hardware, e.g., cameras or sensors at each parking space [112, 115, 233, 293]; such approaches imply extra installation costs and ongoing maintenance expenses. Second, the chapter presented an empirical exploration of the use of deep learning approaches for forecasting multiple time series obtained from heterogeneous data streams. In particular, different GRU models were tested on parking information which was collected from a variety of IoT devices. The experiments were conducted on six parking lots at the University of Essex, where there were six smart pay stations and only three camera sensors. To the best of our knowledge, this is the first study which has empirically explored GRU models in association with time series forecasting. The results show the effectiveness of using GRU models in terms of providing accurate predictions in dynamic environments. Moreover, the results indicate that GRU models outperform the existing statistical methods, for this task, since GRU models can handle the complex seasonality across multiple time series data streams. However, the results show that there is no significant difference in the performance between a simple Multi-Layer Perceptron model (MLP) and a deep GRUs model. We argue that GRU models may perform equal to simple MLPs models in the task of forecasting time series. In fact, deep is not necessarily better; deep networks often have more parameters, thus, they are more likely to be over-fitted. However, this needs a further investigation using larger datasets, and datasets from different domains.

Chapter 7

Conclusion

This study was undertaken for the purpose of exploiting real-time heterogeneous data streams produced by a variety sources. It explored data mining techniques which operate on non-stationary streams of this kind. In particular, it investigated problems which are encountered in the course of mining such data streams that exhibit skewed distributions in non-stationary environments. Novel methods were developed, capable of learning from such data streams, which can capture the dynamic of the patterns which emerge and thus predict future behaviour relating to items in these streams. The novelty of the techniques presented here lies in the fact that they tackle the joint problems of concept drift and multi-class imbalance in data streams scenarios. A new technique was presented to extend concept adaptation techniques into class imbalance scenarios (more than two classes). Moreover, an algorithm using dynamically calculated thresholds was presented aimed at discovering patterns from evolving streams, tackling the changes in emerging patterns, and overcoming the rare patterns problem. Moreover, a new forecasting model for time series data streams was developed for the purpose of handling the complex seasonality in these data streams, and providing as-accurate-as-possible predictions across multiple time series data streams generated by dynamic environments.

This chapter provides a brief summary of the work presented in this thesis; it identifies the main contributions of this work, and highlights the findings, indicating the strengths and weaknesses of the approaches taken. The chapter then describes some potential areas for future research by which this work can be extended.

7.1 Summary

In summary, this thesis aims to contribute to the research focused on mining evolving data streams which have been generated from heterogeneous sources, and to addressing of the multi-class problem related to such streams.

Chapter 2 presented a survey of the state-of-the-art work related to mining data streams. That chapter identified the challenges that may emerge when mining data streams; particularly, it focused on three problem: concept drift, class imbalance, and the discovering of patterns from evolving data streams and thus the tackling of the change in the patterns which are encountered. It illustrated how drifts occur in streaming scenarios, and discussed the methods presented in the literature relating to concept drift adaptation. A detailed description of the class imbalance problem and the impact of this problem on data mining techniques was given, in terms of developments and evaluations. In addition, it discussed the difficulty of discovering patterns from evolving data streams using the existing approaches - which require multiple scans of the data set and predefined thresholds values. In addition, an overview related to time series forecasting was provided, with regard to both machine learning approaches designed for forecasting time series and also current statistical techniques. Furthermore, a brief discussion was presented with regard to deep learning approaches and artificial neural networks, focusing on the Gated Recurrent Unit models.

Chapter 3 presented the formalisation of the problem of working with multiple streams of data from a variety of IoT devices. An illustrative scenario involving a

number of different data streams derived from the same dynamic environment — a scenario concerned with car movements within a parking lots environments — was presented in Chapter 3. The chapter identified the characteristics of the sample, data from the University of Essex parking lots, which was selected to represent the research problem and which was used to create the simulation model. This simulation sought to produce the necessary heterogeneous data streams which change dynamically and which exhibit multi-class imbalance problem. This simulation can also serve as a benchmark to facilitate the development and the evaluation of data mining techniques which operate over non-stationary data streams.

Data mining techniques presented in the literature capable of classifying non-stationary data streams which exhibit internal imbalance were focused on the use of ensemble classifiers [137, 271, 274]. Chapter 4 presented means by which concept drift adaptation techniques can be extended, specifically, the developing of adaptive learning models and training set modification into the area of imbalanced data streams. While the work presented in the literature focused on only two class problems [46, 137, 198, 271, 274], this chapter addressed multi-class imbalance problems and presented a new method, the ICE-Stream technique, capable of classifying evolving imbalanced data streams. An adaptive learning model was developed using a time-window based approach. A window of the instances which were found to have been classified incorrectly was maintained and used to modify the training window of the adaptive classifier which was used for the subsequent time-windows. The experiments were conducted on the data streams derived from the simulation model, and from two other, real, data sets. As far as we know, no previous study has investigated the application of adaptive learning models or the modification of training sets for tackling the class imbalance problem in non-stationary streams [137, 274]. The results obtained from the ICE-Stream technique, which were presented in this chapter, showed the efficiency of the proposed technique in adapting to changes in the underlying concepts being represented in these streams, and for detecting instances from minority classes.

Extracting patterns of interest from evolving data streams is difficult and time consuming as a huge number of patterns can emerge from such streams; therefore, the patterns should be identified in relation to their to level of interest. Chapter 5 was focused on discovering patterns from differing data streams which have skewed distributions and which emerge from dynamic environments. The chapter presented a novel technique, the FP-EStream, for extracting both frequent and rare patterns from such streams. This technique was developed based on the integration of two mining techniques: classification, used to extract the only patterns of interest; and frequent pattern mining, using a variant of FP-Growth algorithm and a dynamically calculated support threshold. The chapter showed the way in which the FP-Growth technique [123] was extended into evolving data stream scenarios. Thus, a new method, which uses the joint probabilities calculated by the naive Bayes classifier, was presented for building the frequent items list, FIS, for the FP-Growth algorithm [123], and then construction of the FP-tree using the classifier attributes. It also presents a novel method for defining a dynamically calculated threshold; this threshold was computed for each time-window, based on the minimum joint probability derived from the classifier; it was then used to identify patterns and tackle the drifts in these patterns as obtained from the evolving streams. The experiments were conducted on the data streams derived from the simulation model and also on a real data set. The results showed the usability of the proposed technique in terms of its ability to reduce the number of meaningless patterns returned. In addition, the results supported the merit of using a dynamically calculated, instead of a fixed, threshold to detect the rare patterns over evolving imbalanced data streams.

Time series data derived from dynamic environments capture their dynamic behaviours and provide the means by which to monitor and predict changes in these environments. With the proliferation of data streams generated from our everyday activities, there is an increasing necessity to develop time series forecasting techniques which can deal with data generated at high speed and which exhibit

the presence of multiple variables and complex seasonality. Existing statistical forecasting techniques cannot handle complex seasonality in multiple time series data [40, 41, 144]. Chapter 6 described a new method for providing near-real time prediction for a car-parking system, across a variety of parking lots. Unlike existing parking prediction methods which require the installation of extra hardware, the proposed method used the data streams derived from the existing IoT devices, deployed in the parking lot infrastructure, to develop Gated Recurrent Unit models. The forecasting models presented were capable of providing accurate near-real time prediction over multiple time intervals. In addition, this chapter presented an empirical exploration of using GRU models to forecast time series derived from dynamic environments. To the best of our knowledge, no previous work has investigated the use of GRU models for forecasting time series. The results obtained revealed that the exiting statistical forecasting techniques can be outperformed by the GRU models. Interestingly, there is no significant difference between the results obtained from the deep learning approach using GRU models and a simple Multi-Layer Perceptron (MLP) model, the results indicated that GRU models may perform equal or worse than simple MLP models in the task of forecasting time series.

7.1.1 Contributions

The contributions of the work presented here can be summarised for the reader as follows: first, we extended the adaptation techniques of concept drift into imbalanced classes scenarios. The ICE-Stream technique, an adaptive classifier using naive Bayes and with an incremental nature, was developed in order to classify heterogeneous streams over a sequence of titled time windows. The adaptive learner worked based on a windows approach; thus, it was retrained with the arrival of each new window batch of the data stream. As far as we know, the ensemble classifiers technique is the only approach which has been used (up until now) for

classifying non stationary data streams which exhibit the presence of class imbalance. However, there are two other adaptation techniques which have not as yet been investigated in relation to the class imbalance problem often encountered in evolving streams. This present study represents the first time that an adaptive learner has been used, even experimentally, to classify evolving data streams and handle class imbalance across more than two classes.

Then, we maintained a window of the instances which had been incorrectly classified by the adaptive learner; this window was then used to modify the training set to be used by the adaptive learner; the contents of this window were added to the training sets for subsequent windows. This was in order to enhance the performance of the classifier of the ICE-Stream method and to reduce the number of incorrectly classified instances (often these instances belong to a minority class). To the best of our knowledge, the ICE-Stream method presented here is the first proposal which has been developed to tackle the multi-class (more than two classes) imbalance problem as it can occur in evolving streams.

Moreover, we presented a novel technique, FP-EStream, for detecting patterns from evolving imbalanced data streams, which uses a variant of the FP-Growth mining technique. The means of constructing the frequent item list was changed; this list was built using the joint probabilities derived from the classifier, whereby items were ordered according to their class attributes. Unlike other pattern discovery methods presented in the literature [156, 179, 258], this technique does not generate candidates, and it accelerates the algorithm execution time by reducing number of data set scans. This methodology also helps to reduce the number of patterns derived from the huge amounts of data items likely to be encountered in evolving data streams — by omitting meaningless patterns.

In addition, we presented a dynamically calculated threshold which was used to identify patterns which emerged over titled time windows. The threshold values varied across different time windows, and they were computed based on the minimum join probabilities derived from the classier used in the ICE-Stream tech-

nique. In contrast to other methods which utilise predefined single/ or multiple threshold(s) to discover patterns from data sets, the FP-EStream's dynamically calculated threshold value allows the method to detect rare patterns appearing in imbalanced non-stationary data streams.

Furthermore, we provided an empirical exploration of the forecasting of multiple time series data streams which exhibit more than one seasonal pattern, using GRU based models. The results showed the ability of GRU models to provide an accurate prediction over multiple time steps across a variety of time series data streams. The performance of the GRUs model used here surpassed the performance of current statistical methods, which can either deal with complex seasonality in only one time series, or deal with multiple time series which exhibit only one seasonal component. This is the first time that GRU based models were experimentally tested on forecasting multiple time series.

Moreover, we presented a new method for predicting car-parking space availability using deep learning GRU models and multivariate time series (occupancy/time) derived from heterogeneous data streams, where these data streams were generated by diverse kinds of IoT devices already deployed in the parking lot infrastructure. This is unlike the parking prediction solutions discussed in the literature which entail extra costs relating to the installation and maintenance of special sensors at each parking space.

7.2 Future Work

This research has enhanced the data mining techniques which are available to operate over evolving data streams generated from a variety of sources, especially with regards to the multi-class imbalance problem. Despite the fact that a wide range of issues have been addressed in this study, and that it has achieved its intended aim and objectives, further improvements to the proposed techniques can be pointed to; further work along these lines would extend this study and

provide additional insights into mining techniques for evolving imbalanced data streams.

- Although we have established the functionality of an adaptive learning technique which can cope with class imbalance in non-stationary streams, it would nevertheless be possible, and possibly instructive, to experiment with different types of base classifiers in place of the naive Bayes classier (e.g., the k-Nearest Neighbour algorithm). This in order to support empirically the insights from applying adaptive learners into imbalanced evolving data streams and to improve the accuracy of adaptive learners on the minority classes.

- To evalute the adaptive learner performance in this work, we used Kappa statistics [65] that take class imbalance into account, comparing relative to the ground truth, and the accuracy to identify the performance of the proposed technique and to compare it with other approaches. One of the most significant current discussions in the literature is how streaming classifiers should be evaluated [31, 137]. An important limitation of this research lies in the fact that not only we need measures for evaluating single aspects of stream mining algorithms, but also ways of combining several aspects (i.e., concept drift, imbalanced distributions) into the evaluation procedure [168], for example, more performance evaluation metrics for data stream are required which are sensitive to multi-class imbalance [31] and which can detect change in more than one minority class.

- Although the proposed ICE-Stream method was tested on real-world datasets, the generalisation of the results obtained in this investigation is subject to certain limitations; for instance, the choice of the window size in terms of the adaptive learner's input. Further research regarding the choice of the size of the time based windows would be worthwhile. It would be interesting to attempt to adjust the size of the window dynamically while the mining

and processing of data streams carries on, in order to find the appropriate window size in cases where there is a lack of domain knowledge.

- The issue of how to maintain the minority window for the ICE-Stream method is an intriguing one which could be usefully explored further and improved in future research, in terms of deciding when items in the window should be discarded. For example, minority classes may evolve and instances stored in the minority window which belong to the old concept may affect the classifier performance (training using old concepts).

- Although the proposed FP-EStream technique was tested, here, only on a car parking lot environment, the technique itself could be adapted and used across many different application domains: for discovering patterns in IoT streams (e.g., tube fault diagnosis, or water leakages monitoring), where we have meters, sensors, and open-data sources. The method presented in this study could be applied to the analysis of these streams in relation to the detection of anomalous behaviour patterns (e.g. leakages), and may help reacting quickly to critical situations and so support predictive maintenance.

- Considerably more work on very large data streams and on using data sets from different domains, will need to be done in order to determine if the deep learning technique using the GRU based models are really appropriate for forecasting multiple time series derived from heterogeneous sources and which exhibit complex seasonality. Deep learning models can be easily over-fitted when they are trained on short time series data streams, this may lead to poor generalisation and worsen the performance of the forecasting model. These models should be investigated for both short-term and long-term forecasting.

We aim to explore some of these avenues in future research.

Acronyms

AL Active Learning. 17

ANN Artificial Neural Network. 35

ANOVA Analysis of Variance. 142, 143, 153

ARF Adaptive Random Forest algorithm. 25

ARIMA Auto Regressive Integrated Moving Average. 32, 33

AUE Accuracy Updated Ensemble technique. 24, 25

AWE Accuracy Weighted Ensemble technique. 24, 25

BATS Box-Cox transformation, ARMA errors, Trend and Seasonal components. 33

CEPs Complex Event Processing Systems. 19

CFP-Growth Conditional Frequent Pattern-Growth. 29, 30

CFP-Growth++ Conditional Frequent Pattern-Growth++. xi, 29, 106–108

CNNs Convolutional Neural Networks. 35

DBMSs Database Management Systems. 19

DBNs Deep Learning Belief Networks. 34

DDM-OCI Drift Detection Method for Online Class Imbalance. 67

DL Deep Learning. 17, 34

DSMSs Data Stream Management Systems. 19

ES Exponential Smoothing. 32

FP-EStream Frequent Patterns from imbalanced Evolving Streams. xi, 8, 95, 97–99, 101, 103, 104, 106–109, 111, 113, 114, 159, 161, 162, 164

FP-tree Frequent Pattern Tree. 29, 30, 99, 101, 102, 109, 113, 114, 159

GRU Gated Recurrent Unit. 8, 35–38, 117, 118, 121, 123, 133, 135–137, 144, 145, 147, 148, 151, 153–155, 157, 160, 162, 164, 231

ICE-Stream Imbalanced Classes in Evolving Streams. 7, 68, 73, 75, 76, 81–83, 85–93, 101, 109, 158, 160, 161, 163, 164

IoT Internet of Things. 1–7, 40, 41, 43, 44, 50, 52, 53, 62, 75, 97, 103, 115, 117, 119, 121, 123, 153–155, 157, 160, 162, 164

KDD Knowledge Discovery in Databases. 16

LMS least minimum support. 30

LSTM Long-Short Term Memory. viii, 35, 36, 38, 39, 118, 121, 123, 144, 145, 151, 153

MAE Mean Absolute Error. 32, 137, 142, 145, 148, 151, 153

MIS minimum support thresholds. 29, 30

MLP Multi-Layer Perceptron. viii, 35, 118, 123, 144, 147, 148, 151, 153, 155, 160, 236

MOA Massive Online Analysis. 77, 78, 85

MSE Mean Squared Error. 32, 136, 137, 142, 145, 148, 151, 153, 231, 236

OOB Oversampling-based Online Bagging. 26

RL Reinforcement Learning. 17

RMSE Root Mean Squared Error. 32

RNN Recurrent Neural Network. 35, 36, 119, 144

ROC curve Receiver Operating Characteristic curve. 28

SADEs Stacked Denoising Auto-Encoders. 33

SARIMA the seasonal ARIMA. 33

SEA Streaming Ensemble Algorithm. 24

SL Supervised Learning. 16

SSL Semi-supervised Learning. 16

SVR Support Vector Regression. 34

TBATS Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components. viii, x, 33, 34, 123, 144, 151, 153

UL Unsupervised Learning. 16

UOB Undersampling-based Online Bagging. 26

VARIMA Vector ARIMA. 33

References

[1] Keras documentation: Optimizers. online; https://keras.io/optimizers; accessed 10 March 2019.

[2] C. C. Aggarwal. On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd international conference on Very large data bases*, pages 607–618. VLDB Endowment, 2006.

[3] C. C. Aggarwal. *Data streams: models and algorithms*, volume 31. Springer Science & Business Media, 2007.

[4] C. C. Aggarwal. *Data mining: the textbook*. Springer, 2015.

[5] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003.

[6] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[7] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

[8] M. Almuammar and M. Fasli. Pattern discovery from dynamic data streams using frequent pattern mining with multi-support thresholds. In *2017 International Conference on the Frontiers and Advances in Data Science (FADS)*, pages 35–40. IEEE, 2017.

[9] M. Almuammar and M. Fasli. Learning patterns from imbalanced evolving data streams. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2048–2057. IEEE, 2018.

[10] M. Almuammar and M. Fasli. Deep learning for non-stationary multivariate time series forecasting. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2097–2106. IEEE, 2019.

[11] P. Anantharam, P. Barnaghi, K. Thirunarayan, and A. Sheth. Extracting city traffic events from social streams. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(4):43, 2015.

[12] H. C. Andrade, B. Gedik, and D. S. Turaga. *Fundamentals of stream processing: application design, systems, and analytics*. Cambridge University Press, 2014.

[13] S. M. Arisona, G. Aschwanden, J. Halatsch, and P. Wonka. *Digital Urban Modeling and Simulation*, volume 242. Springer, 2012.

[14] J. S. Armstrong. *Principles of forecasting: a handbook for researchers and practitioners*, volume 30. Springer Science & Business Media, 2001.

[15] T. B. P. Association. Searching for parking spaces report, 2016. online; https://www.britishparking.co.uk/News/motorists-spend-nearly-four-days-a-year-looking-for-a-parking-space; accessed 10 February 2019.

[16] E. M. Azoff. *Neural network time series forecasting of financial markets*. John Wiley & Sons, Inc., 1994.

[17] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM, 2002.

[18] S. H. Bach and M. A. Maloof. Paired learners for concept drift. In *2008 Eighth IEEE International Conference on Data Mining*, pages 23–32. IEEE, 2008.

[19] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten. Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51:134–142, 2016.

[20] D. W. Bates, S. Saria, L. Ohno-Machado, A. Shah, and G. Escobar. Big data in health care: using analytics to identify and manage high-risk and high-cost patients. *Health Affairs*, 33(7):1123–1131, 2014.

[21] R. J. Bayardo Jr and R. Schrag. Using csp look-back techniques to solve real-world sat instances. In *Aaai/iaai*, pages 203–208. Providence, RI, 1997.

[22] M. Ben-Akiva, M. Bierlaire, H. Koutsopoulos, and R. Mishalani. Dynamit: a simulation-based system for traffic prediction. In *DACCORD Short Term Forecasting Workshop*, pages 1–12. Delft The Netherlands, 1998.

[23] M. Ben-Akiva, M. Bierlaire, H. N. Koutsopoulos, and R. Mishalani. Real time simulation of traffic demand-supply interactions within dynamit. In *Transportation and network analysis: current trends*, pages 19–36. Springer, 2002.

[24] V. Bewick, L. Cheek, and J. Ball. Statistics review 9: one-way analysis of variance. *Critical care*, 8(2):130, 2004.

[25] M. Beyer. Gartner says solving'big data'challenge involves more than just managing volumes of data. *Gartner. Archived from the original on*, 10, 2011.

[26] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[27] A. Bifet and R. Gavaldà. Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, pages 249–260. Springer, 2009.

