Soft Computing manuscript No.
(will be inserted by the editor)

Heuristic Procedures for Improving the Predictability of a Genetic Programming Financial Forecasting Algorithm

Michael Kampouridis · Fernando E. B. Otero

Received: date / Accepted: date

Abstract Financial forecasting is an important area in computational finance. EDDIE (Evolutionary Dynamic Data Investment Evaluator) is an established Genetic Programming financial forecasting algorithm, which has successfully been applied to a number of international datasets. The purpose of this paper is to further improve the algorithm's predictive performance, by incorporating heuristics in the search. We propose the use of two heuristics: a sequential covering strategy to iteratively build a solution in combination with the GP search and the use of an entropy-based dynamic discretisation procedure of numeric values. To examine the effectiveness of the proposed improvements, we test the new EDDIE version (EDDIE 9) across 20 datasets and compare its predictive performance against three previous EDDIE algorithms. In addition, we also compare our new algorithm's performance against C4.5 and RIPPER, two state-of-the-art classification algorithms. Results show that the introduction of heuristics is very successful, allowing the algorithm to outperform all previous EDDIE versions and the state-of-the-art. Results also show that the algorithm is able to return significantly high rates of return across the majority of the datasets.

Keywords genetic programming · financial forecasting · EDDIE · sequential covering · dynamic discretisation

1 Introduction

Data mining is an active research area focused on the design and development of computational methods to discover (create) a model from real-world data (Piatetsky-Shapiro and Frawley, 1991; Fayyad et al, 1996). Classification is one of the main data mining tasks, where the goal is to create a predictive model that represents the relationships between input attributes' values and the values of a class attribute by analysing the data. It usually involves two steps. In the first step (model creation), the algorithm builds a model by analysing cases from the training data. At this point, the algorithm has access to the information of both input and class attributes. In the second step (model evaluation), the model created is used to classify new data, to simulate the use of future (unseen) data. During this step, the algorithm has only access to the input attributes' values to make a prediction. The class attribute value is only used to evaluate the model's prediction: if the predicted value is the same as the actual value, the prediction is marked as correct; otherwise, it is marked as incorrect. Hence, the goal is to create the most accurate model given a set of training data.

Genetic Programming (GP) is an evolutionary technique inspired by natural evolution, where computer programs act as the individuals of a population (Koza, 1992; Poli et al, 2008). GP has been extensively used for classification problems. Its characteristic of being able to produce white-box models makes it a more trustworthy algorithm to its users. GP has been successfully applied for classification in different real-life applications, ranging from medical diagnosis (Giacobini et al, 2014), to fraud detection (Phua et al, 2010) and remote sensing (Dos Santos et al, 2011).

An application that we will be focusing on this paper is financial forecasting, and particularly the prediction of buy opportunities. Financial forecasting is a vital area in computational finance (Tsang and Martinez-Jaramillo, 2004). There are numerous works that attempt to forecast the future price movements of a stock; several examples can be found in Chen (2002); Binner et al (2004). Furthermore, GP has many times been used in the past for financial forecasting (Wilson and Banzhaf, 2010; Wang et al, 2010; Abdelmalek

M. Kampouridis (✉) · F.E.B. Otero
School of Computing, University of Kent, UK
E-mail: M.Kampouridis@kent.ac.uk

et al, 2009; Agapitos et al, 2010; Abdou, 2009), and has shown it is able to identify patterns in the data.

EDDIE is a well-established genetic programming financial forecasting tool, which has been found to outperform traditional decision rule induction methods, such as C4.5, and return high accuracy results over different international stock markets (Li, 2001; Tsang et al, 2000). Recently, EDDIE 8-ATTR (Kampouridis and Otero, 2013) was introduced, which is one of the latest algorithms from the EDDIE series. While previous EDDIE algorithms were using prespecified periods for the indicators from technical analysis (e.g., *20 days* Moving Average, *50 days* Momentum), EDDIE 8-ATTR was the first algorithm to allow these periods to be directly selected by the GP. Thus, instead of the algorithm's user pre-specifying a number of fixed period values for the technical indicators, as it traditionally happens in both academia and industry, EDDIE 8-ATTR allowed the GP to evolve different periods for each technical indicator. In addition, the algorithm used attribute construction, which allowed for a better representation and search of the problem search space. As a result to the above modifications, EDDIE 8-ATTR was able to produce new technical indicators, which improved the algorithm's predictive performance.

The purpose of this paper is to further improve the predictive performance of EDDIE by incorporating heuristics into EDDIE's search. This is paramount, because of the significance of the financial forecasting field itself, which requires the continuous development of new and improved algorithms. The new version, EDDIE 9, uses two well-known heuristics in combination with the GP search: a sequential covering strategy to iteratively build a solution in combination with the GP search and the use of an entropy-based dynamic discretisation procedure of numeric values. A sequential covering strategy has been successfully used in GP for boolean domains (Otero and Johnson, 2013), allowing the automatic decomposition of the original problem into smaller (more tractable) subproblems. Furthermore, supervised learning algorithms usually employ a dynamic discretisation procedure to handle numeric attributes, with the aim of deterministically calculating a threshold value that best fits the data. To examine the effectiveness of the proposed improvements, we test the new EDDIE 9 version across 20 datasets and compare its predictive performance against three previous EDDIE algorithms. In addition, we also compare our new algorithm's performance against C4.5 and RIPPER, two state-of-the-art classification algorithms.

The rest of this paper is organised as follows: Section 2 presents a general overview of previous EDDIE algorithms. Section 3 then details how we incorporated heuristics into EDDIE's search. Sections 4 and 5 then present the experimental setup and discuss the obtained results, respectively. Finally, Section 6 concludes this paper and discusses future work.