[28] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148. ACM, 2009.

[29] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[30] A. Bifet, J. Read, I. Žliobaitė, B. Pfahringer, and G. Holmes. Pitfalls in benchmarking data stream classification and how to avoid them. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 465–479. Springer, 2013.

[31] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer. Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 59–68. ACM, 2015.

[32] A. Bifet, R. Gavaldà, G. Holmes, and B. Pfahringer. *Machine learning for data streams: with practical examples in MOA*. MIT Press, 2018.

[33] A. Bifet, J. Read, G. Holmes, and B. Pfahringer. Streaming data mining with massive online analytics (moa). *SERIES IN MACHINE PERCEPTION AND ARTIFICIAL INTELLIGENCE*, 83(1):1–25, 2018.

[34] D. Billsus and M. J. Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2-3):147–180, 2000.

[35] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[36] J. M. Bland and D. G. Altman. Multiple significance tests: the bonferroni method. *Bmj*, 310(6973):170, 1995.

[37] G. Bontempi, S. B. Taieb, and Y.-A. Le Borgne. Machine learning strategies for time series forecasting. In *European business intelligence summer school*, pages 62–77. Springer, 2012.

[38] R. J. C. Bose, W. M. van der Aalst, I. Žliobaitė, and M. Pechenizkiy. Handling concept drift in process mining. In *International Conference on Advanced Information Systems Engineering*, pages 391–405. Springer, 2011.

[39] G. E. Box and G. M. Jenkins. Time series analysis forecasting and control. Technical report, WISCONSIN UNIV MADISON DEPT OF STATISTICS, 1970.

[40] J. C. Brocklebank, D. A. Dickey, and B. Choi. *SAS for forecasting time series*. SAS institute, 2018.

[41] P. J. Brockwell, R. A. Davis, and M. V. Calder. *Introduction to time series and forecasting*, volume 2. Springer, 2002.

[42] J. Brownlee. *Deep Learning With Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras*. Machine Learning Mastery. Online; `https://machinelearningmastery.com/deep-learning-with-python/`; accessed 19 September 2018.

[43] J. BROŻYNA, G. Mentel, B. Szetela, and W. Strielkowski. Multi-seasonality in the tbats model using demand for electric energy as a case study. *Economic Computation & Economic Cybernetics Studies & Research*, 52(1), 2018.

[44] R. Bruns, J. Dunkel, H. Masbruch, and S. Stipkovic. Intelligent m2m: Complex event processing for machine-to-machine communication. *Expert Systems with Applications*, 42(3):1235–1246, 2015.

[45] D. Brzeziński and J. Stefanowski. Accuracy updated ensemble for data streams with concept drift. In *International conference on hybrid artificial intelligence systems*, pages 155–163. Springer, 2011.

[46] D. Brzezinski and J. Stefanowski. Prequential auc for classifier evaluation and drift detection in evolving data streams. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 87–101. Springer, 2014.

[47] D. Brzezinski and J. Stefanowski. Prequential auc: properties of the area under the roc curve for data streams with concept drift. *Knowledge and Information Systems*, 52(2):531–562, 2017.

[48] D. Brzezinski and J. Stefanowski. Ensemble classifiers for imbalanced and evolving data streams. *SERIES IN MACHINE PERCEPTION AND ARTIFICIAL INTELLIGENCE*, 83(1):44–68, 2018.

[49] W. Burghout, H. N. Koutsopoulos, and I. Andreasson. Hybrid mesoscopic–microscopic traffic simulation. *Transportation Research Record*, 1934(1): 218–225, 2005.

[50] F. Caicedo, C. Blazquez, and P. Miranda. Prediction of parking space availability in real time. *Expert Systems with Applications*, 39(8):7281–7290, 2012.

[51] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell. An overview of machine learning. In *Machine learning*, pages 3–23. Elsevier, 1983.

[52] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[53] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[54] O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 2010.

[55] C. P. Chen and C.-Y. Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information sciences*, 275: 314–347, 2014.

[56] M. Chen, Y. Ma, J. Song, C.-F. Lai, and B. Hu. Smart clothing: Connecting human with clouds and big data for sustainable health monitoring. *Mobile Networks and Applications*, 21(5):825–845, 2016.

[57] S. Chen, H. He, K. Li, and S. Desai. Musera: Multiple selectively recursive approach towards imbalanced stream data mining. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2010.

[58] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[59] F. Chollet. Keras: The python deep learning library, 27 March 2015. online; https://keras.io; accessed 28 November 2018.

[60] P. H. Christian Chabot, Chris Stolte. Tableau : An interactive data visualization software, 2003. online; https://www.tableau.com/; accessed 20 August 2016.

[61] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[62] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan. *Data mining: a knowledge discovery approach*. Springer Science & Business Media, 2007.

[63] K. J. Cios, W. Pedrycz, and R. W. Swiniarski. *Data mining methods for knowledge discovery*, volume 458. Springer Science & Business Media, 2012.

[64] L. Codeca, R. Frank, and T. Engel. Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE, 2015.

[65] J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[66] D. Cournapeau. scikit-learn: Machine learning in python, June 2007. online; https://scikit-learn.org; accessed 20 January 2019.

[67] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[68] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44 (3):15, 2012.

[69] A. Das, J. Gehrke, and M. Riedewald. Approximate join processing over data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 40–51. ACM, 2003.

[70] S. Das, W.-K. Wong, T. Dietterich, A. Fern, and A. Emmott. Incorporating expert feedback into active anomaly discovery. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 853–858. IEEE, 2016.

[71] A. V. Dastjerdi and R. Buyya. Fog computing: Helping the internet of things realize its potential. *Computer*, 49(8):112–116, 2016.

[72] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM journal on computing*, 31(6):1794–1813, 2002.

[73] A. P. Dawid. Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2):278–290, 1984.

[74] G. De Francisci Morales, A. Bifet, L. Khan, J. Gama, and W. Fan. Iot big data stream mining. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2119–2120. ACM, 2016.

[75] J. G. De Gooijer and R. J. Hyndman. 25 years of time series forecasting. *International journal of forecasting*, 22(3):443–473, 2006.

[76] A. M. De Livera, R. J. Hyndman, and R. D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.

[77] A. M. De Livera et al. Automatic forecasting with a modified exponential smoothing state space framework. *Monash Econometrics and Business Statistics Working Papers*, 10(10), 2010.

[78] L. Deng, D. Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[79] G. Ditzler and R. Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2283–2301, 2013.

[80] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4): 12–25, 2015.

[81] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proceedings of the 2002 ACM SIG-*

MOD international conference on Management of data, pages 61–72. ACM, 2002.

[82] P. Domingos and G. Hulten. Mining high-speed data streams. In *Kdd*, volume 2, page 4, 2000.

[83] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[84] P. M. Domingos. A few useful things to know about machine learning. *Commun. acm*, 55(10):78–87, 2012.

[85] D. Dorner, M. Books, and H. Holt. The logic of failure: Why things go wrong and what we can do to make them right. *FACILITATION: AResearch & APPLICATIONS*, page 86, 2001.

[86] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[87] P. Duda, M. Jaworski, and L. Pietruczuk. On pre-processing algorithms for data stream. In *International Conference on Artificial Intelligence and Soft Computing*, pages 56–63. Springer, 2012.

[88] R. Elwell and R. Polikar. Incremental learning of concept drift in non-stationary environments. *IEEE Transactions on Neural Networks*, 22(10): 1517–1531, 2011.

[89] B. Fang, Q. Xu, T. Park, and M. Zhang. Airsense: an intelligent home-based sensing system for indoor air quality analytics. In *Proceedings of the 2016 ACM International joint conference on pervasive and ubiquitous computing*, pages 109–119. ACM, 2016.

[90] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37, 1996.

[91] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, et al. Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88, 1996.

[92] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera. *Learning from imbalanced data sets*. Springer, 2018.

[93] E. Fiesler. Neural network classification and formalization. *Computer Standards & Interfaces*, 16(3):231–239, 1994.

[94] P. Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[95] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng. Spmf: a java open-source pattern mining library. *The Journal of Machine Learning Research*, 15(1):3389–3393, 2014.

[96] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. *AI magazine*, 13(3):57–57, 1992.

[97] M. Freire, Á. Serrano-Laguna, B. M. Iglesias, I. Martínez-Ortiz, P. Moreno-Ger, and B. Fernández-Manjón. Game learning analytics: learning analytics for serious games. *Learning, Design, and Technology: An International Compendium of Theory, Research, Practice, and Policy*, pages 1–29, 2016.

[98] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[99] J. Gabrys, H. Pritchard, and B. Barratt. Just good enough data: Figuring data citizenships through air pollution sensing and data stories. *Big Data & Society*, 3(2):2053951716679677, 2016.

[100] J. Gama. *Knowledge discovery from data streams*. Chapman and Hall/CRC, 2010.

[101] J. Gama. A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence*, 1(1):45–55, 2012.

[102] J. Gama and M. M. Gaber. *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.

[103] J. Gama and P. P. Rodrigues. Data stream processing. In *Learning from Data Streams*, pages 25–39. Springer, 2007.

[104] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[105] J. Gama, P. P. Rodrigues, and R. Sebastião. Evaluating algorithms that learn from data streams. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1496–1500. ACM, 2009.

[106] J. Gama, R. Sebastião, and P. P. Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–338. ACM, 2009.

[107] J. Gama, R. Sebastião, and P. P. Rodrigues. On evaluating stream learning algorithms. *Machine learning*, 90(3):317–346, 2013.

[108] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46 (4):44, 2014.

[109] J. Gao, B. Ding, W. Fan, J. Han, and S. Y. Philip. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, 12(6):37–49, 2008.

[110] S. García, J. Luengo, and F. Herrera. *Data preprocessing in data mining*. Springer, 2015.

[111] M. Garofalakis, J. Gehrke, and R. Rastogi. *Data stream management: processing high-speed data streams*. Springer, 2016.

[112] Y. Geng and C. G. Cassandras. New "smart parking" system based on resource allocation and reservations. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1129–1139, 2013.

[113] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining frequent patterns in data streams at multiple time granularities. *Next generation data mining*, 212:191–212, 2003.

[114] E. R. Girden. *ANOVA: Repeated measures*. Number 84. Sage, 1992.

[115] T. Giuffrè, S. M. Siniscalchi, and G. Tesoriere. A novel architecture of parking management for smart cities. *Procedia-Social and Behavioral Sciences*, 53:16–28, 2012.

[116] M. Goebel and L. Gruenwald. A survey of data mining and knowledge discovery software tools. *ACM SIGKDD explorations newsletter*, 1(1):20–33, 1999.

[117] L. Golab and M. T. Özsu. Issues in data stream management. *ACM Sigmod Record*, 32(2):5–14, 2003.

[118] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, 50(2):23, 2017.

[119] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495, 2017.

[120] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[121] P. S. F. Guido van Rossum. Python: an interpreted, high-level, general-purpose programming language, 1990. online; https://www.python.org; accessed 10 July 2018.

[122] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[123] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000.

[124] J. Han, Y. Chen, G. Dong, J. Pei, B. W. Wah, J. Wang, and Y. D. Cai. Stream cube: An architecture for multi-dimensional analysis of data streams. *Distributed and Parallel Databases*, 18(2):173–197, 2005.

[125] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery*, 15(1): 55–86, 2007.

[126] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[127] D. J. Hand. Principles of data mining. *Drug safety*, 30(7):621–622, 2007.

[128] J. Harri, F. Filali, and C. Bonnet. Mobility models for vehicular ad hoc networks: a survey and taxonomy. *IEEE Communications Surveys & Tutorials*, 11(4):19–41, 2009.

[129] M. Harries and N. S. Wales. Splice-2 comparative evaluation: Electricity pricing. 1999.

[130] P. Harrington. *Machine learning in action*. Manning Publications Co., 2012.

[131] H. Hassani and E. S. Silva. Forecasting with big data: A review. *Annals of Data Science*, 2(1):5–19, 2015.

[132] C. S. Hemalatha, V. Vaidehi, and R. Lakshmi. Minimal infrequent pattern based approach for mining outliers in data streams. *Expert Systems with Applications*, 42(4):1998–2012, 2015.

[133] M. Hermans and B. Schrauwen. Training and analysing deep recurrent neural networks. In *Advances in neural information processing systems*, pages 190–198, 2013.

[134] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining—a general survey and comparison. *ACM sigkdd explorations newsletter*, 2(1):58–64, 2000.

[135] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[136] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[137] T. R. Hoens, R. Polikar, and N. V. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101, 2012.

[138] C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages, office of naval research. *Research memorandum*, 52, 1957.

[139] Y. Hu, B. Feng, X. Zhang, E. Ngai, and M. Liu. Stock trading rule discovery with an evolutionary trend following model. *Expert Systems with Applications*, 42(1):212–222, 2015.

[140] Y.-H. Hu and Y.-L. Chen. Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. *Decision Support Systems*, 42(1):1–24, 2006.

[141] D. Huang, Y. S. Koh, and G. Dobbie. Rare pattern mining on data streams. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 303–314. Springer, 2012.

[142] D. T. J. Huang, Y. S. Koh, G. Dobbie, and R. Pears. Detecting changes in rare patterns from data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 437–448. Springer, 2014.

[143] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM, 2001.

[144] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018. Online; `https://otexts.com/fpp2/`; accessed 15 October 2018.

[145] R. Ihaka and R. Gentleman. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996.

[146] M. Jafari, A. Gauchia, K. Zhang, and L. Gauchia. Simulation and analysis of the effect of real-world driving styles in an ev battery performance and aging. *IEEE Transactions on Transportation Electrification*, 1(4):391–401, 2015.

[147] N. Japkowicz and M. Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[148] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami. An information framework for creating a smart city through internet of things. *IEEE Internet of Things journal*, 1(2):112–121, 2014.

[149] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350, 2015.

[150] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money. Big data: Issues and challenges moving forward. In *2013 46th Hawaii International Conference on System Sciences*, pages 995–1004. IEEE, 2013.

[151] J. M. Kanter and K. Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2015.

[152] K. K. Khedo, R. Perseedoss, A. Mungur, et al. A wireless sensor network air pollution monitoring system. *arXiv preprint arXiv:1005.1737*, 2010.

[153] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment, 2004.

[154] G.-H. Kim, S. Trimi, and J.-H. Chung. Big-data applications in the government sector. *Communications of the ACM*, 57(3):78–85, 2014.

[155] R. U. Kiran and M. Kitsuregawa. Mining correlated patterns with multiple minimum all-confidence thresholds. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 295–306. Springer, 2013.

[156] R. U. Kiran and P. K. Reddy. Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. In *Proceedings of the 14th international conference on extending database technology*, pages 11–20. ACM, 2011.

[157] C. D. Kirkpatrick II and J. A. Dahlquist. *Technical analysis: the complete resource for financial market technicians*. FT press, 2010.

[158] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[159] R. Klinkenberg. Meta-learning, model selection, and example selection in machine learning domains with concept drift. In *LWA*, volume 2005, pages 164–171, 2005.

[160] M. M. Kokar and K. Baclawski. Modeling combined time-and event-driven dynamic systems. In *Proc. 9th OOPSLA Workshop on Behavioral Semantics*, pages 112–129. Citeseer, 2000.

[161] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456. ACM, 2009.

[162] S. Kostadinov. Understanding gru networks, December 2016. online; https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be ; accessed 10 April 2019.

[163] I. Koychev. Tracking changing user interests through prior-learning of context. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 223–232. Springer, 2002.

[164] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. Recent development and applications of sumo-simulation of urban mobility. *International Journal On Advances in Systems and Measurements*, 5(3&4), 2012.

[165] B. Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[166] B. Krawczyk and M. Woźniak. One-class classifiers with incremental learning and forgetting for data streams with concept drift. *Soft Computing*, 19 (12):3387–3400, 2015.

[167] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[168] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, et al. Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, 16(1): 1–10, 2014.

[169] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi. Time series forecasting using a deep belief network with restricted boltzmann machines. *Neurocomputing*, 137:47–56, 2014.

[170] K. P. Lakshmi and C. Reddy. Efficient classifier generation over stream sliding window using associative classification approach. *International Journal of Computer Applications*, 115(22), 2015.

[171] F. Landriscina. *Simulation and learning*. Springer, 2013.

[172] P. Langley, W. Iba, K. Thompson, et al. An analysis of bayesian classifiers. In *Aaai*, volume 90, pages 223–228, 1992.

[173] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl. Time-series extreme event forecasting with neural networks at uber. In *International Conference on Machine Learning*, number 34, pages 1–5, 2017.

[174] D. T. Larose and C. D. Larose. *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[175] M. M. Lazarescu, S. Venkatesh, and H. H. Bui. Using multiple windows to track concept drift. *Intelligent data analysis*, 8(1):29–59, 2004.

[176] C. K.-S. Leung and B. Hao. Mining of frequent itemsets from streams of uncertain data. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 1663–1670. IEEE, 2009.

[177] H.-F. Li and S.-Y. Lee. Mining frequent itemsets over data streams using efficient window sliding techniques. *Expert systems with applications*, 36(2): 1466–1477, 2009.

[178] R. N. Lichtenwalter and N. V. Chawla. Learning to classify data streams with imbalanced class distributions. *New Frontiers in Applied Data Mining. LNCS. Springer, Heidelberg*, 2009.

[179] B. Liu, W. Hsu, Y. Ma, et al. Mining association rules with multiple minimum supports. In *KDD*, volume 99, pages 337–341, 1999.

[180] J. N. Liu, Y. Hu, J. J. You, and P. W. Chan. Deep neural network based feature representation for weather forecasting. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer, 2014.

[181] L.-M. Liu. Identification of seasonal arima models using a filtering method. *Communications in Statistics-Theory and Methods*, 18(6):2279–2288, 1989.

[182] Z. Liu, D. Rexachs, E. Luque, F. Epelde, and E. Cabrera. Simulating the micro-level behavior of emergency department for macro-level features prediction. In *Proceedings of the 2015 winter simulation conference*, pages 171–182. IEEE Press, 2015.

[183] D. J. MacKay and D. J. Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[184] S. Madden. From databases to big data. *IEEE Internet Computing*, 16(3): 4–6, 2012.

[185] T. Mahapatra, I. Gerostathopoulos, and C. Prehofer. Towards integration of big data analytics in internet of things mashup tools. In *Proceedings of the Seventh International Workshop on the Web of Things*, pages 11–16. ACM, 2016.

[186] G. Manogaran, V. Vijayakumar, R. Varatharajan, P. M. Kumar, R. Sundarasekar, and C.-H. Hsu. Machine learning based big data processing

framework for cancer diagnosis using hidden markov model and gm clustering. *Wireless personal communications*, 102(3):2099–2116, 2018.

[187] S. Mansalis, E. Ntoutsi, N. Pelekis, and Y. Theodoridis. An evaluation of data stream clustering algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 11(4):167–187, 2018.

[188] F.-J. H. Marc'Aurelio Ranzato, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07). IEEE Press*, volume 127, 2007.

[189] A. Margara, J. Urbani, F. Van Harmelen, and H. Bal. Streaming the web: Reasoning over dynamic data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 25:24–44, 2014.

[190] D. L. Mark Beyer. The importance of 'big data': A definition, 21 June 2012. online; https://www.gartner.com/en/documents/2057415; accessed 12 February 2019.

[191] B. Marr. How muuch data we create every day?, 21 May 2018. online; https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read; accessed 15 May 2019.

[192] MathWorks. Simulink: Simulation and model-based design, 2002. online; https://www.mathworks.com/products/simulink.html; accessed 28 February 2016.

[193] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[194] C. J. Merz, D. S. Clair, and W. E. Bond. Semi-supervised adaptive resonance theory (smart2). In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 3, pages 851–856. IEEE, 1992.

[195] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, C.-C. Chang, C.-C. Lin, and M. D. Meyer. Package 'e1071', 2019.

[196] L. L. Minku, A. P. White, and X. Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on knowledge and Data Engineering*, 22(5):730–742, 2009.

[197] B. Mirza, Z. Lin, J. Cao, and X. Lai. Voting based weighted online sequential extreme learning machine for imbalance multi-class classification. In *ISCAS*, pages 565–568, 2015.

[198] B. Mirza, Z. Lin, and N. Liu. Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. *Neurocomputing*, 149:316–329, 2015.

[199] S. Mohamad. *Active learning for data streams.* PhD thesis, Bournemouth University, 2017.

[200] D. C. Montgomery, C. L. Jennings, and M. Kulahci. *Introduction to time series analysis and forecasting.* John Wiley & Sons, 2015.