2 The EDDIE algorithm

In this section, we present the different versions of EDDIE that are going to be used in our experiments and the reasons for using each version.

2.1 EDDIE 7

EDDIE 7, and EDDIE in general, is a forecasting tool, which learns and extracts knowledge from a set of data. The kind of question EDDIE tries to answer is 'will the price increase within the n following days by $r\%$?' The user first feeds the system with a set of past data; EDDIE then uses this data and through a GP process, it produces and evolves Genetic Decision Trees (GDTs), which make recommendations of buy (1) or not-to-buy (0).

The set of data used is composed of three parts: daily closing price of a stock, a number of attributes and signals. Stocks' daily closing prices can be obtained online in websites such as `http://finance.yahoo.com` and also from financial statistics databases like *Datastream*. The attributes are indicators commonly used in technical analysis (Edwards and Magee, 1992); which indicators to use depends on the user and his/her belief of their relevance to the prediction. The technical indicators that we use in this work are: Moving Average (MA), Trade Break Out (TBR), Filter (FLR), Volatility (Vol), Momentum (Mom), and Momentum Moving Average (MomMA).[1]

The signals are calculated by looking ahead of the closing price for a time horizon of *n* days, trying to detect if there is an increase of the price by *r%* (Tsang et al, 2000). For this set of experiments, *n* was set to 20 and *r* to 4%. In other words, the GP is trying to use some of the above indicators to forecast whether the daily closing price is going to increase by 4% within the following 20 days.

After we feed the data to the system, EDDIE creates and evolves a population of GDTs. Figure 1 presents the Backus Normal Form (BNF) (Backus, 1959) (grammar) of EDDIE 7. As we can see, the root of the tree is an If-Then-Else statement. The first branch is either a boolean (testing whether a technical indicator is greater than/less than/equal to a value), or a logic operator (and, or, not), which can hold multiple boolean conditions. The first branch also includes a $Variable$, which can be anyone of 12 pre-defined technical analysis indicators;[2] this variable is tested whether it is

[1] We use these indicators because they have been proved to be quite useful in developing GDTs in previous works like Martinez-Jaramillo (2007), Allen and Karjalainen (1999) and Austin et al (2004). Of course, there is no reason why not use other information like fundamentals or limit order book. However, the aim of this work is not to find the ultimate indicators for financial forecasting.

[2] These are the 6 indicators mentioned earlier; each indicator has two different period lengths, 12 and 50 days, thus resulting to a total of 12 technical indicators.

```
<Tree> ::= If-Then-Else <Condition> <Tree> <Tree>
         | Decision

<Condition> ::= <Condition> AND <Condition>
              | <Condition> OR <Condition>
              | NOT <Condition>
              | Variable <RelationOperation> Threshold

<Variable> ::= MA_12 | MA_50 | TBR_12 | TBR_50 | FLR_12
             | FLR_50 | Vol_12 | Vol_50 | Mom_12 | Mom_50
             | MomMA_12 | MomMA_50

<RelationOperation> ::= ">" | "<" | "="

Terminals:
Decision is an integer (Positive or Negative implemented)
Threshold is a real number
```

Fig. 1 The Backus Naur Form of the EDDIE 7.

greater than/less than/equal to a $Threshold$, and depending on the result, we move to the 'Then' or the 'Else' branch. The 'Then' and 'Else' branches can be a new GDT, or a decision, to buy or not-to-buy (denoted by 1 and 0).

A sample GDT is presented in Fig. 2. This tree informs us that if the 12 days Moving Average is less than 6.4, then the trader should buy; otherwise, the tree checks whether the 50 days Momentum is greater than 5.57. If the above is true, then the trader is advised not-to-buy; if it's false, the trader is advised to buy.

Depending on the classification of the predictions, we can have four cases: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). As a result, we can use the metrics presented in Equations 1, 2 and 3.

Rate of Correctness:

$$RC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Rate of Missing Chances:

$$RMC = \frac{FN}{FN + TP} \tag{2}$$

Rate of Failure:

$$RF = \frac{FP}{FP + TP} \tag{3}$$

The above metrics combined give the following fitness function, presented in Equation 4:

$$ff = w_1 * RC - w_2 * RMC - w_3 * RF \ , \tag{4}$$

where w_1, w_2 and w_3 are the weights for RC, RMC and RF respectively. These weights are given in order to reflect

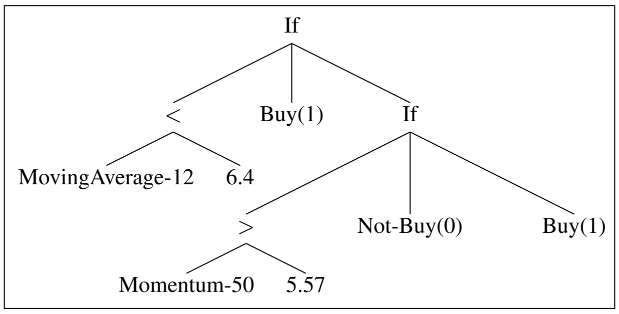


Fig. 2 Sample GDT generated by EDDIE 7.

the preferences of investors. For instance, a conservative investor would want to avoid failure; thus a higher weight for RF should be used. For our experiments, we choose to include strategies that mainly focus on correctness and reduced failure. Thus these weights have been set to 0.6, 0.1 and 0.3 respectively.

The fitness function is a constrained one, which allows EDDIE to achieve lower RF. The effectiveness of this constrained fitness function has been discussed in Tsang et al (2005); Li (2001). The constraint is denoted by R, which consists of two elements represented by a percentage, given by:

$$R = [C_{min}, C_{max}] \ , \tag{5}$$

where $C_{min} = \frac{P_{min}}{N_{tr}} \times 100\%$, $C_{max} = \frac{P_{max}}{N_{tr}} \times 100\%$, and $0 \leq C_{min} \leq C_{max} \leq 100\%$. N_{tr} is the total number of training data cases, P_{min} is the minimum number of positive position predictions required, and P_{max} is the maximum number of positive position predictions required.

Therefore, a constraint of $R = [50, 65]$ means that the percentage of positive signals that a GDT predicts[3] should fall into this range. When this happens, then w_1 remains as it is (i.e. 0.6 in our experiments). Otherwise, w_1 takes the value of zero.

2.1.1 Advantages and Disadvantages of EDDIE 7

EDDIE 7 is a re-implementation of a previous EDDIE algorithm, named FGP-2. This algorithm was one of the very first tested on financial markets and *was found very successful*. It was even compared against a Random Walk model and the well-known classifier C4.5 (Quinlan, 1993); FGP-2 outperformed both. We then added a few more technical indicators and re-implemented Li's FGP-2, and we called it EDDIE 7.

[3] As we have mentioned, each GDT makes recommendations of buy (1) or not-to-buy (0). The former denotes a positive signal and the latter a negative. Thus, within the range of the training period, which is t days, a GDT will have returned a number of positive signals.

```
<Tree> ::= If-Then-Else <Condition> <Tree> <Tree>
           | Decision

<Condition> ::= <Condition> AND <Condition>
          | <Condition> OR <Condition>
          | NOT <Condition>
          | <VarConstructor> <RelationOperation> Threshold

<VarConstructor> ::= MA period | TBR period | FLR period
                   | Vol period | Mom period
                   | MomMA period

<RelationOperation> ::= ">" | "<" | "="

Terminals:
MA, TBR, FLR, Vol, Mom, MomMA are function symbols
Period is an integer within a parameterised range, [MinP, MaxP]
Decision is an integer (Positive or Negative implemented)
Threshold is a real number
```

Fig. 3 The Backus Normal Form of EDDIE 8.

A main disadvantage of the EDDIE 7 algorithm is the fact that it is using technical indicators, pre-specified by the user, and with a fixed period length. As we saw above, EDDIE 7 can accept 12 and 50 days MA, 12 and 50 days TBR, and so on. The periods 12 and 50 are fixed. Nevertheless, one could argue that the choice of these two periods is not the optimal one. To address this issue, we created EDDIE 8, which is presented next.

2.2 EDDIE 8

The novelty of EDDIE 8 is in its extended grammar, which allows the GP to search in the space of indicators to form its Genetic Decision Trees (Kampouridis and Tsang, 2010, 2012). While EDDIE 7 was using 12 indicators pre-specified by the user, EDDIE 8 is not constrained in using any pre-specified indicators, but it is left up to the GP to choose the optimal ones.

As we can see from the grammar in Fig. 3, there is a function called <*VarConstructor*>, which takes two children. The first one is the indicator, and the second one is the <Period>. <Period> is an integer within the parameterised range $[MinP, MaxP]$ that the user specifies. As a result, EDDIE 8 can return decision trees with indicators like 15 days Moving Average, 17 days Volatility, and so on. The advantage of EDDIE 8 is thus that the period is not an issue any more, and it is up to the GP to decide which lengths are more valuable for the prediction. This makes EDDIE 8 a more dynamic and powerful algorithm.

A sample GDT is presented in Fig. 4. As we can observe, the periods 12 and 50 are now in a leaf node, and thus are subject to genetic operators, such as crossover and mutation.

The rest of the EDDIE 8 algorithm behaves in exactly the same way with EDDIE 7.

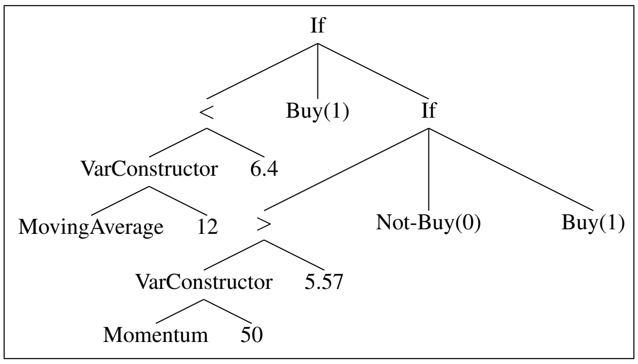


Fig. 4 Sample GDT generated by EDDIE 8.

2.2.1 Advantages and Disadvantages of EDDIE 8

The major advantage of EDDIE 8 is that it is not restricted to use pre-specified periods. In order to see the new algorithm's effectiveness, we compared it with EDDIE 7, over 10 different datasets. EDDIE 7 was proven to be more robust, in terms of average results (Kampouridis and Tsang, 2010, 2012). This was because, EDDIE 8's performance was compromised by the enlarged search space. With the old grammar (EDDIE 7), EDDIE used 6 indicators from technical analysis with two pre-specified period lengths. For instance, if one of the indicators was Moving Average, then the two period lengths used would be 12 and 50 days. On the contrary, EDDIE 8 could use any period within a given parameterised range, which for our experiments was set to 2-65 days. Thus, the algorithm could come up with any indicator within that range, and not just with 12 and 50 days. As we can see, the search space of EDDIE 8 was much bigger than the one of its predecessor.[4] In order to address the issue of exploring better EDDIE 8's big search space, we introduced attribute construction into the algorithm, which will be presented below, in Section 2.3.