[201] U. o. M. Montreal Institute for Learning Algorithms (MILA). Theano: a python library, 2007. online; http://deeplearning.net/software/theano; accessed 20 April 2019.

[202] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[203] M. Nielsen. Neural networks and deep learning, October 2018. online; http://neuralnetworksanddeeplearning.com; accessed 20 April 2019.

[204] M. A. Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA:, 2015.

[205] K. Nishida, K. Yamauchi, and T. Omori. Ace: Adaptive classifiers-ensemble system for concept-drifting environments. In *International Workshop on Multiple Classifier Systems*, pages 176–185. Springer, 2005.

[206] K. Nishida, S. Shimada, S. Ishikawa, and K. Yamauchi. Detecting sudden concept drift with knowledge of human behavior. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 3261–3267. IEEE, 2008.

[207] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[208] I. of Transportation Systems/ DLR. Sumo: Simulation of urban mobility, 2001. online; http://sumo.sourceforge.net/; accessed 10 March 2016.

[209] O. Orrie, B. Silva, and G. P. Hancke. A wireless smart parking system. In *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*, pages 004110–004114. IEEE, 2015.

[210] V. E. Owen, D. Ramirez, A. Salmon, and R. Halverson. Capturing learner trajectories in educational games through adage (assessment data aggregator for game environments): a click-stream data framework for assessment of learning in play. In *American Educational Research Association Annual Meeting*, pages 1–7, 2014.

[211] A. K. Palit and D. Popovic. *Computational intelligence in time series forecasting: theory and engineering applications*. Springer Science & Business Media, 2006.

[212] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 344–353. ACM, 2013.

[213] E. Panigati, F. A. Schreiber, and C. Zaniolo. Data streams and data stream management systems and languages. In *Data Management in Pervasive Systems*, pages 93–111. Springer, 2015.

[214] K. Patroumpas and T. Sellis. Window specification over data streams. In *International Conference on Extending Database Technology*, pages 445–464. Springer, 2006.

[215] B. R. Prasad and S. Agarwal. Stream data mining: platforms, algorithms, performance evaluators and research trends. *International Journal of Database Theory and Application*, 9(9):201–218, 2016.

[216] Y. Qin, Q. Z. Sheng, N. J. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos. When things matter: A survey on data-centric internet of things. *Journal of Network and Computer Applications*, 64:137–153, 2016.

[217] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga. Ensemble deep learning for regression and time series forecasting. In *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, pages 1–6. IEEE, 2014.

[218] W. Raghupathi and V. Raghupathi. Big data analytics in healthcare: promise and potential. *Health information science and systems*, 2(1):3, 2014.

[219] T. Rajabioun and P. A. Ioannou. On-street and off-street parking availability prediction using multivariate spatiotemporal models. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2913–2924, 2015.

[220] A. Rajaraman and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[221] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57, 2017.

[222] P. Rashidi and D. J. Cook. Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 39(5):949–959, 2009.

[223] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho. Urban planning and building smart cities based on the internet of things using big data analytics. *Computer Networks*, 101:63–80, 2016.

[224] J. Read, A. Bifet, G. Holmes, and B. Pfahringer. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, 88 (1-2):243–272, 2012.

[225] J. Read, A. Bifet, B. Pfahringer, and G. Holmes. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In *International Symposium on Intelligent Data Analysis*, pages 313–323. Springer, 2012.

[226] I. Rish, J. Hellerstein, and J. Thathachar. An analysis of data characteristics that affect naive bayes performance. *IBM TJ Watson Research Center*, 30, 2001.

[227] P. Romeu, F. Zamora-Martínez, P. Botella-Rocamora, and J. Pardo. Time-series forecasting of indoor temperature using pre-trained deep neural networks. In *International Conference on Artificial Neural Networks*, pages 451–458. Springer, 2013.

[228] Y. Rong, Z. Xu, R. Yan, and X. Ma. Du-parking: Spatio-temporal big data tells you realtime parking availability. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 646–654. ACM, 2018.

[229] F. Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[230] H. Rouanet and D. Lepine. Comparison between treatments in a repeated-measurement design: Anova and multivariate methods. *British Journal of Mathematical and Statistical Psychology*, 23(2):147–163, 1970.

[231] Y. D. Rubinstein, T. Hastie, et al. Discriminative vs informative learning. In *KDD*, volume 5, pages 49–53, 1997.

[232] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[233] I. Samaras, A. Arvanitopoulos, N. Evangeliou, J. Gialelis, and S. Koubias. A fuzzy rule-based and energy-efficient method for estimating the free size of parking places in smart cities by using wireless sensor networks. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8. IEEE, 2014.

[234] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, et al. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61: 217–238, 2014.

[235] J. C. Schlimmer and R. H. Granger. Beyond incremental processing: Tracking concept drift. In *AAAI*, pages 502–507, 1986.

[236] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[237] T. S. Sethi and M. Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77–99, 2017.

[238] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, and V. A. Kamaev. A survey of forecast error measures. *World Applied Sciences Journal*, 24(24):171–176, 2013.

[239] H. Shi, M. Xu, and R. Li. Deep learning for household load forecasting—a novel pooling deep rnn. *IEEE Transactions on Smart Grid*, 9(5):5271–5280, 2018.

[240] R. H. Shumway and D. S. Stoffer. *Time series analysis and its applications: with R examples*. Springer, 2017.

[241] V. J. Shute, M. Ventura, M. Bauer, and D. Zapata-Rivera. Melding the power of serious games and embedded assessment to monitor and foster learning. *Serious games: Mechanisms and effects*, 2:295–321, 2009.

[242] E. Siow, T. Tiropanis, and W. Hall. Analytics for the internet of things: A survey. *ACM Computing Surveys (CSUR)*, 51(4):74, 2018.

[243] E. Spyromitros-Xioufis, M. Spiliopoulou, G. Tsoumakas, and I. Vlahavas. Dealing with concept drift and class imbalance in multi-label stream classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[244] K. O. Stanley. Learning concept drift with a committee of decision trees. *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*, 2003.

[245] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM, 2001.

[246] L. Su, H.-y. Liu, and Z.-H. Song. A new classification algorithm for data stream. *IJ Modern Education and Computer Science*, 4:32–39, 2011.

[247] Y. Sun, Z. Wang, H. Liu, C. Du, and J. Yuan. Online ensemble using adaptive windowing for data streams with concept drift. *International Journal of Distributed Sensor Networks*, 12(5):4218973, 2016.

[248] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[249] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

[250] C. J. Taylor, P. G. McKenna, P. C. Young, A. Chotai, and M. Mackinnon. Macroscopic traffic flow modelling and ramp metering control using matlab/simulink. *Environmental Modelling & Software*, 19(10):975–988, 2004.

[251] J. W. Taylor. Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society*, 54(8):799–805, 2003.

[252] G. B. Team. Tensorflow: An end-to-end open source machine learning platform, 9 November 2015. online; https://www.tensorflow.org; accessed 5 December 2018.

[253] R. C. Team et al. R: A language and environment for statistical computing. 2013.

[254] S. Tisue and U. Wilensky. Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Boston, MA, 2004.

[255] R. Tönjes, P. Barnaghi, M. Ali, A. Mileo, M. Hauswirth, F. Ganz, S. Ganea, B. Kjærgaard, D. Kuemper, S. Nechifor, et al. Real time iot stream processing and large-scale data analytics for smart city applications. In *poster session, European Conference on Networks and Communications*. sn, 2014.

[256] E. Z. Tragos, V. Angelakis, A. Fragkiadakis, D. Gundlegard, C.-S. Nechifor, G. Oikonomou, H. C. Pöhls, and A. Gavras. Enabling reliable and secure iot-based smart city applications. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pages 111–116. IEEE, 2014.

[257] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos. Big data analytics: a survey. *Journal of Big data*, 2(1):21, 2015.

[258] S. Tsang, Y. S. Koh, and G. Dobbie. Rp-tree: rare pattern tree mining. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 277–288. Springer, 2011.

[259] A. Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.

[260] S. Urbanek. rjava: Low-level r to java interface, 2013.

[261] M. Van Heeswijk, Y. Miche, T. Lindh-Knuutila, P. A. Hilbers, T. Honkela, E. Oja, and A. Lendasse. Adaptive ensemble models of extreme learning machines for time series prediction. In *International Conference on Artificial Neural Networks*, pages 305–314. Springer, 2009.

[262] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren. The online

performance estimation framework: heterogeneous ensemble learning for data streams. *Machine Learning*, 107(1):149–176, 2018.

[263] A. Varga. Discrete event simulation system. In *Proc. of the European Simulation Multiconference (ESM'2001)*, pages 1–7, 2001.

[264] R. Vilalta and S. Ma. Predicting rare events in temporal domains. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 474–481. IEEE, 2002.

[265] E. I. Vlahogianni, K. Kepaptsoglou, V. Tsetsos, and M. G. Karlaftis. Exploiting new sensor technologies for real-time parking prediction in urban areas. In *Transportation Research Board 93rd Annual Meeting Compendium of Papers*, pages 14–1673, 2014.

[266] P. Vorburger and A. Bernstein. Entropy-based concept shift detection. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 1113–1118. IEEE, 2006.

[267] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. AcM, 2003.

[268] H.-z. Wang, G.-q. Li, G.-b. Wang, J.-c. Peng, H. Jiang, and Y.-t. Liu. Deep learning based ensemble approach for probabilistic wind power forecasting. *Applied energy*, 188:56–70, 2017.

[269] S. Wang and X. Yao. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1119–1130, 2012.

[270] S. Wang and X. Yao. Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, 62(2):434–443, 2013.

[271] S. Wang, L. L. Minku, D. Ghezzi, D. Caltabiano, P. Tino, and X. Yao. Concept drift detection for online class imbalance learning. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2013.

[272] S. Wang, L. L. Minku, and M. Yao, Xin. A learning framework for online class imbalance learning. In *2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, pages 36–45. IEEE, 2013.

[273] S. Wang, L. L. Minku, and X. Yao. Dealing with multiple classes in online class imbalance learning. In *IJCAI*, pages 2118–2124, 2016.

[274] S. Wang, L. L. Minku, and X. Yao. A systematic study of online class imbalance learning with concept drift. *IEEE transactions on neural networks and learning systems*, (99):1–20, 2018.

[275] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 international joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[276] R. H. Weber and R. Weber. *Internet of things*, volume 12. Springer, 2010.

[277] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4):449–475, 2013.

[278] P. H. Westfall and S. S. Young. *Resampling-based multiple testing: Examples and methods for p-value adjustment*, volume 279. John Wiley & Sons, 1993.

[279] G. Widmer and M. Kubat. Effective learning in dynamic environments by explicit context tracking. In *European Conference on Machine Learning*, pages 227–243. Springer, 1993.

[280] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[281] E. Wilder-James. What is big data?an introduction to the big data landscape., 11 January 2012. online; https://www.oreilly.com/ideas/what-is-big-data; accessed 20 April 2019.

[282] U. Wilensky. Netlogo: a multi-agent programmable modeling environment, 1999. online; https://ccl.northwestern.edu/netlogo/; accessed 20 March 2016.

[283] U. Wilensky and W. Rand. *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo.* MIT Press, 2015.

[284] P. R. Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[285] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques 2nd edition.* Morgan Kaufmann, San Francisco, 2005.

[286] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2016.

[287] F. Xia, L. T. Yang, L. Wang, and A. Vinel. Internet of things. *International Journal of Communication Systems*, 25(9):1101–1102, 2012.

[288] Y. Yang, X. Wu, and X. Zhu. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. *Data mining and knowledge discovery*, 13(3):261–289, 2006.

[289] S.-i. Yoshida, K. Hatano, E. Takimoto, and M. Takeda. Adaptive online prediction using weighted windows. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1917–1923, 2011.

[290] W. Zhang and J. Wang. A hybrid learning framework for imbalanced stream classification. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 480–487. IEEE, 2017.

[291] Y. Zhao and K. Takano. An artificial neural network approach for broadband seismic phase picking. *Bulletin of the Seismological Society of America*, 89 (3):670–680, 1999.

[292] J. Zheng, C. Xu, Z. Zhang, and X. Li. Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2017.

[293] Y. Zheng, S. Rajasegarar, C. Leckie, and M. Palaniswami. Smart car parking: temporal clustering and anomaly detection in urban car parking. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6. IEEE, 2014.

[294] Y. Zheng, S. Rajasegarar, and C. Leckie. Parking availability prediction for sensor-enabled car parks in smart cities. In *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6. IEEE, 2015.

[295] I. Žliobaitė. Combining time and space similarity for small size learning under concept drift. In *International Symposium on Methodologies for Intelligent Systems*, pages 412–421. Springer, 2009.

[296] I. Žliobaitė, M. Pechenizkiy, and J. Gama. An overview of concept drift applications. In *Big data analysis: new algorithms for a new society*, pages 91–114. Springer, 2016.

Appendix A

Snapshots of the usage report for the Multi decked Car Park in the summer term of 2016

This appendix provides snapshots, shown in Figure A.1 and Figure A.2, of the report concerning the daily usage of the Multi-deck Car Park over three months period in the summer term of 2016, from 08/03/2016 until 17/06/2016.

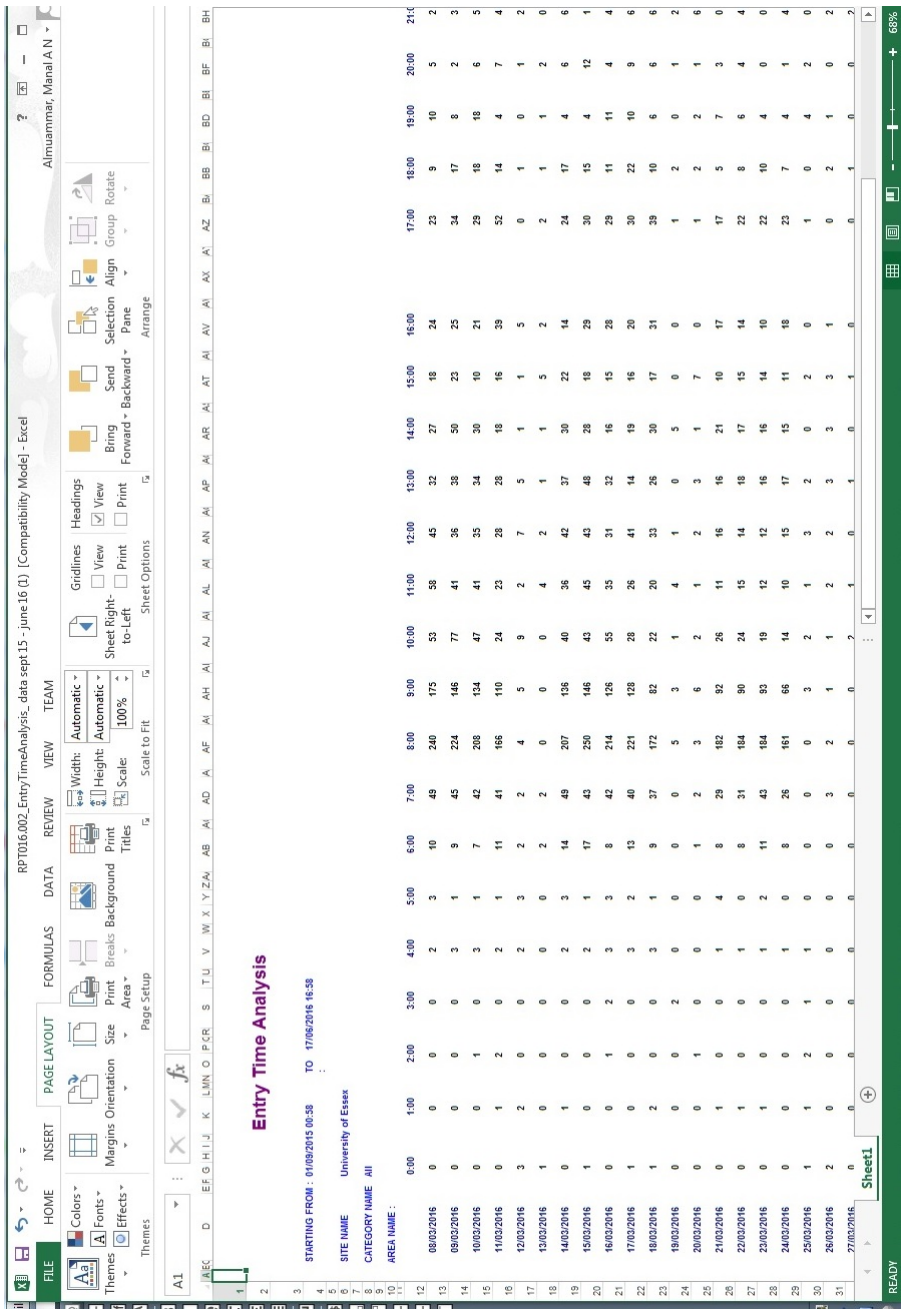
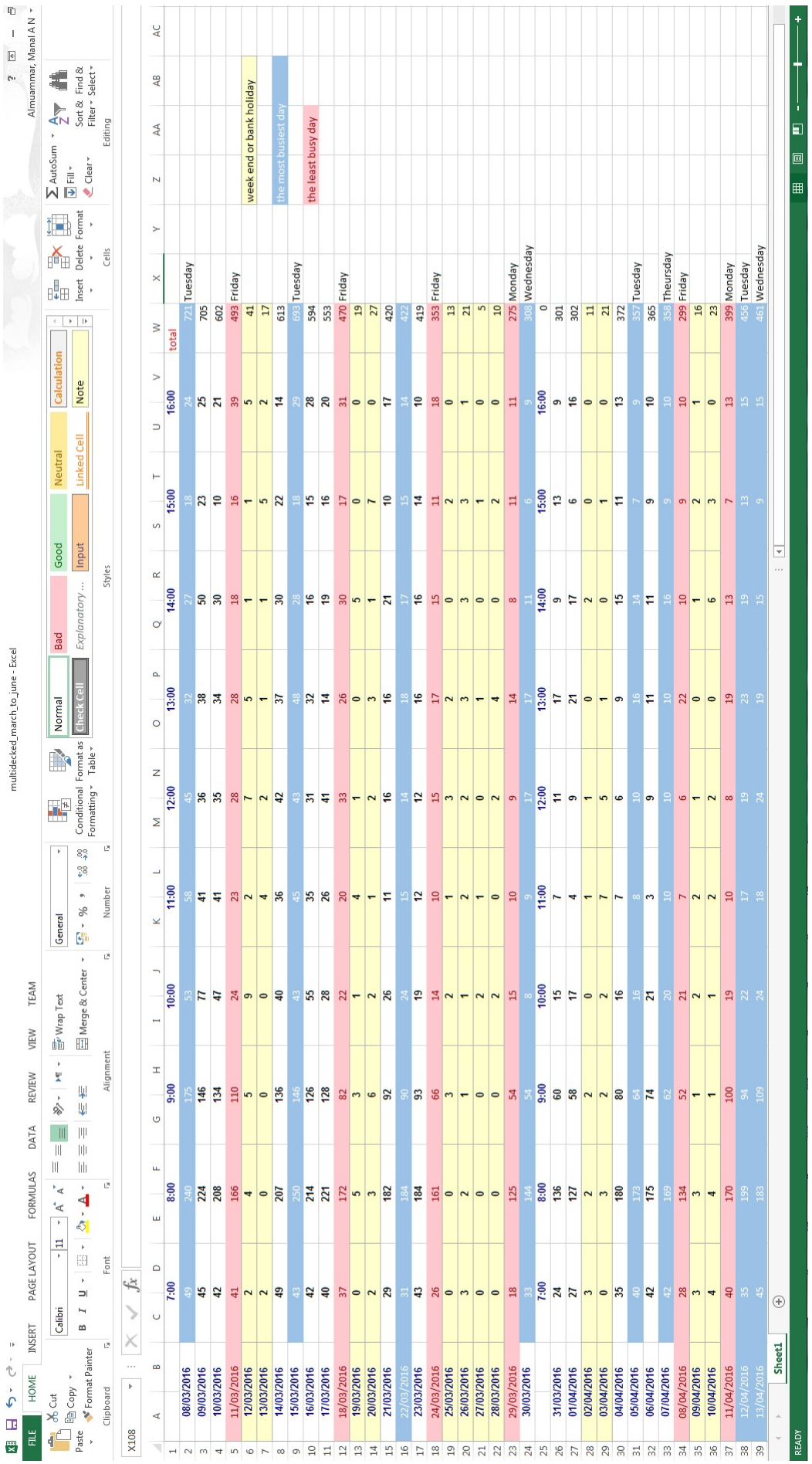