On the other hand, an advantage of the algorithm was that it would usually find better optimal solutions (e.g. out of 50 individual runs, EDDIE 8 would normally have its best tree with significantly higher fitness than the best tree of EDDIE 7). Thus, EDDIE 8 was better than EDDIE 7 in terms of best results.

[4] To make this clearer, let us give an example: if a given GP tree can have a maximum of k indicators, then the permutations of the available 12 indicators (we are using 6 different indicators, with 2 periods each, thus $6 * 2 = 12$) under EDDIE 7 are 12^k; on the other hand, if EDDIE 8 is using the same 6 indicators with periods within the range of 2 to 65 days, then the permutations of the available 384 indicators (we are using 6 different indicators with 65-1=64 periods each, thus $64 * 6 = 384$) are 384^k. It is thus obvious that EDDIE 8's search space is significantly larger, which can therefore explain the difficulties of EDDIE 8 of consistently finding good solutions.

```
<Tree> ::= If-Then-Else <Condition> <Tree> <Tree>
         | Decision

<Condition> ::= <Condition> AND <Condition>
  | <Condition> OR <Condition>
  | NOT <Condition>
  | <VarConstructor> <RelationOperation> Threshold
  | <VarConstructor> <RelationOperation> <VarConstructor>

<VarConstructor> ::= MA period | TBR period | FLR period
                   | Vol period | Mom period
                   | MomMA period

<RelationOperation> ::= ">" | "<" | "="

Terminals:
MA, TBR, FLR, Vol, Mom, MomMA are function symbols
Period is an integer within a parameterised range, [MinP, MaxP]
Decision is an integer (Positive or Negative implemented)
Threshold is a real number
```

Fig. 5 The Backus Normal Form of EDDIE 8-ATTR.

2.3 EDDIE 8-ATTR

The previous versions of EDDIE only created GDTs involving the combination of tests composed by a triple (attribute, operator, value), where the value is a numeric constant, as most of machine learning algorithm used for knowledge discovery. In order to allow the creation of new attributes, EDDIE's grammar is extended to allow the creation of tests involving the direct comparison of indicator values using a relational operator, presented in Figure 5. This new version is called EDDIE 8-ATTR.

The main modification is the introduction of the production "*<VarConstructor> <RelationOperation> <VarConstructor>*" to the symbol "*<Condition>*", which defines the rules for creating the conditions of `If-Then-Else` statements of the GDTs. The new grammar allows EDDIE 8-ATTR to create GDTs with the same structure as EDDIE 8 and also GDTs that can define new attributes, in a similar fashion as GP-based attribute construction methods (Hu, 1998; Otero et al, 2003; Krawiec, 2002)—i.e., creating new boolean conditions combining indicators (attributes) using `AND`, `OR` and `NOT` operators. A sample GDT of EDDIE 8-ATTR is presented in Figure 6. It is important to emphasise that this GDT could not be created by the original EDDIE 8, since it involves a condition comparing indicator values directly—i.e., a new boolean attribute represented by the condition "MovingAverage 20 $>$ Momentum 50".

2.3.1 Advantages and Disadvantages of EDDIE 8-ATTR

EDDIE 8-ATTR successfully addressed the problem of ineffective search of EDDIE 8 by introducing attribute construction. Results in Kampouridis and Otero (2013) showed that this attribute construction was beneficial to the algorithm,

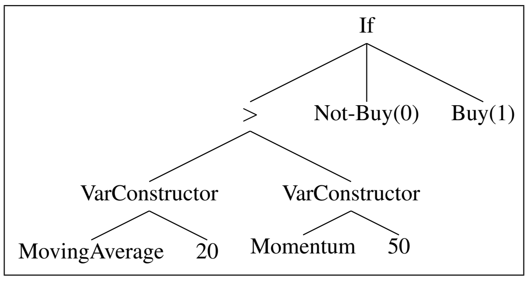


Fig. 6 Sample GDT generated by EDDIE 8-ATTR using a new boolean attribute represented by the condition "MovingAverage 20 $>$ Momentum 50".

which was able to outperform its two predecessors, EDDIE 7 and EDDIE 8, in 8 of the 10 tests examined. Results also indicated that the introduction of more productions that allow the direct comparison of indicators in a single tree, can have a significantly positive effect to the tree's predictive performance.

However, although the results ranked EDDIE 8-ATTR first, they were not statistical significant, indicating that there was still room for improvement in the search process of EDDIE. This thus led us to consider using further heuristic processes in the search, which will be described in Section 3. Next, we briefly present the two state-of-the-art algorithms that are going to act as benchmarks for our experiments.

3 Incorporating heuristics into EDDIE's search

The current EDDIE 8-ATTR algorithm (and also previous EDDIE versions) follows the traditional GP approach, where the algorithm is given solution components (non-terminal and terminal symbols) and the evolutionary process is responsible to find an optimal combination of these components to create a solution to the problem. The aim of this process is to "get a computer to do what needs to be done, without telling it how to do it" (Koza, 1992). If we look at the structure of GDTs generated by EDDIE 8-ATTR, there is a dependency regarding the position of conditions in the tree—e.g., a condition sub-tree that works well in a top level `If-Then-Else` statement does not have the same effect if moved down the tree, since the top-level condition has a bigger effect on the overall tree than a lower-level one. Therefore, individuals are fragile to the application of search operators—it is very easy for a crossover or mutation operation to disrupt the quality of an individual. A small change in a near-optimal individual can produce a very poor individual. That puts extra pressure on the GP search, since not only the correct conditions have to be created, but also they need to be placed in the correct position of the GDT. On top of this, the conditions involve numeric threshold values, testing a particular indicator value using a relational operator (e.g., $\mathrm{MA\_12} < 0.5$), which should be created throughout the evolutionary process and are subject to the search operators; and the "Decision" (output of the GDT) is a random selected integer (1 [buy] or 0 [not-buy]). Therefore, even if the correct condition is created and placed in the correct position, the candidate solution can have a poor fitness if the decision node of the GDT outputs the incorrect value. There is no way of rewarding a candidate solution for partial correctness—even when the structure of the individual contains correct structures, the fitness is calculated taking into account only the correctness of the output.