Figure A.1: Snapshots of the usage report for the Multi decked Car Park in the summer term of 2016 (1)

Appendix A. Snapshots of the usage report for the Multi decked Car Park in the summer term of 2016



	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	total		
08/03/2016	49	240	175	53	58	45	32	27	18	24	721	Tuesday	
09/03/2016	45	224	146	77	41	36	38	50	23	25	705		
10/03/2016	42	208	134	47	41	35	34	30	10	21	602		
11/03/2016	41	166	110	24	23	28	28	18	16	39	493	Friday	
12/03/2016	2	4	5	9	2	7	5	1	1	5	41		week end or bank holiday
13/03/2016	2	0	0	0	4	2	1	1	5	2	17		
14/03/2016	49	207	136	40	36	42	37	30	22	14	613		the most busiest day
15/03/2016	43	250	146	43	45	43	48	28	18	29	693	Tuesday	
16/03/2016	42	214	126	55	35	31	32	16	15	28	594		the least busy day
17/03/2016	40	221	128	28	26	41	14	19	16	20	553		
18/03/2016	37	172	82	22	20	33	26	30	17	31	470	Friday	
19/03/2016	0	5	3	1	4	1	0	5	0	0	19		
20/03/2016	2	3	6	2	1	2	3	1	7	0	27		
21/03/2016	29	182	92	26	11	16	16	21	10	17	420		
22/03/2016	31	184	90	24	15	14	18	17	15	14	422		
23/03/2016	43	184	93	19	12	12	16	16	14	10	419		
24/03/2016	26	161	66	14	10	15	17	15	11	18	353	Friday	
25/03/2016	0	0	3	2	1	3	2	0	2	0	13		
26/03/2016	3	2	1	1	2	2	3	3	3	1	21		
27/03/2016	0	0	0	2	1	0	1	0	1	0	5		
28/03/2016	0	0	0	2	0	2	4	0	2	0	10		
29/03/2016	18	125	54	15	10	9	14	8	11	11	275	Monday	
30/03/2016	33	144	54	8	9	17	17	11	6	9	308	Wednesday	
	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	0		
31/03/2016	24	136	60	15	7	11	17	9	13	9	301		
01/04/2016	27	127	58	17	4	9	21	17	6	16	302		
02/04/2016	3	2	2	0	1	1	0	2	0	0	11		
03/04/2016	0	3	2	2	7	5	1	0	1	0	21		
04/04/2016	35	180	80	16	7	6	9	15	11	13	372		
05/04/2016	40	173	64	16	8	10	16	14	7	9	357	Tuesday	
06/04/2016	42	175	74	21	3	9	11	11	9	10	365		
07/04/2016	42	169	62	20	10	10	10	16	9	10	358	Theursday	
08/04/2016	28	134	52	21	7	6	22	10	9	10	299	Friday	
09/04/2016	3	3	1	2	2	1	0	1	2	1	16		
10/04/2016	4	4	1	1	2	2	0	6	3	0	23		
11/04/2016	40	170	100	19	10	8	19	13	7	13	399	Monday	
12/04/2016	35	199	94	22	17	19	23	19	13	15	456	Tuesday	
13/04/2016	45	183	109	24	18	24	19	15	9	15	461	Wednesday	

Figure A.2: Snapshots of the usage report for the Multi decked Car Park in the summer term of 2016 (2)

Appendix B

The full transcripts of the Interviews with the traffic officers at University of Essex

This appendix presents the full transcripts of the interviews with two of the traffic officers at the University of Essex, Mr. Gary Gibbons and Mr. Joe Preston.

- **Car park A** is designed for student use

    1. What is the peak hour /hours during term time?

       Gary: it becomes full very quickly, about 9, and then the number of cars declines after 12

       Joe: most of students here not are living in campus. This car park is usually busy between 10 to 4. And usually it is full between 10-2.

    2. Is the peak hour the same for all weekdays?

       Gary: maybe Tuesday is busier.

       Joe: usually Monday morning and Friday afternoon are less crowded. Wednesdays are busier than other days.

    3. Does the use of the car park decline over the term? And what are the differences?

Gary: usually in the beginning of term the car parks are busier, the number of cars declines slightly over the term.

Joe: at the end of term the number of students is slightly decline. Also there are differences between terms, fall and spring is much busier than summer, but for example in May in summer term becomes busy (exams)

4. In the summer, what is the peak hour? Does this parking lot reach its full capacity?

   Gary: the peak hour is around 11. No, A car park is very empty and like these days there are works in this park lot.

   Joe: it depends on summer schools, but it does not reaches full capacity. Like this summer most of spaces closed because there are some works.

5. Are the drivers parking in Car Park A are permit holders or not? What is the (approximate) percentage of non-registered cars that park there per day?

   Gary: most of drivers are permit holders, and the percentage of non-registered cars is around 10%

   Joe: most of them are permit holders they around 90% and most of them are students. Also the non-registered drivers are from students who live in campus.

6. What is the average time of parking (approximately) in this parking lot in term time?

   Gary: for staff, mostly all day "6 -7 hours" where for students the average time of parking is around 4 hours.

   Joe: about 4 hours.

7. What is the average time of parking (approximately) in this parking lot in the summer?

Gary: in the summer, less number of cars are parked there but they stay for longer time.

Joe: yes, from June to September it is mostly empty.

8. Is there any noticeable difference between fall, spring and summer term?

   Gary: maybe we could say that the spring is the busiest one.

   Joe: the fall is the busiest and the summer term is more quitter.

9. Does the use of the car park by students decline over the teaching terms?

   Gary: yes, the end of term is usually less crowded.

   Joe: yes, they decline over the term.

- **Valley Car Park** is designed for student, staff and visitors.

1. What is the peak hour /hours during term time?

   Gary: it becomes full quicker, especially staff lines in the bottom, so the peak hour is around 8.

   Joe: staff filled up earlier by 8:30 (around 40 spaces), where the car park filled by 9:30 to 10 , usually student park for 3-4 hours.

2. Is the peak hour the same for all weekdays?

   Gary: usually yes it is the same, but Tuesday is the busiest. **Joe**: yes but Tuesday, Wednesday and Thursday are much busiest.

3. Does the use of the car park by the three group of users (staff/students/visitors) decline over the term? And what are the differences?

   Gary: for the staff it is the same, where for student it declines over the term. Regarding the visitors they usually more at the beginning and end of term.

   Joe : it is constant for the staff, but for students it slightly declines. And there is no marked differences for visitors.

4. In the summer, what is the peak hour? Does this parking lot reach its full capacity?

   Gary: it is little bit later, around 9-10. No it doesn't reach its full capacity.

   Joe: it is between 10-11 , and it doesn't reach its full capacity.

5. What is the percentage of visitors' cars that park there per day (in case of term day, summer day)?

   Gary: 10% in the term time. There are more visitors in the summer.

   Joe : there are a row for visitors' spaces and the percentages are usually low, it is around 30% of visitors' spaces, depending on what happen at university (short courses, visitors for bank or library or for consultation)

6. What is the average time of parking (approximately) in this parking lot in term time?

   Gary: it is similar to other parking lots, for students from 3-4 hour, for staff all day around 6-7 hours and for visitors around 2 hours.

   Joe: it is around 4 hours and for staff it is longer.

7. What is the average time of parking (approximately) in this parking lot in the summer?

   Gary: in the summer less number of students parking there and little bit more visitors, but they there is no change in the average time of parking.

   Joe: it is less than term time.

- **Constable Building Car Park** is designed for staff and WH /Edge use.

1. Is any student allowed to park there? Or the parking lot is only allocated to WH/Edge students?

   Gary: No, it is allocated to WH/Edge only.

Joe: the whole spaces are allocated for WH/Edge use.

2. Is any permit holder driver (staff or student) allowed to park there in the morning?

   Gary: No.

   Joe: No

3. What is the peak hour /hours during term time?

   Gary: The peak hour is earlier around 8, it is small and reach full capacity quickly.

   Joe: it depends how busy is the hotel but usually 9-10.

4. Is the peak hour the same for all weekdays?

   Gary: yes it is the same for all weekdays.

   Joe: maybe it less on Monday since it becomes much busier at weekend.

5. Does the use of the car park by the two group of users decline over the term? And what are the differences?

   Gary: no they are mostly staff members.

   Joe: they are just staff , so there is no much differences.

6. In the summer, what is the peak hour? Does this parking lot reach its full capacity?

   Gary: no difference, and it reaches its full capacity.

   Joe: Between 9-10. And it almost full.

7. What is the average time of parking (approximately) in this parking lot in term time?

   Gary: it is around 6 hours.

   Joe : it depends.

8. What is the average time of parking (approximately) in this parking lot in the summer?

Gary: the same.

Joe : it depends.

- **Car Park B** is designed for a mix of students and staff.

  1. What is the peak hour /hours during term time?

     Gary: the peak hour is early around 8, because it is near the sport center.

     Joe Perston: it is around 10 .

  2. Does the use of the car park by the two group of users (staff/students) decline over the term? And what are the differences?

     Gary: there are less students at the end of term, but there is no differences regarding staff members.

     Joe: the staff is constant but for students they declines by half.

  3. Is the peak hour the same for all weekdays?

     Gary: yes, it is the same.

     Joe: it is the same but Monday morning and Friday afternoon is quitter.

  4. In the summer, what is the peak hour? Does this parking lot reach its full capacity?

     Gary: no differences and it is pretty full of staff members, visitors and club members.

     Joe: around 10. It almost full but it doesn't its full capacity.

  5. Are the drivers parking in Car Park a permit holders or not? What is the percentage of non-registered cars that park there per day?

     Gary: they are both, the percentage of non-registered cars is around 25%, and the percentage of staff who parking there is around 70%.

     Joe: the majority of them are permit holders around 80%

6. What is the average time of parking (approximately) in this parking lot in term time?

   Gary: it is around 3-4 hours and for staff it is longer around 6-7 hours.

   Joe: it is around 3-4 hours.

7. What is the average time of parking (approximately) in this parking lot in the summer?

   Gary: for the staff it is the same around 6-7 hours, for the sport center members, it is around 2 hours where for student it becomes less in the summer.

   Joe: it is also 3-4 hours , and in the summer it is busier than car park A.

- **North Car Park** is designed for mix student and staff.

1. How many spaces are allocated to the Nursery drop off? (The Nursery has 3 intervals 8-12, 12-2, 2-6 and know it becomes 2 intervals 8-1, 1-6)?

   Gary: I do not know exactly.

   Joe: around 15

2. How many cars on average use the Nursery park spaces?

   Gary: I do not know exactly.

   Joe: maybe 20-30

3. What is the peak hour /hours during term time?

   Gary: they are mostly staff members, the peak hour is 8.

   Joe: around 9 , mostly staff.

4. Is the peak hour the same for all weekdays?

   Gary: yes it is usually the same.

   Joe: yes but Friday is quitter.

5. Does the use of the car park by the two group of users (staff/students) decline over the term? And what are the differences?

   Gary: 80% of cars there are staff members, so there is no differences over the term but there are less students at the end of term.

   Joe: it slightly declines at the end of summer term time.

6. In the summer, what is the peak hour? Does this parking lot reach its full capacity?

   Gary: no it is same as term time, and it reaches full capacity.

   Joe: around 9, and it nearly reaches its full capacity.

7. Are the drivers parking in the North Car Park a permit holders or not? What is the percentage of non-registered cars that park here per day?

   Gary: yes, the percentage on non-registered is around 20%

   Joe: the majority of drivers are permit holders.

8. What is the average time of parking (approximately) in this parking lot in term time?

   Gary: they are mostly staff and the average time of parking is 5-6 hours.

   Joe: between 6-7 hours 'part time staff'

9. What is the average time of parking (approximately) in this parking lot in the summer?

   Gary: again it is the same to term time.

   Joe: it is the same maybe slightly less.

- **For Special Days at university such as visit days or graduations days,**

1. What are the changes in parking lots administration? How can these limited parking lots be sufficient for visitors?

Gary: there is limited spaces so for the special days we open new spaces.

Joe: encourage staff to use the multi-storey park lot, and usually A or valley car park are allocated for visitors.

2. Are students or staff not allowed to use specific parking lots on these days?

   Gary: Yes but it depends, we usually use barrier and car park A and Valley car park for visitors.

   Joe: they advised to avoid car park A and Valley car park.

3. How many special days on average are there during the academic year? (term time and summer time)

   Gary: the major disruption is graduation days and they are 4 days. The total is varies but usually couple of weeks around 14-15 days.

   Joe: between 15-20 days most of them in summer like summer schools or graduation days.

Appendix C

The details of the simulation model of the University of Essex parking lots

This appendix provides the full details of the simulation model of the movements of traffic within the University of Essex parking lots. To simulate the data set, Netlogo 5.3.1 was used. The first step was to download the map of University of Essex from Google Maps API Styled Wizard then convert it using GIMP 2 and a small python code to text format to represent it in NetLogo.

Before discussing the details of the simulation, the simulation graphic user interface (GUI) will be illustrated first. As can be seen in Figure C.1, there are two main buttons, **Start Up**, which initiates the simulation, initializes parameters and draws the university's map, whereas the second button, **Run Simulation**, turns on the simulation, reflects cars movements in the parking lots. In addition, there are a number of sliders and choices which allow us to control the simulation, the Figure C.1 shows these following buttons:

1. Total number of cars This slider defines the average number of cars arriving on campus per day between 7:00 in the morning and 16:00 in the afternoon. We assumed that cars will arrive following a Normal Distribution, where the

Figure C.1: Parmeters setting in the simulation model GUI

peak hours are between 8:00 and 10:00 in the morning. Following Normal Distribution, the percentage of cars in the interval between 7:00 in the morning and 16:00 in the afternoon is 0.65, coupled with the number of registered users which equals 2942, this results in 1912 cars per day. We add a 10 % of total parking spaces to represent visitors and non-registered users which is around 160 cars. Also we add between thirty to forty cars to represent cars that park for a while in the nursery drop off park, these numbers are based on the interview with parking officers. Given these points, we define 2100 as the default value of number of arriving cars per day (a normal term day). At the same time, we can increase or decrease this number through the slider to control the different situations in the simulation.

2. Time, Week number and Day There is a variation in the parking occupancy rates influenced by the day of year (term/the summer, beginning of term/end of term etc.). We define Time parameter, which allows us to control time in the simulation, choosing between:

- Autumn term
- Spring term
- Summer term
- The summer period
- Random time (which represents holidays such Christmas holiday)

The Day parameter determines the day of week (in our model, Tuesdays and Wednesdays are busier). Moreover, the week slider represents the week in the University's calendar. It starts with a default value one, which is the first week in the Autumn term.

3. Entry time and Entry time in minutes We define the car's entry time to campus in both hour and minutes level, to represent the difference in cars' arrival rates over the same day.

4. Duration of the simulation In this model, we consider three levels of running the simulation:

   - Day level simulation, here we can specify the day of week and the week number in University's calendar or just choose the time and the simulation will be for the first week of the chosen time interval (the default time is Autumn term).
   - The term level simulation, just choose any term from the drop down menu box to get the simulation of the interval from 7:00 in the morning until 16:00 in the afternoon of weekdays.
   - The longest simulation, for one year, it will show the simulation from the first week of Autumn term until the end of the summer period for the same interval.

On the whole, inside the simulation we define a number of procedures that first load the parking lots map, initiate the variables and set up the simulation

parameters. In addition, we define the agents — which are cars in our model— and agents' attributes. Particularly, we define the global variables that we have used in the model code, and define the patches (map) and agents (cars) attributes as illustrated in the following Figure C.2.

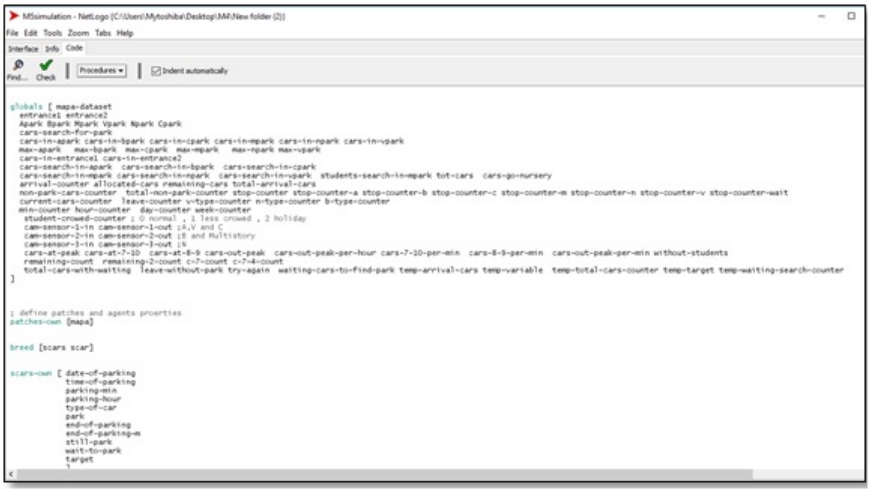


Figure C.2: Snapshots of defining variables, patches and agents codes

Then we define four procedures to load the university map and match it with the patches as in the below Figure C.3.

Figure C.3: Snapshot of loading map procedures

In the **Start Up** procedure, we initialize the model variables, set shapes of agents as cars, defining parking lots location and the entrances on the map and set up four MC-EXCEL files where we will export simulation to. Also set up the term and weeks and match them to each other according to university calendar.

At this point we set up a file for cars entrance information and where they are park, Also there is a couple of files of cam sensors data and payment machine data (which both derived from the simulation data).

Furthermore, as we have pointed already, we define the average number of arriving cars which we can change, but at the same time it should be kept unchanged until the end of the simulation, for example in the week or term level simulation. Hence we define a number of procedures that reflect the variances in the arriving cars during the simulation run , taking into account factors such as the time of day, the day itself and time of term - beginning or the end of term- which influences the number of arrival cars of students especially. (The number of students' cars drops over term time, also Tuesday, Wednesday on average are more crowded than other weekdays to some extent). For simplicity we consider intervals of one hour and define the arrival cars rate per minute at each interval. Inside the simulation too, we implement a number of procedures that reflect the parking patterns and driver behaviors at parking lots. As we calculate the arrival cars per hour and distribute cars per minute, we add some if statements to control the distribution of cars over parking lots. In detail, we define four main procedures to control the number of arrival cars, based on the analysis of parking patterns:

- **Car arrival** which takes the number of arriving cars that we select in the model, initializes the arrivals counters and distributes arriving cars over the entrances.

- **Set arrival cars** that calculates the cars arrival rate at different intervals -one interval lasts one hour-, based on the total number of arrival cars - which are defined as parameter already- coupled with the day of week and time of year. It also considers the hour of day as the arriving cars rate varies throughout the day and among week days.

- **Check students' crowd** which retrieves a variable that measures the

crowd of students' cars in the parking lots based on the time of year. The number of students' cars decrease over term and drops sharply in the summer. For instance, we set the crowd variable to zero in the normal term day (beginning and mid of term). Whereas at the end of term, the number of students decreases especially in (A and Valley park), so we set the crowd variable to one. Furthermore, in the holiday time, the number of students drops sharply (as park A become empty), so we set the crowd variable to two .

- **Locate cars** which takes the variable retrieved from the previous procedure -that indicates the variation of number of students' cars- to calculate the cars arrival rate at different intervals of the day.

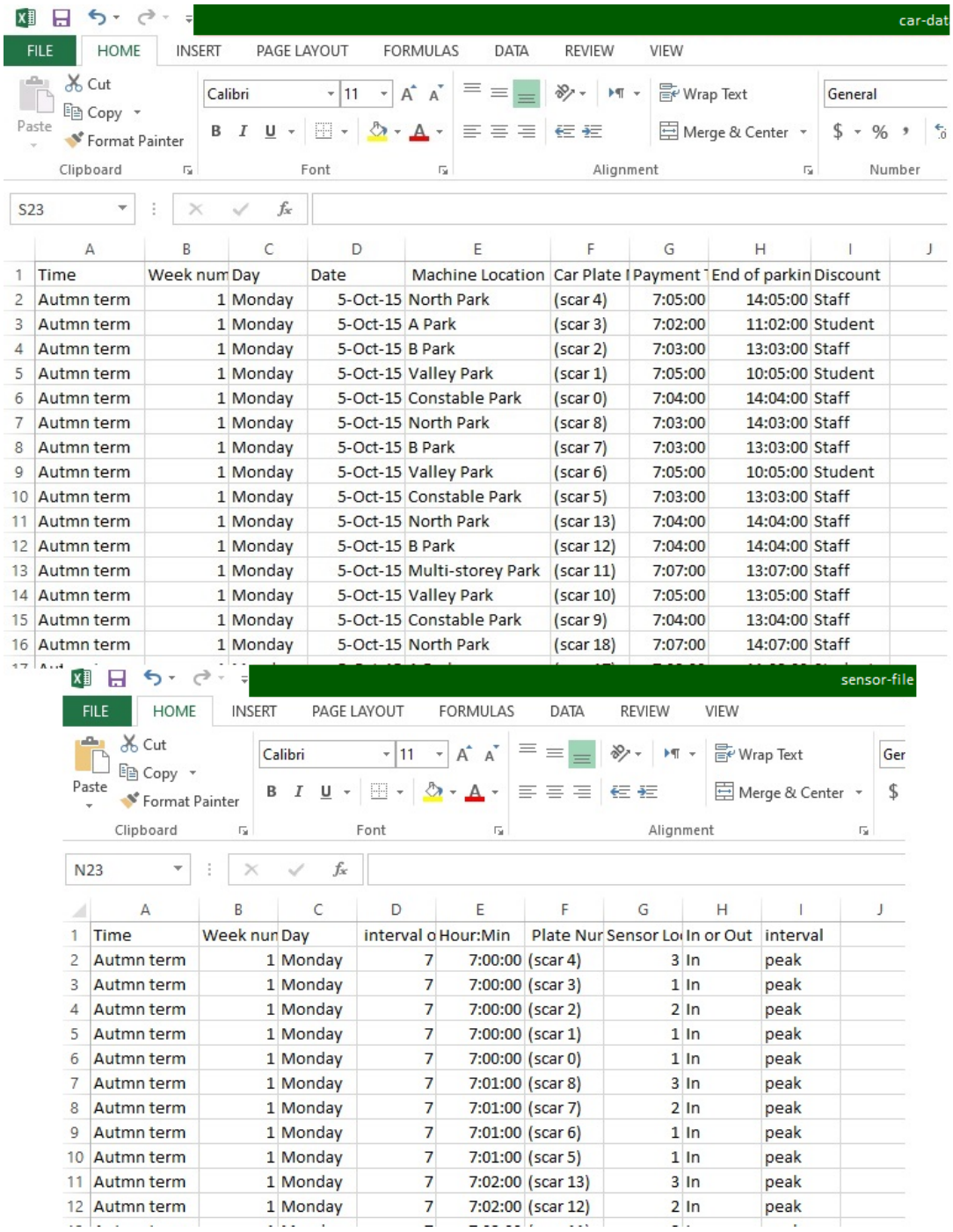
Moreover based on the analysis of drivers' behavior patterns at the University parking lots, we define a number of procedures that distribute cars over parking lots to represent these patterns as close to a reality as possible. In particular, we implement three main procedures which contain many functions as follows:

1. **Distribute searching cars** We use the calculated probabilities together with the number of arriving cars which we have calculated it the procedures in the previous paragraph in order to calculate the number of cars that arrive/search at each park lot per minute.

2. **Do search** This procedure assigns cars to parking lots, based on the above procedures and parking lots capacity and occupancy. At this point we allow cars that search in specific parking lots to search in the nearest permitted parking lot. For example, cars which are assigned to search in Park A could search in Valley Park if Park A is full , but these cars are not allowed to search in Park C -which is allocated for WH Edge and WH employees-. In case a car does not find a space, we keep the target parking lot for the car and allow the car to keep searching for a parking space, while new cars also arrive, for 15 minutes before they leave the car park.

3. **Check which car will leave** Not only the cars arrive at parking lots, but also cars leave the parking lots. We calculate the exit time for each car parked, based on the driver type (staff member, student, visitor, gym member or just use Day nursery park spaces).

For the most important part of the model, we import data to MS-Excel files to create streams in our model. The first file includes all information about cars that park and leave parking lots (type of drivers, car park, entry time and minute, expected leaving time, waiting time to find park). Also there are a couple of files for camera sensors data, and payment information from smart machines. 40% of users pay annually or termly and do not use payment machines, so we assume their payment data comes from the mobile application. The Figure C.4 below shows parts of the generated streams files.



car-dat

	Time	Week num	Day	Date	Machine Location	Car Plate I	Payment	End of parkin	Discount
2	Autmn term	1	Monday	5-Oct-15	North Park	(scar 4)	7:05:00	14:05:00	Staff
3	Autmn term	1	Monday	5-Oct-15	A Park	(scar 3)	7:02:00	11:02:00	Student
4	Autmn term	1	Monday	5-Oct-15	B Park	(scar 2)	7:03:00	13:03:00	Staff
5	Autmn term	1	Monday	5-Oct-15	Valley Park	(scar 1)	7:05:00	10:05:00	Student
6	Autmn term	1	Monday	5-Oct-15	Constable Park	(scar 0)	7:04:00	14:04:00	Staff
7	Autmn term	1	Monday	5-Oct-15	North Park	(scar 8)	7:03:00	14:03:00	Staff
8	Autmn term	1	Monday	5-Oct-15	B Park	(scar 7)	7:03:00	13:03:00	Staff
9	Autmn term	1	Monday	5-Oct-15	Valley Park	(scar 6)	7:05:00	10:05:00	Student
10	Autmn term	1	Monday	5-Oct-15	Constable Park	(scar 5)	7:03:00	13:03:00	Staff
11	Autmn term	1	Monday	5-Oct-15	North Park	(scar 13)	7:04:00	14:04:00	Staff
12	Autmn term	1	Monday	5-Oct-15	B Park	(scar 12)	7:04:00	14:04:00	Staff
13	Autmn term	1	Monday	5-Oct-15	Multi-storey Park	(scar 11)	7:07:00	13:07:00	Staff
14	Autmn term	1	Monday	5-Oct-15	Valley Park	(scar 10)	7:05:00	13:05:00	Staff
15	Autmn term	1	Monday	5-Oct-15	Constable Park	(scar 9)	7:04:00	13:04:00	Staff
16	Autmn term	1	Monday	5-Oct-15	North Park	(scar 18)	7:07:00	14:07:00	Staff

sensor-file

	Time	Week nun	Day	interval o	Hour:Min	Plate Nur	Sensor Lo	In or Out	interval
2	Autmn term	1	Monday	7	7:00:00	(scar 4)	3	In	peak
3	Autmn term	1	Monday	7	7:00:00	(scar 3)	1	In	peak
4	Autmn term	1	Monday	7	7:00:00	(scar 2)	2	In	peak
5	Autmn term	1	Monday	7	7:00:00	(scar 1)	1	In	peak
6	Autmn term	1	Monday	7	7:00:00	(scar 0)	1	In	peak
7	Autmn term	1	Monday	7	7:01:00	(scar 8)	3	In	peak
8	Autmn term	1	Monday	7	7:01:00	(scar 7)	2	In	peak
9	Autmn term	1	Monday	7	7:01:00	(scar 6)	1	In	peak
10	Autmn term	1	Monday	7	7:01:00	(scar 5)	1	In	peak
11	Autmn term	1	Monday	7	7:02:00	(scar 13)	3	In	peak
12	Autmn term	1	Monday	7	7:02:00	(scar 12)	2	In	peak

Figure C.4: Snapshots of the generated streams (MS-Excel files)

Appendix D

The occupancy of the University of Essex parking lots over the Autumn term 2015

This appendix provides diagrams that represent the occupancy patterns of the University of Essex parking lots from the simulated datasets (the first dataset was provided in Chapter 6 and here the remaining 9 datasets).

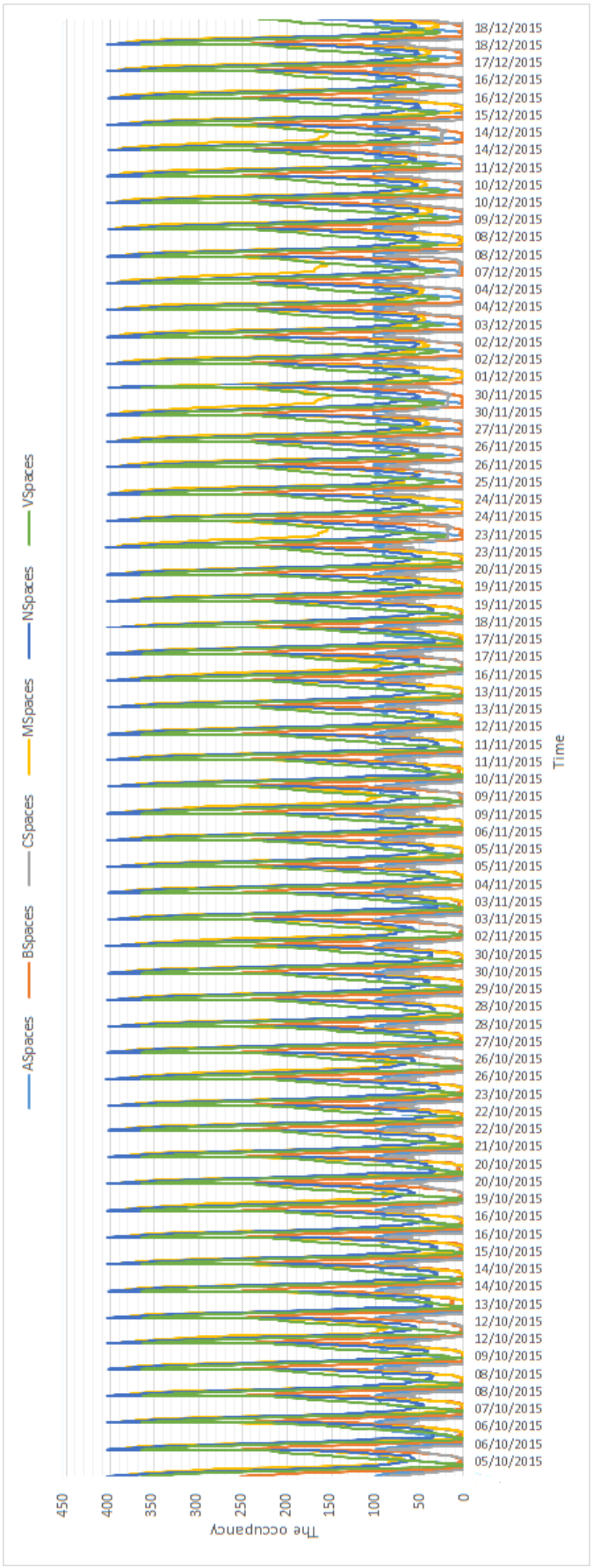


Figure D.1: The occupancy of the parking lots over the Autumn term 2015 (Dataset 2)

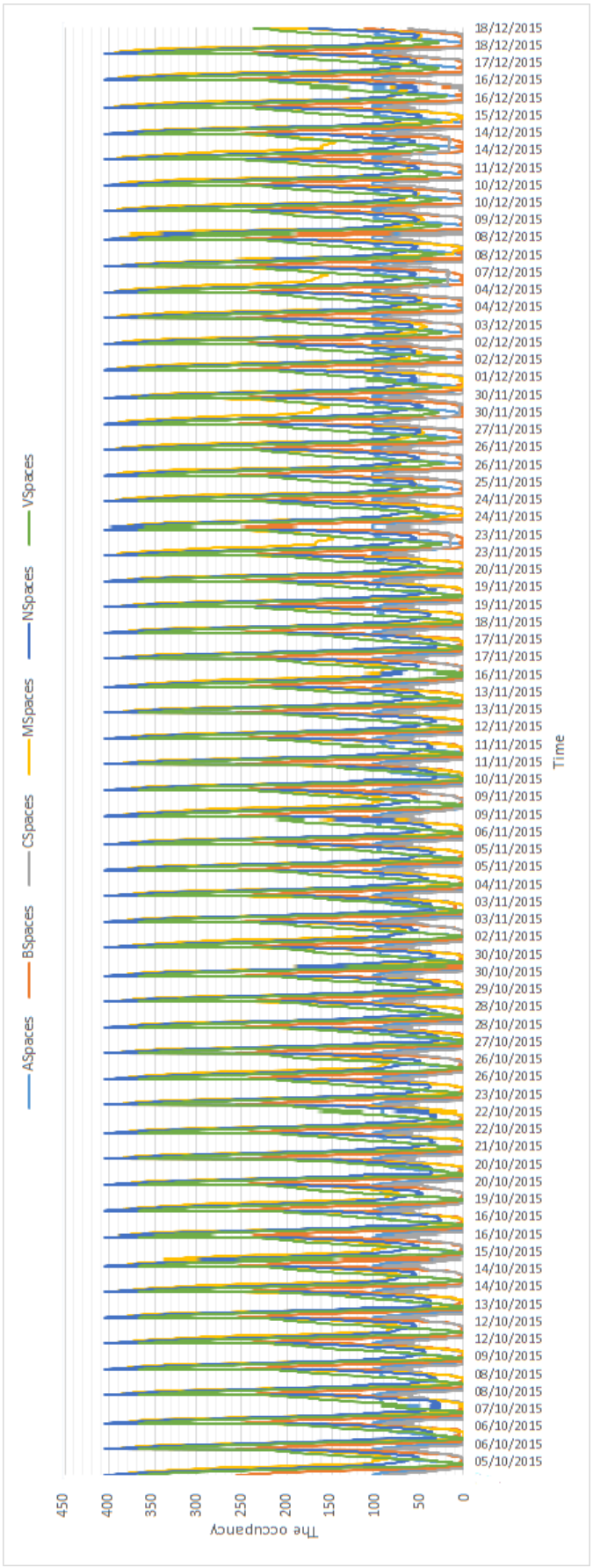


Figure D.2: The occupancy of the parking lots over the Autumn term 2015 (Dataset 3)

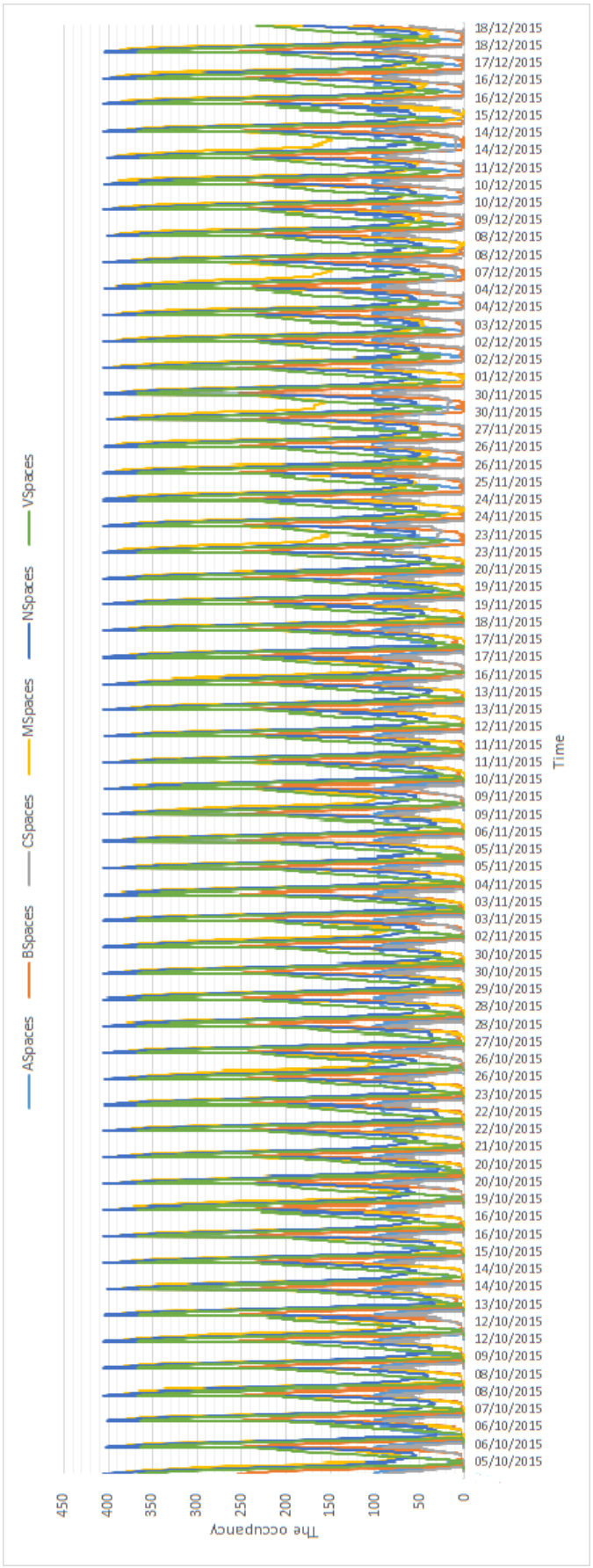


Figure D.3: The occupancy of the parking lots over the Autumn term 2015 (Dataset 4)

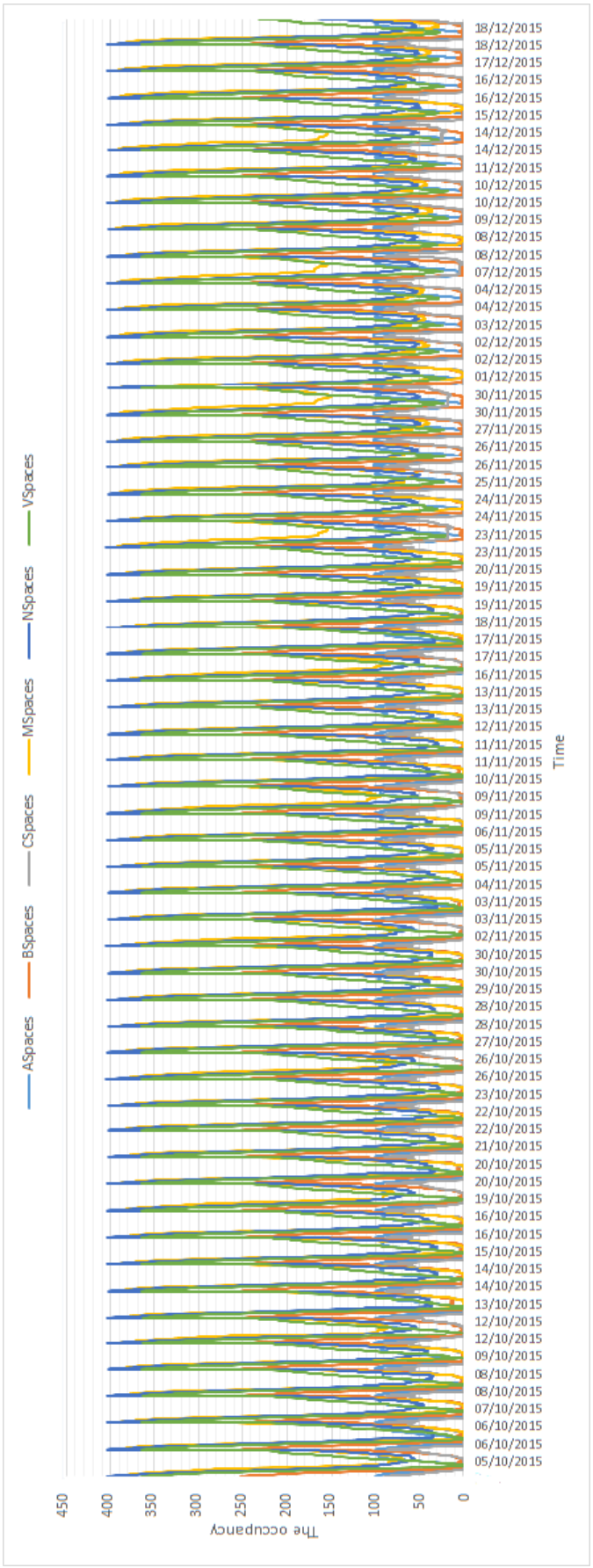


Figure D.4: The occupancy of the parking lots over the Autumn term 2015 (Dataset 5)