Let's consider the search strategy employed by other supervised learning algorithms. The majority of them do not attempt to create the complete solution in one step, they instead use a heuristic to decompose the original problem into smaller (more tractable) subproblems. The divide-and-conquer strategy is employed by top-down decision tree induction algorithms to build a decision tree. At the start of the top-down process, an attribute is selected to divide the training data—the first attribute selected corresponds to the root node of the tree. Each branch originating from the node tests a different value in the domain of the attribute and the training data is divided according to the outcome of the tests (i.e., each branch is associated with the subset of the training data that satisfies its test). The procedure of selecting an attribute is then repeated to further divide the subsets.

Rule induction algorithms usually employ a sequential covering strategy to generate a list of classification rules, with the aim of reducing the complexity of creating a complete list by transforming the problem into a sequence of subproblems concerning in creating a single rule. The sequential covering is an iterative procedure, in each iteration a rule is created and the training cases covered by the rule (i.e., the training cases that satisfy the condition of the rule) are removed. A rule solves a subproblem—i.e., it classifies a subset of the training data. This process is then repeated until all training cases are covered by a rule. Note that each iteration is dealing with a different problem, since the training data changes from one iteration to the next.

In this section we present the details of proposed extension to EDDIE (named EDDIE 9), which incorporates heuristics into the GP search. The aim is to apply the GP in a modular way: instead of relying on the GP to evolve a complete GDT (candidate solution), the GP is used in combination with a sequential covering strategy to iteratively build the solution. Additionally, threshold values in the condition tests are automatically determined by a dynamic discretisation procedure, instead of randomly selected. In this way, the GP search is focused in finding an optimal combination of conditions to classify a subset of the training data (i.e., search for subproblem solutions), while the complete solution is created by the sequential covering procedure.

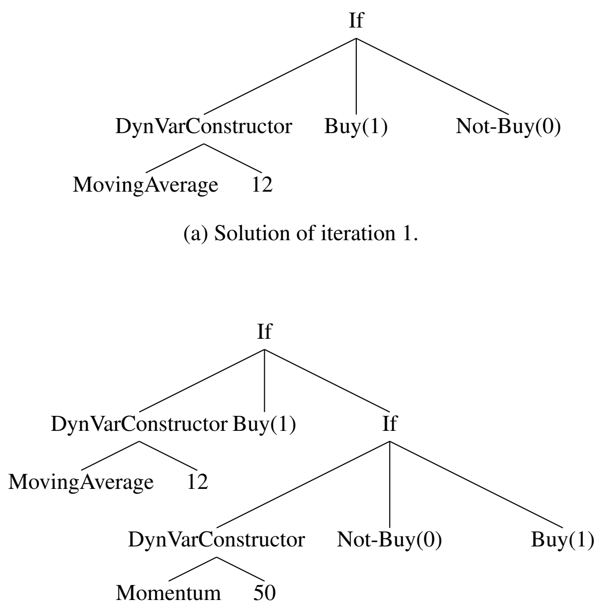
```
Input: training data
Output: GDT
1: $training \leftarrow all\ training\ data$;
2: $GDT \leftarrow \{\}$;
3: while $|training|$ not empty do
4:     $rule \leftarrow RunGP(training)$;
5:     $training \leftarrow training - CoveredData(rule, training)$;
6:     $GDT \leftarrow GDT + rule$;
7: end while
8: return $GDT$;
```

Fig. 7 High-level pseudocode of the sequential covering procedure in EDDIE 9.

3.1 The new EDDIE 9 algorithm

The sequential covering employed in EDDIE 9 is presented in Figure 7. Starting with an empty solution and the complete training data, it evolves a partial solution using the GP, removes all training cases that satisfy its conditions and adds it to the solution. A partial solution is represented by a single `If-Then-Else`, where both `Then` and `Else` correspond to a "Decision"—it is regarded as a partial solution, since it corresponds to a rule that classifies only a subset of the training data. The procedure is then repeated until there are no training cases remaining. In other words, each execution of the GP evolves a rule, which classifies a subset of the training data. Since the training data classified by previous rules is removed, subsequent executions of the GP evolves a different rule—a rule classifying a different subset of the training data.

Given that the GP is not used to generate a complete solution to the problem, EDDIE 9 grammar is a simplified version of the grammar used by EDDIE 8 and EDDIE 8-ATTR. Instead of evolving a complete GDT, a single rule is evolved by the GP. Therefore, no nested `If-Then-Else` are allowed and both `Then` and `Else` are fixed to "Decision". Figure 9 presents the grammar of EDDIE 9. Note that while each execution of the GP creates a rule, where nested `If-Then-Else` are not allowed, the sequential covering creates a GDT to combine the individual solutions created by the different executions of the GP. Individual solutions are combined by nesting their `If-Then-Else`—the `If-Then-Else` of iteration 2 is added under the `Else` branch of the rule created in iteration 1; the one from iteration 3 is added under the `Else` of the rule created in iteration 2, and so forth. The iteratively solution construction is illustrated in Figure 8. As a result of this procedure, the structure of the solution created by EDDIE 9 is the same as the structure of EDDIE 8. The main difference between the algorithms is related to the search strategy: in EDDIE 8 and EDDIE 8-ATTR, the GP is responsible to evolve the complete solution for the problem, while in EDDIE 9 the GP evolves partial solutions that



(a) Solution of iteration 1.