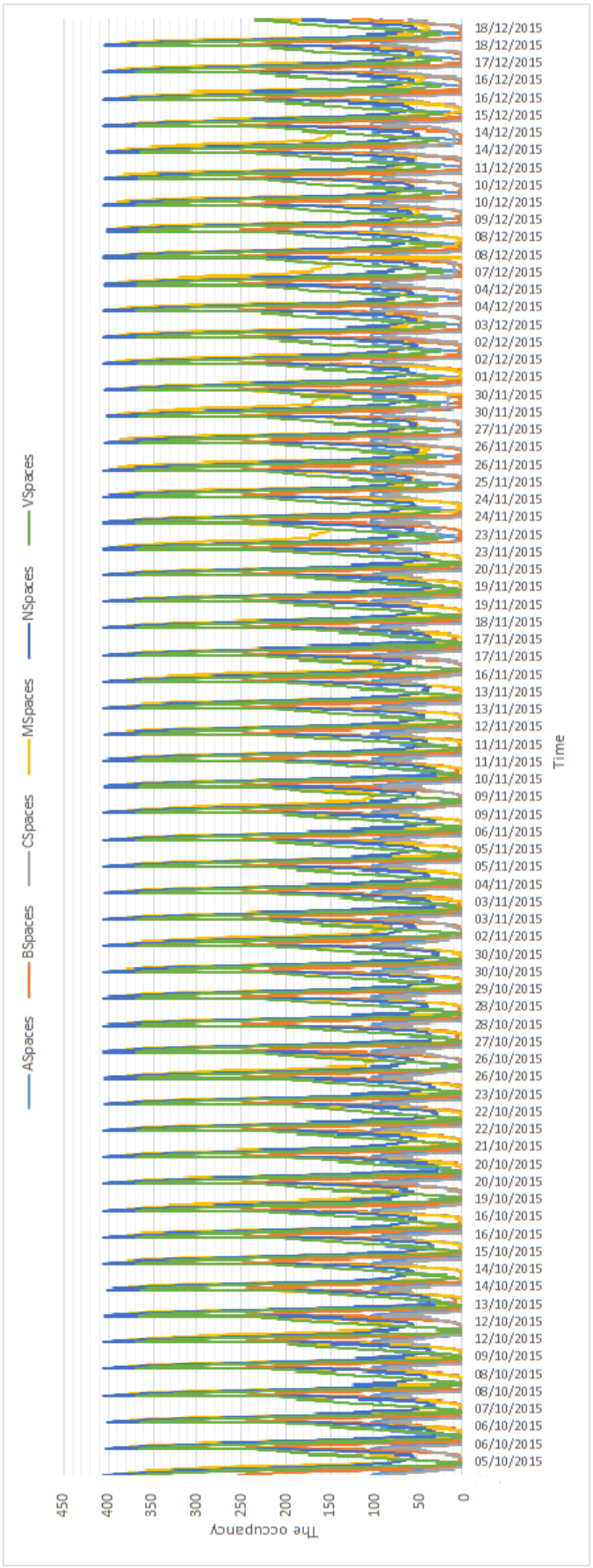


Figure D.5: The occupancy of the parking lots over the Autumn term 2015 (Dataset 6)

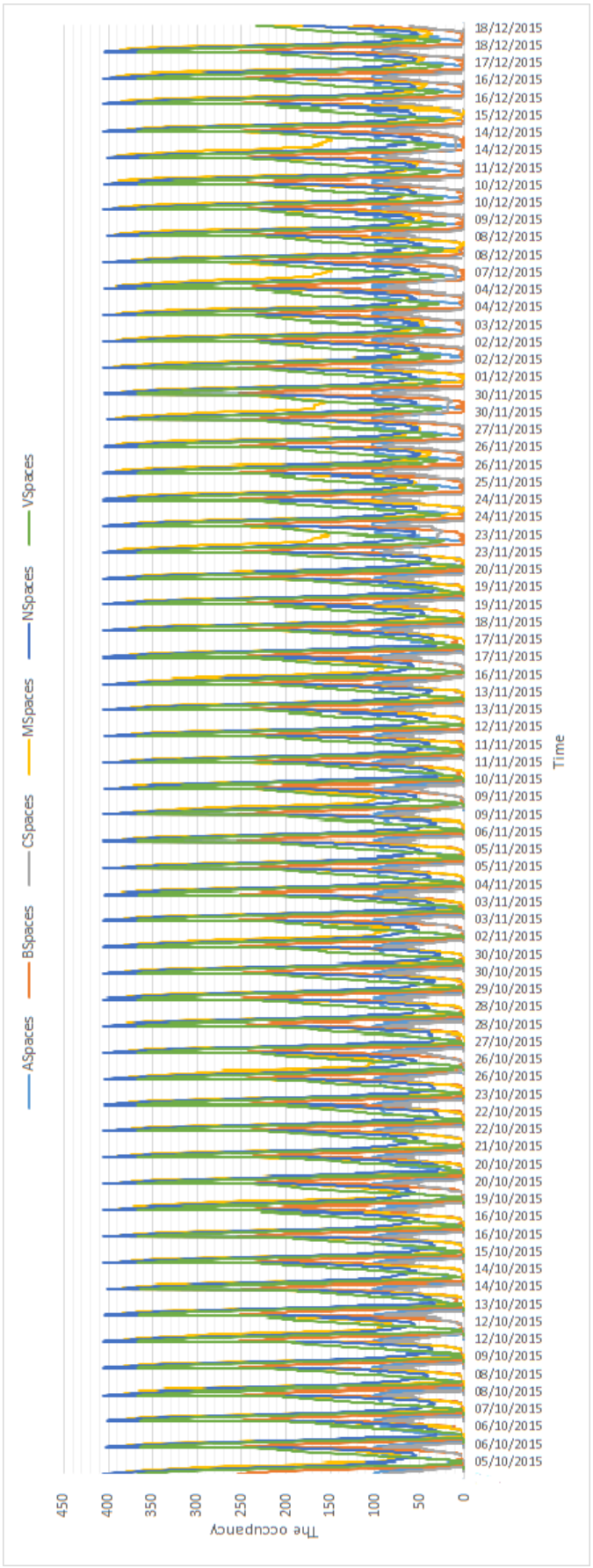


Figure D.6: The occupancy of the parking lots over the Autumn term 2015 (Dataset 7)

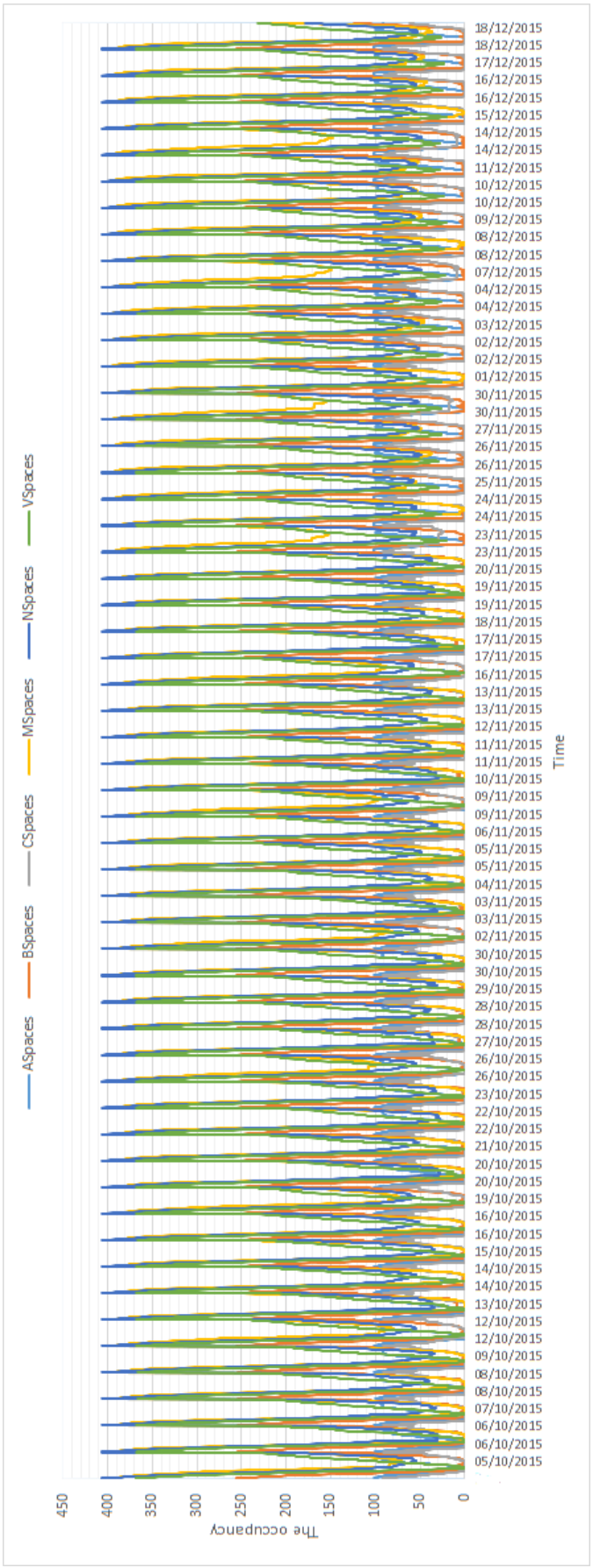


Figure D.7: The occupancy of the parking lots over the Autumn term 2015 (Dataset 8)

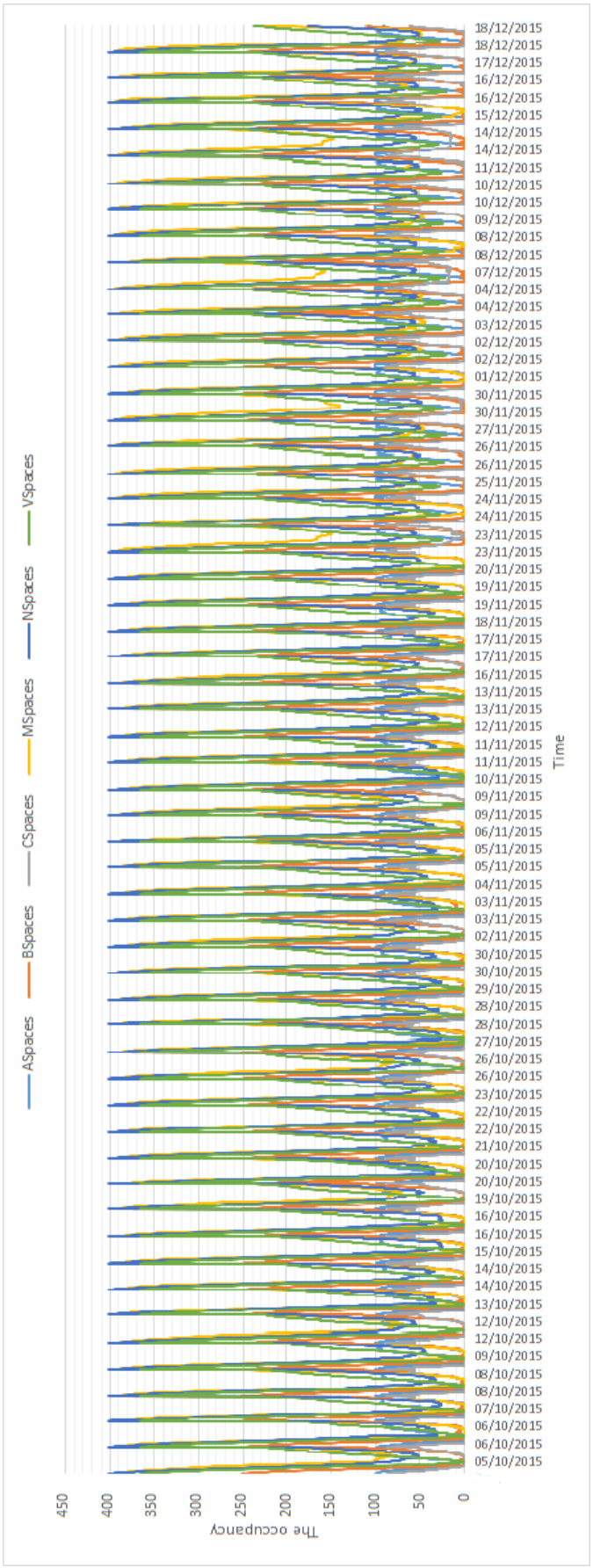


Figure D.8: The occupancy of the parking lots over the Autumn term 2015 (Dataset 9)

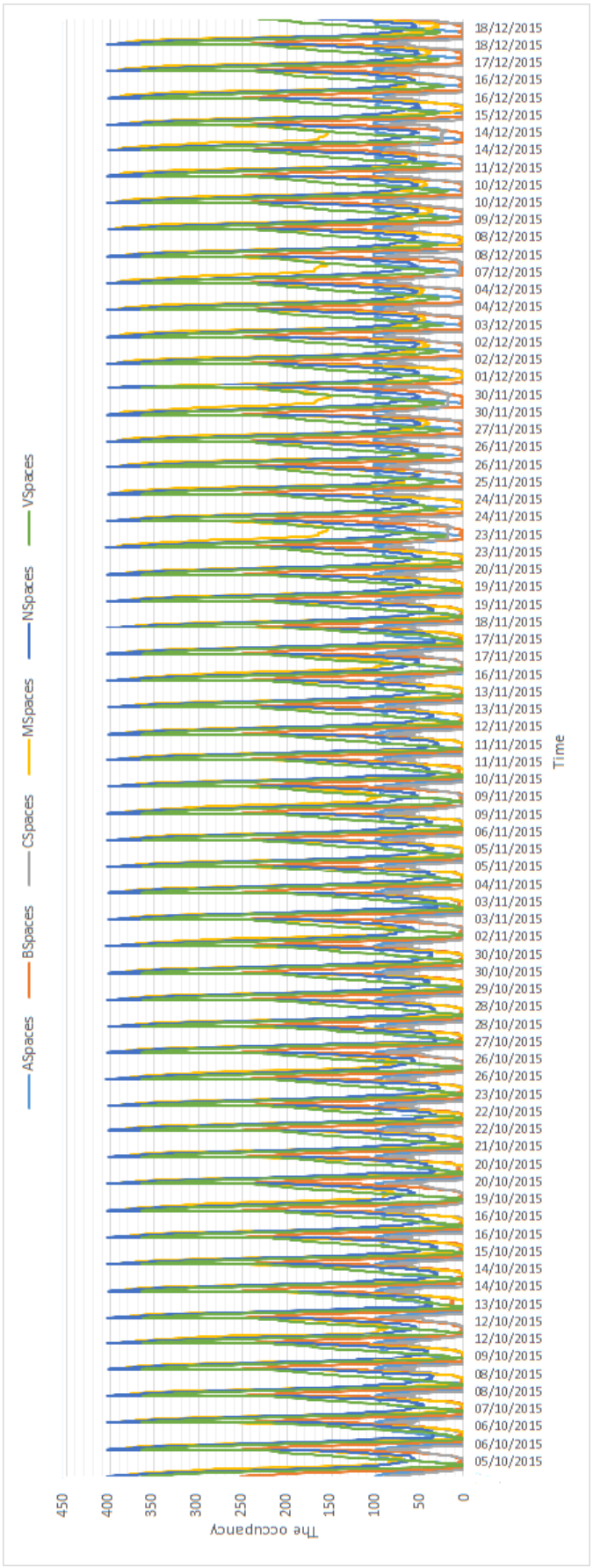


Figure D.9: The occupancy of the parking lots over the Autumn term 2015(Dataset 10)

Appendix E

The performance evaluation of the proposed forecasting models

This appendix provides the details about the individual performance mse values from forecasting models presented in Chapter 6. Furthermore, the results of the pairwise comparison between these GRUs model are presented here.

- Details of the individual MSE values across the different outputs (parking lots) for the GRU models, these measures are reported previously as an aggregated values in Figure 6.11

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 1	Dataset1	0.008691	0.005568	0.020077	0.013452	0.007102	0.005601	0.010081653
	Dataset2	0.008365	0.007412	0.01704	0.014057	0.008886	0.006107	0.010311114
	Dataset3	0.007619	0.004721	0.016118	0.013689	0.007632	0.0057	0.009246581
	Dataset4	0.009025	0.004484	0.016384	0.01246	0.007511	0.004955	0.009136561
	Dataset5	0.010154	0.011564	0.022547	0.018655	0.012919	0.006897	0.013789498
	Dataset6	0.007141	0.005064	0.019314	0.012163	0.006167	0.006165	0.009335604
	Dataset7	0.010612	0.007413	0.022597	0.014049	0.009762	0.005399	0.011638661
	Dataset8	0.010235	0.005452	0.023808	0.01869	0.007765	0.004577	0.011754568
	Dataset9	0.007616	0.006409	0.023172	0.013192	0.008915	0.005299	0.010767385
	Dataset10	0.006597	0.00742	0.021323	0.011761	0.008627	0.005798	0.010254233

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 2	Dataset1	0.007068	0.005194	0.015249	0.011947	0.007129	0.005326	0.008652286
	Dataset2	0.008534	0.006976	0.011727	0.014432	0.00765	0.006725	0.009340648
	Dataset3	0.007302	0.003374	0.014676	0.011711	0.005743	0.004833	0.007939675
	Dataset4	0.007961	0.003629	0.014971	0.012975	0.006088	0.003941	0.008260673
	Dataset5	0.005005	0.004383	0.018939	0.011831	0.005875	0.003954	0.008331094
	Dataset6	0.006038	0.006012	0.01598	0.014819	0.008464	0.004099	0.00923522
	Dataset7	0.018303	0.009207	0.021425	0.017857	0.011143	0.007663	0.01426636
	Dataset8	0.006212	0.004621	0.016769	0.01452	0.007153	0.005318	0.009098724
	Dataset9	0.00716	0.005358	0.018502	0.012985	0.007594	0.006735	0.009722398
	Dataset10	0.006753	0.005421	0.023026	0.010071	0.00698	0.004976	0.009537698

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 3	Dataset1	0.008323	0.007309	0.015883	0.013762	0.007409	0.006156	0.00980709
	Dataset2	0.010082	0.007238	0.018017	0.017533	0.009511	0.006235	0.01143589
	Dataset3	0.01011	0.00744	0.016785	0.015903	0.010578	0.008073	0.01148128
	Dataset4	0.008523	0.006007	0.018648	0.0114	0.006739	0.005833	0.00952495
	Dataset5	0.008871	0.005854	0.019763	0.017977	0.00923	0.004372	0.01101114
	Dataset6	0.009548	0.004983	0.020181	0.014432	0.006684	0.004574	0.010067
	Dataset7	0.009648	0.004424	0.021897	0.015277	0.006961	0.004789	0.01049932
	Dataset8	0.007556	0.005305	0.022958	0.013171	0.006685	0.005074	0.01012487
	Dataset9	0.011834	0.005218	0.025399	0.012892	0.008196	0.008886	0.01207068
	Dataset10	0.013226	0.008665	0.025129	0.015115	0.009601	0.00914	0.01347926

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 4	Dataset1	0.007169	0.004785	0.014003	0.012437	0.00579	0.005227	0.00823535
	Dataset2	0.006945	0.005414	0.014241	0.012806	0.007737	0.005779	0.00882033
	Dataset3	0.010728	0.007352	0.018208	0.012716	0.0097	0.00805	0.01112565
	Dataset4	0.009174	0.004829	0.015393	0.013353	0.007677	0.006144	0.00942819
	Dataset5	0.010917	0.006096	0.019954	0.016706	0.009265	0.004943	0.01131353
	Dataset6	0.008022	0.00509	0.016251	0.015735	0.006889	0.005053	0.00950678
	Dataset7	0.007475	0.004682	0.017466	0.013791	0.006116	0.005031	0.00909358
	Dataset8	0.009405	0.004218	0.019262	0.01326	0.00549	0.004423	0.00934315
	Dataset9	0.008964	0.005098	0.02211	0.010185	0.005687	0.005068	0.00951855
	Dataset10	0.013394	0.00486	0.023435	0.012502	0.007836	0.006474	0.01141669

Figure E.1: the individual MSE values across the different parking lots in GRU models

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 5	Dataset1	0.009405	0.006213	0.019811	0.016275	0.00898	0.007027	0.011285176
	Dataset2	0.008334	0.016016	0.01808	0.024033	0.01653	0.011972	0.015827499
	Dataset3	0.007704	0.007522	0.015383	0.020432	0.011265	0.007834	0.011690239
	Dataset4	0.00829	0.008609	0.015721	0.019695	0.011657	0.007741	0.01195222
	Dataset5	0.007427	0.005916	0.01946	0.020802	0.007739	0.00494	0.011047347
	Dataset6	0.007462	0.007537	0.016445	0.020658	0.010998	0.009803	0.012150609
	Dataset7	0.008658	0.008638	0.017084	0.022039	0.011432	0.008679	0.012754744
	Dataset8	0.007865	0.00981	0.016783	0.024891	0.013298	0.008309	0.013492644
	Dataset9	0.009211	0.00741	0.023565	0.015971	0.009341	0.006461	0.011993006
	Dataset10	0.006295	0.014343	0.022863	0.026433	0.017472	0.010379	0.016297535

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 6	Dataset1	0.005127	0.00825	0.011532	0.015588	0.009878	0.006657	0.009505187
	Dataset2	0.004721	0.006372	0.012204	0.01605	0.009914	0.007063	0.009387378
	Dataset3	0.007121	0.004079	0.01303	0.010481	0.005678	0.004503	0.007481997
	Dataset4	0.013011	0.006832	0.019224	0.018547	0.01008	0.007446	0.012523427
	Dataset5	0.006516	0.003817	0.015675	0.020133	0.011191	0.005344	0.010445878
	Dataset6	0.004675	0.009342	0.014255	0.016596	0.009869	0.005773	0.010084864
	Dataset7	0.008962	0.004243	0.021289	0.01601	0.006812	0.004863	0.010363092
	Dataset8	0.006212	0.004621	0.016769	0.01452	0.007153	0.005318	0.009098724
	Dataset9	0.007586	0.005989	0.022505	0.012312	0.007799	0.0056	0.010298518
	Dataset10	0.008501	0.005297	0.019374	0.012919	0.007155	0.005595	0.009806918

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 7	Dataset1	0.00919	0.006415	0.018443	0.015539	0.010036	0.005413	0.01083949
	Dataset2	0.011071	0.008775	0.016731	0.01998	0.011649	0.008079	0.01271423
	Dataset3	0.010494	0.008504	0.017793	0.019176	0.011269	0.008273	0.01258485
	Dataset4	0.013011	0.006832	0.019224	0.018547	0.01008	0.007446	0.01252343
	Dataset5	0.009534	0.010755	0.019623	0.027005	0.012931	0.006961	0.01446797
	Dataset6	0.005356	0.011966	0.022492	0.019543	0.011306	0.005885	0.012758
	Dataset7	0.008658	0.008638	0.017084	0.022039	0.011432	0.008679	0.01275474
	Dataset8	0.009015	0.010105	0.019286	0.021722	0.013625	0.007704	0.01357627
	Dataset9	0.009939	0.006297	0.023847	0.014995	0.009159	0.005412	0.01160817
	Dataset10	0.010174	0.006709	0.022942	0.016322	0.008117	0.005814	0.01167965

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 8	Dataset1	0.009018	0.006531	0.016841	0.016742	0.010014	0.005644	0.01079826
	Dataset2	0.006385	0.006659	0.012783	0.014975	0.009343	0.007068	0.00953566
	Dataset3	0.008222	0.004532	0.015651	0.016206	0.007066	0.004529	0.0093675
	Dataset4	0.01122	0.004094	0.017717	0.013495	0.006422	0.005177	0.00968768
	Dataset5	0.007097	0.009904	0.017525	0.022777	0.01206	0.007771	0.01285567
	Dataset6	0.006258	0.00686	0.015345	0.01809	0.009646	0.005763	0.010327
	Dataset7	0.009968	0.008083	0.014798	0.021586	0.008874	0.007299	0.01176811
	Dataset8	0.006479	0.005612	0.015651	0.016697	0.008117	0.004242	0.00946628
	Dataset9	0.01386	0.007978	0.025085	0.018672	0.01013	0.006676	0.01373361
	Dataset10	0.0142	0.010431	0.026389	0.019894	0.011539	0.008432	0.01514742

Figure E.1: the individual MSE values across the different parking lots in GRU models (cont)

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 9	Dataset1	0.007415	0.006882	0.017297	0.017358	0.009166	0.006869	0.010831118
	Dataset2	0.007758	0.007357	0.015809	0.018171	0.00983	0.006732	0.01094283
	Dataset3	0.006731	0.006028	0.016457	0.017268	0.008559	0.00543	0.010078769
	Dataset4	0.005347	0.009519	0.017451	0.021335	0.011132	0.007173	0.011992704
	Dataset5	0.006288	0.007574	0.0186	0.018612	0.008142	0.005386	0.010767105
	Dataset6	0.00532	0.009262	0.019544	0.021005	0.010259	0.005769	0.011859869
	Dataset7	0.008367	0.006134	0.020011	0.018143	0.007958	0.006179	0.011132075
	Dataset8	0.00603	0.008428	0.01965	0.022486	0.010215	0.005521	0.012054869
	Dataset9	0.007158	0.009537	0.021393	0.021741	0.011321	0.00771	0.013143279
	Dataset10	0.007059	0.00867	0.021278	0.019668	0.010601	0.007562	0.012473011

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model10	Dataset1	0.004888	0.005901	0.015347	0.016179	0.009467	0.004978	0.00945996
	Dataset2	0.015674	0.011555	0.027391	0.022424	0.011283	0.006197	0.01575413
	Dataset3	0.007478	0.004342	0.015744	0.015143	0.005985	0.004277	0.008828324
	Dataset4	0.006202	0.004014	0.014685	0.012742	0.006207	0.005841	0.008281852
	Dataset5	0.006112	0.004957	0.017734	0.01839	0.007009	0.004578	0.009796524
	Dataset6	0.00527	0.006055	0.015556	0.015799	0.007795	0.005716	0.009365271
	Dataset7	0.004819	0.015277	0.021207	0.028158	0.01883	0.008972	0.016210597
	Dataset8	0.012978	0.012229	0.029409	0.02427	0.0121	0.008316	0.016550425
	Dataset9	0.006565	0.006512	0.02242	0.015853	0.008241	0.004947	0.010756509
	Dataset10	0.005504	0.005128	0.022004	0.014931	0.006832	0.004251	0.009775118

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 11	Dataset1	0.008057	0.006768	0.016651	0.018872	0.009743	0.005507	0.01093307
	Dataset2	0.008313	0.00461	0.017004	0.016378	0.007329	0.004821	0.00974246
	Dataset3	0.009796	0.005984	0.016579	0.019712	0.008628	0.005808	0.01108471
	Dataset4	0.009675	0.00533	0.019906	0.017007	0.007385	0.00516	0.00882832
	Dataset5	0.007607	0.008172	0.023526	0.02232	0.009779	0.004313	0.01261944
	Dataset6	0.00688	0.005269	0.018287	0.01873	0.00701	0.004899	0.01017904
	Dataset7	0.007565	0.00731	0.016357	0.017882	0.008195	0.006689	0.01066616
	Dataset8	0.011603	0.007243	0.020625	0.021342	0.009227	0.006669	0.01278491
	Dataset9	0.008211	0.007513	0.022941	0.019259	0.010019	0.007227	0.01252818
	Dataset10	0.007529	0.006438	0.02082	0.016787	0.00826	0.007141	0.01116272

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 12	Dataset1	0.00523	0.005785	0.013694	0.014168	0.00762	0.003864	0.0083936
	Dataset2	0.007219	0.00579	0.01635	0.016966	0.00765	0.004978	0.00982551
	Dataset3	0.007184	0.004495	0.016877	0.016268	0.006699	0.004322	0.00930766
	Dataset4	0.006433	0.006011	0.013631	0.01876	0.008845	0.005948	0.00993798
	Dataset5	0.007556	0.004285	0.018101	0.018047	0.006514	0.003648	0.00969163
	Dataset6	0.005563	0.005458	0.017027	0.017512	0.007864	0.003752	0.00952934
	Dataset7	0.006223	0.005955	0.016615	0.016302	0.008289	0.00473	0.00968561
	Dataset8	0.005529	0.009061	0.016256	0.021453	0.011647	0.006691	0.01177278
	Dataset9	0.006719	0.005591	0.019753	0.017029	0.007408	0.005199	0.01028324
	Dataset10	0.006281	0.00507	0.021784	0.01623	0.006694	0.005157	0.01020271

Figure E.1: the individual MSE values across the different parking lots in GRU models (cont)

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 13	Dataset1	0.005447	0.002892	0.013134	0.007581	0.003166	0.003779	0.00599972
	Dataset2	0.004507	0.002781	0.015963	0.009099	0.003494	0.004496	0.006723382
	Dataset3	0.005671	0.002549	0.013094	0.008241	0.002949	0.004337	0.006140126
	Dataset4	0.005295	0.00304	0.013773	0.007139	0.003745	0.003835	0.006137945
	Dataset5	0.005013	0.002922	0.016914	0.009105	0.002937	0.003823	0.006785713
	Dataset6	0.004931	0.002677	0.020413	0.009614	0.002953	0.002772	0.007226556
	Dataset7	0.005239	0.00265	0.017336	0.008099	0.003711	0.004274	0.006884922
	Dataset8	0.005366	0.005739	0.020957	0.020475	0.007224	0.008896	0.011442745
	Dataset9	0.003759	0.003125	0.020845	0.007098	0.002852	0.004204	0.00698055
	Dataset10	0.005457	0.002627	0.022402	0.008786	0.002509	0.003506	0.007547613

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 14	Dataset1	0.006791	0.00346	0.016554	0.010573	0.004136	0.005026	0.007756631
	Dataset2	0.006274	0.002286	0.014577	0.009866	0.003038	0.00465	0.006782044
	Dataset3	0.005273	0.002536	0.016011	0.011576	0.003118	0.004171	0.007114119
	Dataset4	0.005981	0.003205	0.01621	0.011263	0.003383	0.005479	0.007586865
	Dataset5	0.005056	0.002482	0.017789	0.008522	0.002591	0.004009	0.006741345
	Dataset6	0.008524	0.002767	0.017544	0.010118	0.003399	0.004114	0.007744448
	Dataset7	0.00759	0.002294	0.020408	0.00917	0.002755	0.00318	0.007566076
	Dataset8	0.00805	0.001941	0.020899	0.011962	0.002624	0.004024	0.008250141
	Dataset9	0.004497	0.002935	0.020948	0.008733	0.002372	0.004043	0.007254733
	Dataset10	0.007678	0.003757	0.022117	0.012009	0.004982	0.005121	0.009277533

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model15	Dataset1	0.003888	0.00271	0.008799	0.003341	0.002575	0.00329	0.00410049
	Dataset2	0.003399	0.003233	0.009711	0.005426	0.002982	0.003519	0.00471164
	Dataset3	0.004291	0.003278	0.010192	0.005745	0.002933	0.003639	0.00501309
	Dataset4	0.004144	0.003017	0.011447	0.00427	0.003939	0.006699	0.00558589
	Dataset5	0.005572	0.002856	0.013877	0.007006	0.003273	0.004606	0.00619824
	Dataset6	0.0045	0.00266	0.012875	0.005558	0.003382	0.002501	0.00524591
	Dataset7	0.004658	0.003265	0.012578	0.00547	0.003208	0.003515	0.0054487
	Dataset8	0.005628	0.002886	0.013736	0.006578	0.003065	0.003655	0.00592491
	Dataset9	0.005321	0.003699	0.01235	0.005956	0.003369	0.004506	0.00586681
	Dataset10	0.004406	0.003408	0.013442	0.005788	0.003033	0.004493	0.00576156

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
GRU Model 16	Dataset1	0.004879	0.003683	0.00991	0.004333	0.002832	0.004905	0.00509053
	Dataset2	0.004943	0.002777	0.010722	0.005122	0.003388	0.003969	0.00515334
	Dataset3	0.004994	0.00432	0.013585	0.006612	0.004008	0.004272	0.00629845
	Dataset4	0.00503	0.005348	0.009192	0.005924	0.004744	0.005354	0.00593223
	Dataset5	0.005925	0.003804	0.011726	0.00485	0.004756	0.006921	0.00633028
	Dataset6	0.004924	0.002943	0.016295	0.006528	0.003715	0.004428	0.00647208
	Dataset7	0.008815	0.002976	0.014829	0.004825	0.003936	0.003626	0.0065011
	Dataset8	0.007565	0.002424	0.013525	0.007578	0.002882	0.005538	0.00658537
	Dataset9	0.005895	0.003449	0.016219	0.006444	0.003891	0.003381	0.0065464
	Dataset10	0.00477	0.00364	0.014363	0.005296	0.003496	0.004579	0.00602418

Figure E.1: the individual MSE values across the different parking lots in GRU models (cont)

- Details of the individual MSE values across the different outputs (parking lots) for the MLP models, these measures are reported previously as an aggregated values in Figure 6.18

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
MLP Model 1	Dataset1	0.00628	0.00293	0.00567	0.00514	0.00186	0.00626	0.00469145
	Dataset2	0.00372	0.00267	0.00297	0.00528	0.00162	0.00545	0.00361751
	Dataset3	0.00044	0.00159	0.00067	0.00405	0.00107	0.00337	0.001866
	Dataset4	0.00217	0.00261	0.00278	0.00615	0.0017	0.00546	0.00347816
	Dataset5	0.00343	0.00313	0.00386	0.00624	0.00127	0.00462	0.00375923
	Dataset6	0.00359	0.00332	0.00398	0.00591	0.00143	0.00525	0.0039133
	Dataset7	0.00144	0.00272	0.0025	0.00557	0.00121	0.00352	0.00282665
	Dataset8	0.002	0.00283	0.0028	0.00562	0.00124	0.00362	0.00301876
	Dataset9	0.00421	0.0033	0.0061	0.00516	0.00174	0.00573	0.0043755
	Dataset10	0.00098	0.00264	0.00211	0.0037	0.00138	0.00406	0.00247789

		APark	BPark	CPark	MPark	NPark	VPark	MSE (mean)
MLP Model 2	Dataset1	0.01292	0.00511	0.01148	0.00579	0.00193	0.00965	0.0078141
	Dataset2	0.01531	0.00761	0.01581	0.00761	0.00296	0.01231	0.01026905
	Dataset3	0.00411	0.00298	0.00343	0.00545	0.00178	0.00544	0.00386652
	Dataset4	0.0086	0.00363	0.00607	0.00664	0.00205	0.00741	0.00573344
	Dataset5	0.00399	0.00328	0.0049	0.00581	0.00126	0.00484	0.00401266
	Dataset6	0.01483	0.00677	0.01617	0.00795	0.00178	0.00963	0.00952243
	Dataset7	0.01455	0.00572	0.01508	0.00922	0.0024	0.00972	0.00944694
	Dataset8	0.01796	0.0079647	0.01802	0.01157	0.0034	0.01192	0.01180566
	Dataset9	0.00812	0.00417	0.01298	0.00464	0.00241	0.00855	0.00680977
	Dataset10	0.01245	0.00623	0.01602	0.00693	0.00274	0.01253	0.0094825

		APark	BPark	CPark	MPark	NPark	VPark	MSE(mean)
MLP Model 3	Dataset1	0.01444	0.00714	0.01997	0.00682	0.00336	0.0121	0.01063716
	Dataset2	0.01406	0.0066	0.01849	0.00821	0.00284	0.01206	0.01037621
	Dataset3	0.0124	0.0074	0.02009	0.007	0.003	0.01284	0.010458
	Dataset4	0.01737	0.00789	0.01786	0.00822	0.00312	0.01344	0.01131773
	Dataset5	0.01202	0.00693	0.01795	0.00795	0.00237	0.01202	0.00987419
	Dataset6	0.00287	0.00279	0.00469	0.00432	0.00141	0.005	0.00351277
	Dataset7	0.00481	0.00386	0.00921	0.0049	0.00176	0.00655	0.00518034
	Dataset8	0.0111	0.00632	0.01487	0.00673	0.00243	0.01085	0.00871831
	Dataset9	0.01084	0.00689	0.01623	0.00662	0.00269	0.01049	0.00896142
	Dataset10	0.02563	0.0111	0.02588	0.00949	0.00525	0.02008	0.01623896

Figure E.2: the individual MSE values across the different parking lots in MLP models

- The results of the pairwise comparison between GRUs models proposed in Chapter 6 are presented in Figure E.3

	m1	m10	m11	m12	m13	m14	m15	m16	m2	m3	m4	m5	m6	m7	m8
m10	1.00000	-	-	-	-	-	-	-	-	-	-	-	-	-	-
m11	1.00000	1.00000	-	-	-	-	-	-	-	-	-	-	-	-	-
m12	1.00000	1.00000	1.00000	-	-	-	-	-	-	-	-	-	-	-	-
m13	0.00011	2.7e-07	6.1e-06	0.01300	-	-	-	-	-	-	-	-	-	-	-
m14	0.00167	5.9e-06	0.00011	0.12148	1.00000	-	-	-	-	-	-	-	-	-	-
m15	1.3e-10	1.0e-13	3.9e-12	6.4e-08	0.98736	0.14375	-	-	-	-	-	-	-	-	-
m16	4.0e-08	4.2e-11	1.4e-09	1.2e-05	1.00000	1.00000	1.00000	-	-	-	-	-	-	-	-
m2	1.00000	0.34365	1.00000	1.00000	1.00000	0.12397	0.86949	1.6e-06	0.00022	-	-	-	-	-	-
m3	1.00000	1.00000	1.00000	1.00000	1.00000	1.3e-05	0.00022	9.3e-12	3.2e-09	1.00000	-	-	-	-	-
m4	1.00000	1.00000	1.00000	1.00000	1.00000	0.02063	0.18257	1.2e-07	2.2e-05	1.00000	1.00000	-	-	-	-
m5	0.14672	1.00000	1.00000	1.00000	0.00211	4.1e-12	1.4e-10	< 2e-16	3.0e-16	0.00014	0.64775	0.00127	-	-	-
m6	1.00000	1.00000	1.00000	1.00000	1.00000	0.01057	0.10112	4.8e-08	9.4e-06	1.00000	1.00000	1.00000	0.00264	-	-
m7	0.59267	1.00000	1.00000	1.00000	0.01218	5.0e-11	1.6e-09	< 2e-16	4.1e-15	0.00097	1.00000	0.00759	1.00000	0.01496	-
m8	1.00000	1.00000	1.00000	1.00000	1.00000	1.3e-06	2.6e-05	6.1e-13	2.4e-10	0.87263	1.00000	1.00000	1.00000	1.00000	1.00000
m9	1.00000	1.00000	1.00000	1.00000	1.00000	1.8e-07	4.2e-06	6.5e-14	2.8e-11	0.27244	1.00000	1.00000	1.00000	1.00000	1.00000

P value adjustment method: bonferroni

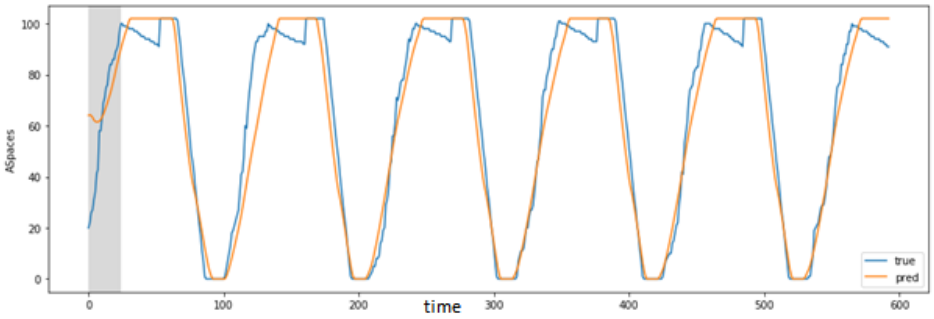
Figure E.3: The results of the pairwise comparison between GRUs models

Appendix F

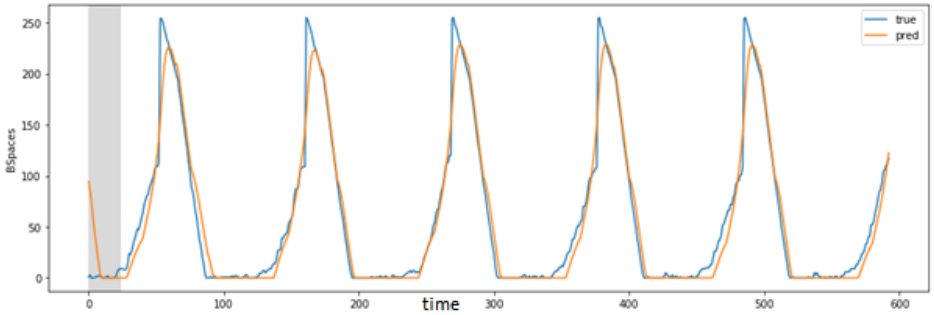
Snapshots of the predictions charts for the University of Essex parking lots

This appendix provides snapshots of the prediction charts from forecasting models presented in Chapter 6. The charts show the difference between the prediction and the real occupancy for the University of Essex parking lots — the prediction from GRUs, MLP, and TBATS models.