(b) Solution of iteration 2.

Fig. 8 Sample GDT generated by the sequential covering in EDDIE 9: in (a) the solution at the end of the first iteration; in (b) the solution at the end of the second iteration.

are combined by a sequential covering strategy to create a complete solution.

Another difference between the proposed EDDIE 9 and EDDIE 8 is regarding the creation of the tests (conditions) involving the indicators. According to the grammar in Figure 9, there are no "<RelationOperator>" nor "Threshold" symbols in the grammar, and the <VarConstructor>" symbol is replaced by a "<DynVarConstructor>". Instead of relying on the evolutionary process to find good combinations of (indicator, relational operator, threshold) to create the conditions, EDDIE 9 uses a data-driven procedure to automatically determine the relational operator and threshold value, given an indicator and the current training data. Therefore, the GP is responsible for creating the structure of the tests (the combination of conditions in the antecedent of the rule) and the actual relational tests are created in a deterministic way by checking the training data—as detailed in Subsection 3.2. Note that the rule returned as a results of the execution of the GP (line 4 in the pseudocode in Figure 7) already contains the relational operator and threshold values in all conditions for all its conditions, in the same structure as EDDIE 8.

3.2 Dynamic discretisation of indicator values

According to the grammar in Figure 9, the candidate solutions evolved by the GP do not contain complete con-

```
<Tree> ::= If-Then-Else <Condition> Decision Decision

<Condition> ::= <Condition> AND <Condition>
  | <Condition> OR <Condition>
  | NOT <Condition>
  | <DynVarConstructor>

<DynVarConstructor> ::= MA period | TBR period | FLR period
                      | Vol period | Mom period
                      | MomMA period

Terminals:
MA, TBR, FLR, Vol, Mom, MomMA are function symbols
Period is an integer within a parameterised range, [MinP, MaxP]
Decision is an integer (Positive or Negative implemented)
```

Fig. 9 The Backus Normal Form of EDDIE 9.

ditions. Conditions represent boolean expressions that test an indication value against a threshold value using a relational operator. Since the algorithm has the training data, it is possible to check whether a test—a combination of (indicator, relation operator, threshold)—is a good test or not. More importantly, looking at the data, the algorithm can deterministically calculate the test that best fits the data. To this end, EDDIE incorporates a dynamic discretisation procedure based on the entropy measure. This is inspired by similar uses of the entropy measure to discretise continuous attributes (Quinlan, 1993; Otero et al, 2008, 2013).

The dynamic discretisation is a procedure applied to the candidate solutions in order to be able to evaluate them—a solution can only be evaluated if it contains complete conditions—and it has an indirect effect on the solutions' structure. While the structural changes resulting from the discretisation are not permanent modifications, the fitness of the candidate solution directly reflects the quality of the conditions created by the discretisation procedure.

Let's consider the candidate solution illustrated in Figure 10(a). This solution is evolved by the GP following the grammar rules of EDDIE 9. As we mentioned, this solution cannot be directly evaluated since the "<DynVarConstructor>" subtrees do not represent a complete boolean test. Therefore, before evaluating each individual of the GP (candidate solution), the dynamic discretisation procedure is applied to transform the individual representation back to EDDIE 8's representation—illustrated in Figure 10(b). The transformation step works as follows. Starting from the root node of the individual's tree with all the training data available, the tree is traversed in depth-first fashion. When a "<DynVarConstructor>" node is found, the dynamic discretisation procedure is used to select a relation operator and threshold to create a condition. In order to select the best threshold value given the current training data, all values in the domain of the indicator (I), which is specified by "<DynVarConstructor>", are considered. A threshold value

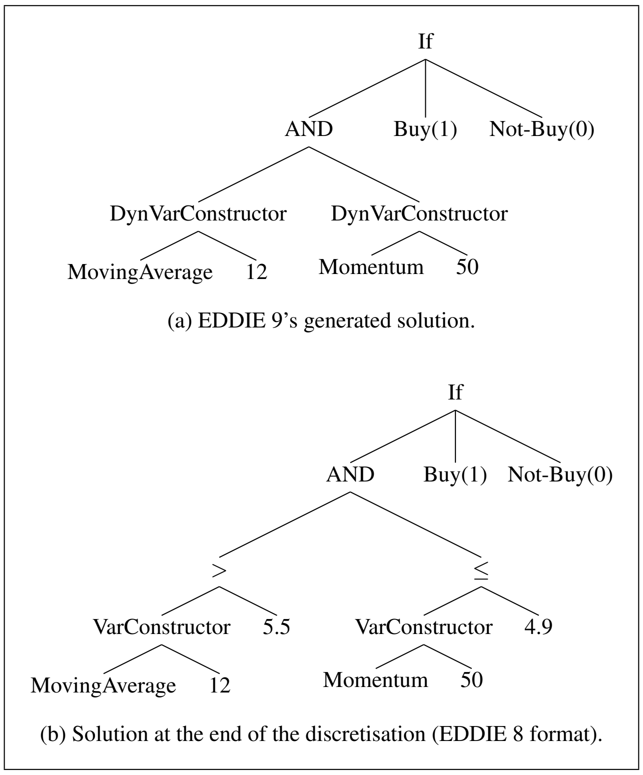


Fig. 10 Illustration of the effect of the dynamic discretisation transformation: in (a) the solution generated by EDDIE 9; in (b) the solution transformed into EDDIE 8's format by the discretisation procedure.