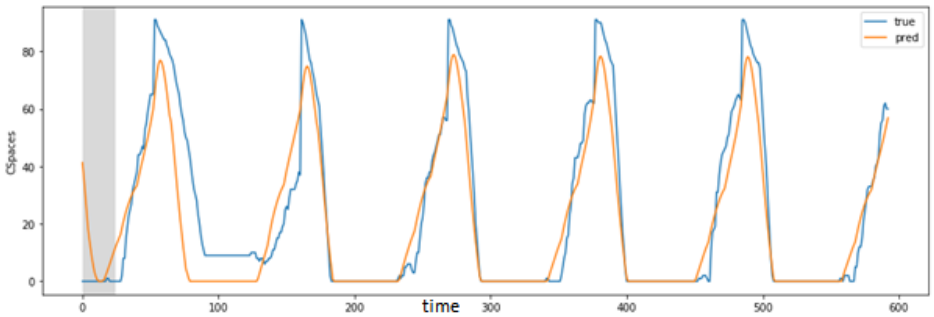
- **GRUs Model**: The prediction of the last week of Autumn term 2015-16 ,Model 1 in Chapter 6, are shown in Figure F.1.
- **Mlp Model**: The prediction of the Autumn term 2015-16 ,from the first experiment in Chapter 6, are shown in Figure F.2.
- **TBATS Model**: The decomposition of each time series are presented in diagrams shown in Figure F.3, and snapshots of the prediction of the Autumn term 2015-16 are shown in Figure F.4.



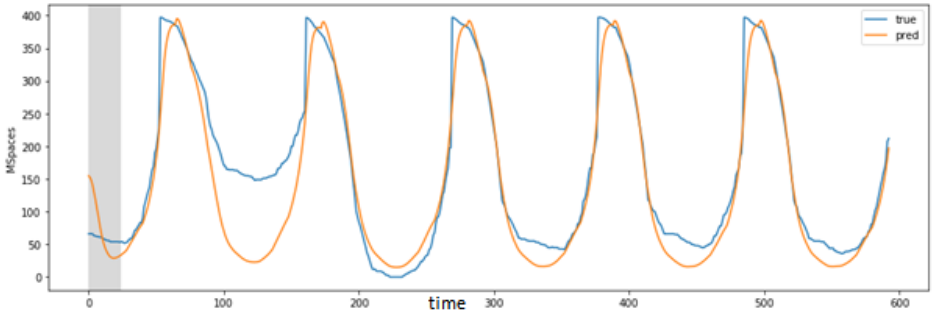
(a) A Park



(b) B Park

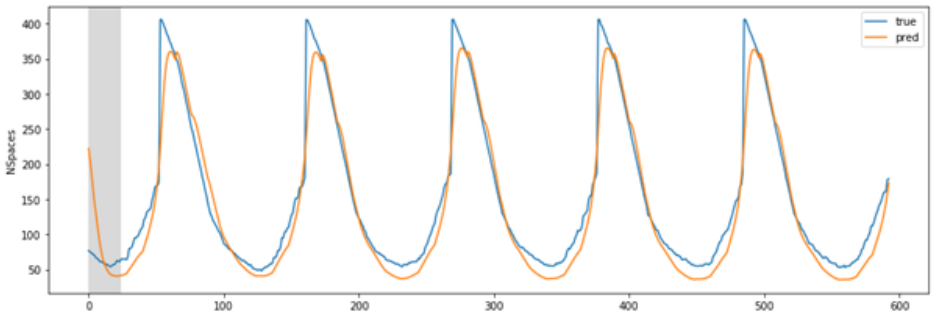


(c) Constable Building Park

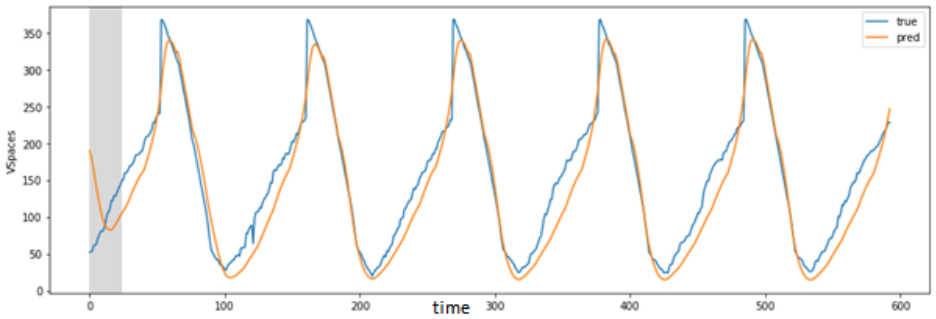


(d) Multi-decked Park

Figure F.1: The difference between the real occupancy of parking lots and the prediction using GRUs model 1

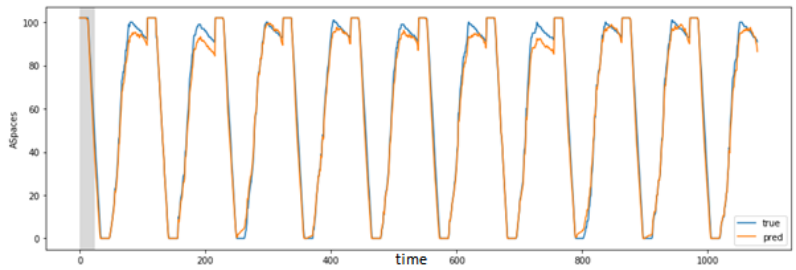


(e) The North Park

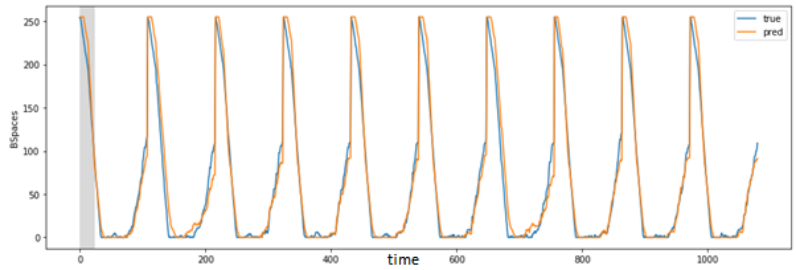


(f) Valley Park

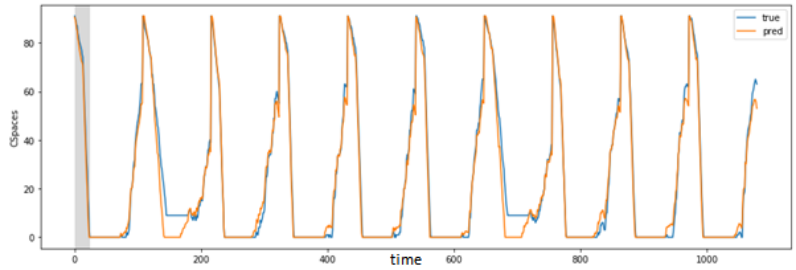
Figure F.1: The difference between the real occupancy of parking lots and the prediction using GRUs model 1 (cont)



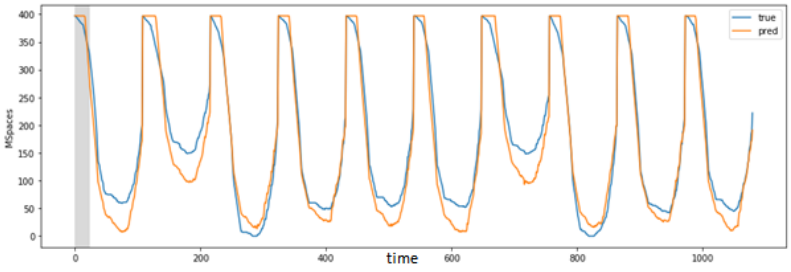
(a) A Park



(b) B Park

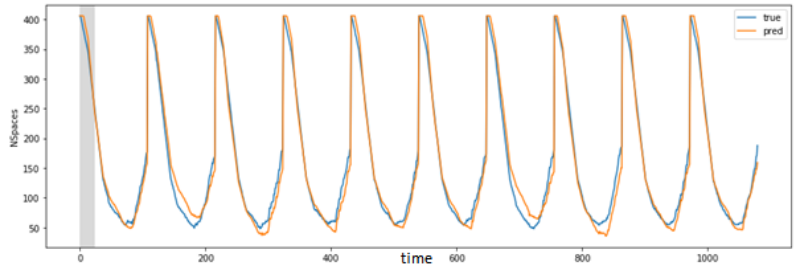


(c) Constable Building Park

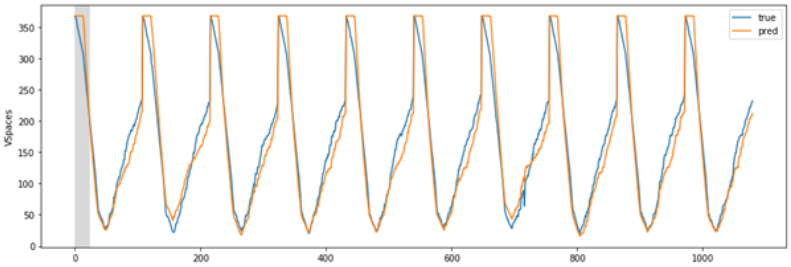


(d) Multi-decked Park

Figure F.2: The difference between the real occupancy of parking lots and the prediction using MLP model 1

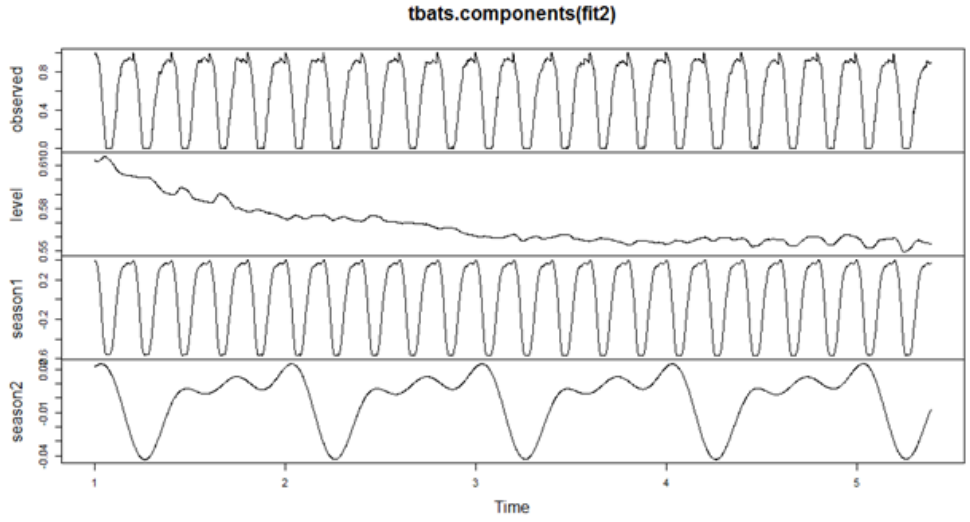


(e) The North Park

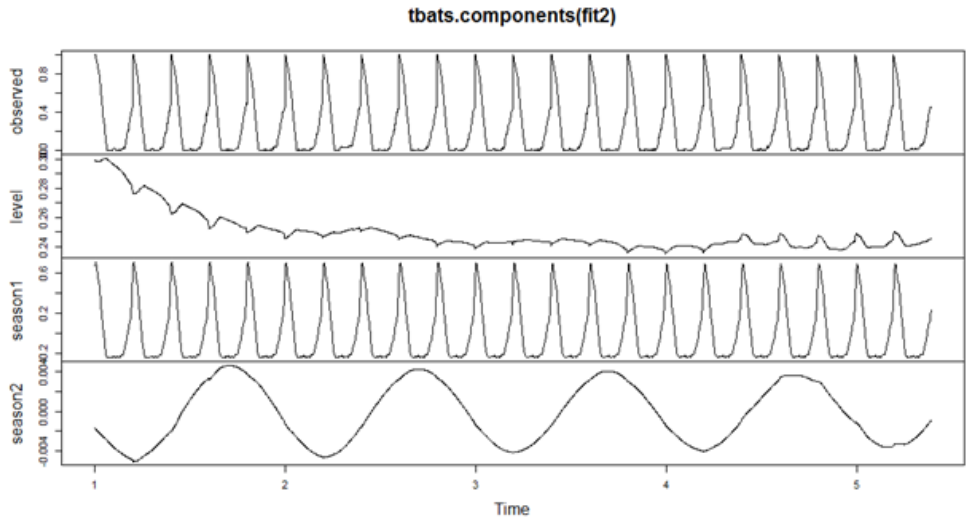


(f) Valley Park

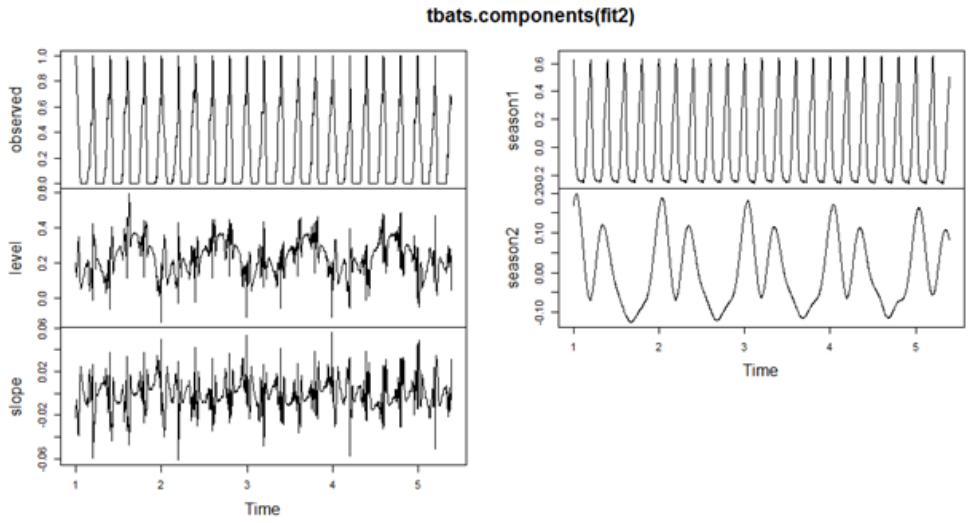
Figure F.2: The difference between the real occupancy of parking lots and the prediction using MLP model 1 (cont)



(a) A Park

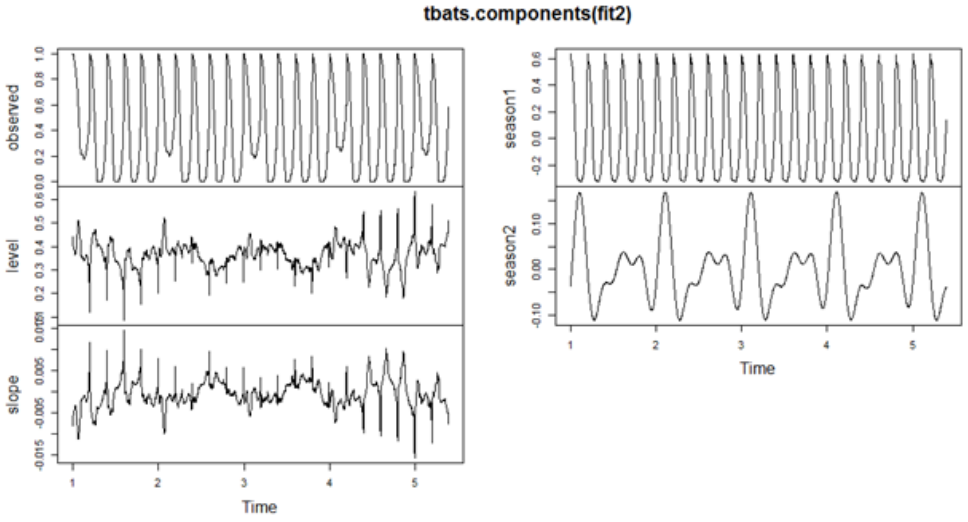


(b) B Park

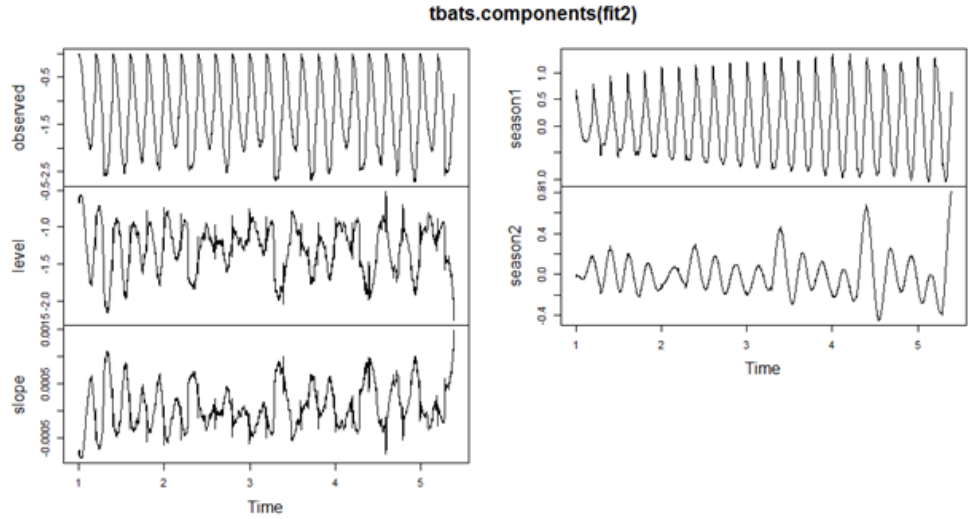


(c) Constable Building Park

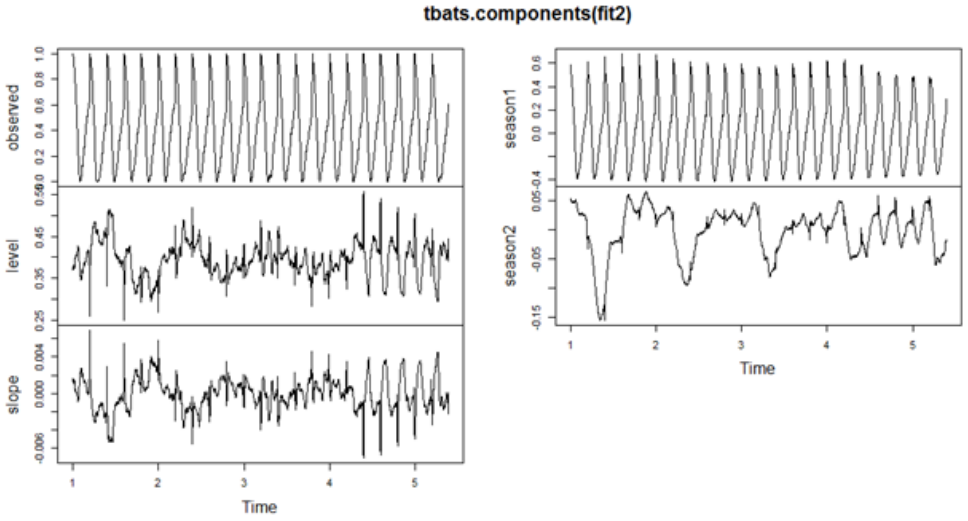
Figure F.3: The decomposition of time series before applying TBATS models



(d) Multi-decked Park



(e) The North Park



(f) Valley Park

Figure F.3: The decomposition of time series before applying TBATS models (cont)

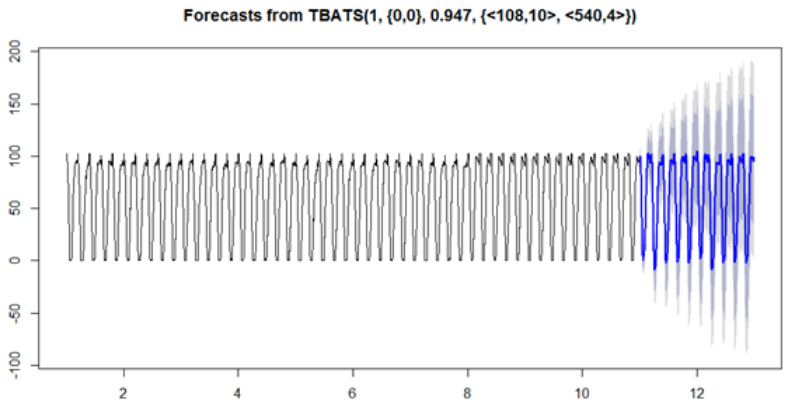


Figure F.4: The difference between the real occupancy of parking lots and the prediction using TBATS model for A Park