v divides the training data into two sets: the set where $I \leq v$ and another set where $I > v$. The best threshold value corresponds to the value v that minimises the entropy in both sets, given by:

$$\begin{aligned} E(I, v; D) = {} & \frac{|D_{I \leq v}|}{|D|} \cdot \mathit{entropy}(D_{I \leq v}) \\ & + \frac{|D_{I > v}|}{|D|} \cdot \mathit{entropy}(D_{I > v}) \ , \end{aligned} \tag{6}$$

where $|D_{I \leq v}|$ is the total number of examples in the interval $I \leq v$ (subset of the training data where the indicator I has a value less than or equal to v), $|D_{I > v}|$ is the total number of examples in the interval $I > v$ (subset of the training data where the indicator I has a value greater than v) and $|D|$ is the size of the training data. Both values of $\mathit{entropy}(I \leq v)$ and $\mathit{entropy}(I > v)$ are given by:

$$\mathit{entropy}(T) = \sum_{S=0}^{1} - \left(\frac{|T_S|}{|T|} \cdot \log_2 \frac{|T_S|}{|T|} \right) \ , \tag{7}$$

where T is the subset of the training data ($D_{I \leq v}$ or $D_{I > v}$) and T_S is the subset of T that is associated with signal S. After selecting the best threshold value v, the relational operator is selected based on the entropy of the two generated sets, given by:

$$\mathit{operator} = \begin{cases} \leq & \text{, if } \mathit{entropy}(D_{I \leq v}) < \mathit{entropy}(D_{I > v}) \\ > & \text{, if } \mathit{entropy}(D_{I \leq v}) > \mathit{entropy}(D_{I > v}) \end{cases} \ . \tag{8}$$

At the end of the discretisation procedure, the "<DynVarConstructor>" node is replaced by EDDIE's 8 equivalent "<VarConstructor> <RelationOperator> Threshold", as illustrated in Figure 10.

Before continuing traversing the tree, the current training data is filtered according the the parent of the "<DynVarConstructor>". If the parent is a node:

- `AND`: if the "<DynVarConstructor>" is the first child to be evaluated, the training data passed to the second child is filtered to include only the subset that satisfies its conditions; after both children are evaluated the training data is filtered to include only the examples that satisfy both "<DynVarConstructor>" conditions;
- `OR`: after both children are evaluated, the training data is filtered to include only the examples that satisfy one of the "<DynVarConstructor>" conditions;
- `NOT`: after both children are evaluated, the training data is filtered to include only the examples that do not satisfy the "<DynVarConstructor>" condition.

In this way, the dynamic discretisation is tailored to the current training data—when a "<DynVarConstructor>" node is reached, the threshold value and relational operator is selected according to the available data.

4 Experimental Setup

4.1 Algorithms

Our goal is to investigate whether the introduction of the heuristics is beneficial to the EDDIE algorithm. We are thus going to compare the performance of EDDIE 9 to EDDIE 7, EDDIE 8, and EDDIE 8-ATTR. Furthermore, we will also compare the performance of EDDIE 9 to two state-of-the-art classification algorithms, C4.5 (Quinlan, 1993) and RIPPER (Cohen, 1995). More specifically, for the purposes of our experiments, we will be using Weka's (Witten and Frank, 2005) implementation of the algorithms, which are J48 and JRip, respectively.

4.2 Datasets

For our experiments, we run tests for 25 datasets. These datasets consist of daily closing prices of 18 stocks from FTSE 100, and 7 international indices. The 18 FTSE 100 stocks are: Aggreko, Amlin, Barclays, British Petroleum

(BP), Cadbury, Carnival, Easyjet, First, Hammerson, Imperial Tobacco, Marks & Spencer, Next, Royal Bank of Scotland (RBS), Schroders, Sky, Tesco, Vodafone and Xstrata. The 7 indices are: Athens Stock Exchange (Greece), Dow Jones Industrial Average (DJIA - USA), Hang Seng Index (HSI - Hong Kong), Mid-cap Deutscher Aktien Index (MDAX - Germany), and National Association of Securities Dealers Automated Quotations (NASDAQ - USA), Nikkei (Japan), and New York Stock Exchange (NYSE - USA).[5] The training period is 1000 days and the testing period 300.

Because we need to tune the parameters of our new algorithm (EDDIE 9), we will use 5 of the above datasets (randomly selected) for tuning purposes. The remaining 20 will then be used for testing all algorithms. The 5 datasets that will be used for the tuning of EDDIE 9 are: Aggreko, Barclays, First, Marks & Spenser, and Xstrata. To avoid any biases, these 5 datasets will not be used during the testing phase presented in the Results section.

4.3 Parameter Tuning

There are two main parameters that are affected by the sequential covering procedure; these are the number of generations and the population size. As we are creating smaller solutions (rules) to cover a subset of the training data, it might not be necessary to run the GP for many generations—in order to avoid overfitting—or even to have a large number of individuals in the population. Furthermore, another parameter we need to experiment with is the minimum number of cases for the sequential covering iteration, we can choose to stop the procedure if the number of available (not covered) cases falls below a minimum value. Since the number of combinations for the above parameters can be high, we divided the tuning process into two phases.

In the first phase, we were interested in selecting the best combination of generations and population size. We experimented with 4 different settings for number of generations: 15, 25, 35, and 50. We also experimented with the following generation size settings: 50, 100, 200, 300, and 500. For this set of experiments, we kept the number of available (not covered) cases (from sequential covering process) the same, and equal to 0 (i.e., the sequential covering goes on as long as the number of uncovered training cases is greater than or equal to 0). Hence, we tested EDDIE 9 with 15 generations and population 50, 100, 200, 300 and 500 (5 different experiments), then with 25 generations and the same population size combinations, and so on. In total, we ran 20 EDDIE 9 experiments with the above different settings. We then ranked the results in terms of Fitness, RC, RMC, and RF by using the non-parametric Friedman statistical test with the post-hoc Hommel's test (Demšar, 2006; García and Herrera, 2008). Results showed that the top ranking algorithm for Fitness was the 25-300-0 configuration (i.e., 25 generations, 300 population size, and availability value equal to 0), for RC the 15-200-0 configuration, for RMC the 25-300-0, and for RF the 35-100-0. We thus decided to tune the above configurations with the availability parameter from sequential covering.

Tuning the number of available cases in sequential covering was the second phase of our parameter tuning. We had already tested the value 0, and we then tested the values 25, 50, 75 and 100. Thus, for each 'winning' configuration from the first phase, we replaced the 0 with 25, 50, 75, and 100. Therefore, in addition to 25-300-0, we also tested 25-300-25, 25-300-50, 25-300-75 and 25-300-100. We also did the same with 15-200-0, and 35-100-0. Hence in total we ran an extra 12 set of experiments. We again followed the same ranking process as with the first phase. Results showed that configurations 35-100-0 and 15-200-0 had equal ranks across the four performance metrics. At the end, we selected the 15-200-0 configuration to be our standard EDDIE 9 algorithm, as it has a lower number of generations and thus runs much faster than the equally performing 35-100-0.

We did not experiment with any other GP parameters, as we did not think that these would be affected by the introduction of the new heuristics into EDDIE. These GP parameters have been tuned in previous experiments and are presented next.

Table 1 GP parameters values used in the experiments.

GP Parameters	
Max Initial Depth	6
Max Depth	8
Generations	50
Population size	500
Tournament size	2
Reproduction probability	0.1
Crossover probability	0.9
Mutation probability	0.01
Weight $\{w_1, w_2, w_3\}$	$\{0.6, 0.1, 0.3\}$
Period	[2,65]

4.4 Other Parameters

The remaining GP parameters for the algorithms tested in this paper are presented in Table 1. For statistical purposes, the GP is run 50 times. The process is as follows. We create a population of 500 GDTs, which are evolved for 50 generations, over a training period of 1000 days. At the last generation, the best performing GDT in terms of fitness is saved and applied to the testing data. As already explained, this procedure is done for 50 individual runs.

Both C4.5 and RIPPER are run with the default values: C4.5 $\{$*confidence factor in the pruning equal to* 0.25, *minimum number of cases per leaf node equal to* 2$\}$; RIPPER

[5] The datasets used in our experiments can be downloaded from: http://www.cs.kent.ac.uk/people/staff/mk451/datasets.html

Table 2 Summary results for the different EDDIE versions: EDDIE 7 (ED7), EDDIE 8 (ED8), EDDIE 8-ATTR (ED8-AT), and EDDIE 9 (ED9). The metrics used for the comparison of the algorithms are: Fitness, Rate of Correctness (RC), Rate of Missing Chances (RMC), and Rate of Failure (RF). Results are in the [0, 1] scale—best results are shown in boldface. The last row of the table shows the average raking according to the Friedman statistical test, where the lower the rank the better the algorithms' performance.

	Fitness				**Rate of Correctness (RC)**				**Rate of Missing Chances (RMC)**				**Rate of Failure (RF)**			
Algorithm	ED7	ED8	ED8-AT	ED9	ED7	ED8	ED8-AT	ED9	ED7	ED8	ED8-AT	ED9	ED7	ED8	ED-AT	ED9
Amlin	0.54	0.55	0.54	**0.58**	0.51	0.52	0.51	**0.55**	0.42	0.43	0.42	**0.33**	0.41	0.4	0.42	**0.39**
Athens	**0.54**	**0.54**	0.52	0.51	**0.53**	**0.53**	0.52	0.51	**0.19**	0.25	0.27	0.26	**0.53**	**0.53**	0.55	0.55
BP	**0.60**	0.58	0.58	0.59	**0.56**	0.53	0.54	0.55	0.39	0.42	0.40	**0.35**	**0.32**	0.34	0.34	0.35
Cadbury	0.66	**0.68**	**0.68**	0.63	0.64	0.65	**0.66**	0.60	0.18	**0.17**	0.18	0.23	0.34	0.33	**0.32**	0.36
Carnival	**0.50**	**0.50**	0.49	0.45	**0.51**	**0.51**	0.50	0.44	0.17	0.15	**0.14**	**0.14**	**0.63**	**0.63**	**0.63**	0.66
DJIA	**0.69**	**0.69**	0.68	0.68	**0.65**	**0.65**	0.64	0.64	**0.13**	0.16	0.2	0.16	0.29	**0.28**	**0.28**	0.30
Easyjet	0.55	0.51	0.53	**0.59**	0.49	0.45	0.47	**0.53**	0.58	0.67	0.63	**0.45**	**0.29**	0.31	**0.29**	**0.29**
Hammerson	0.51	0.55	**0.59**	0.56	0.49	0.53	**0.57**	0.53	0.51	0.38	0.32	**0.27**	0.45	0.43	**0.4**	0.43
HSI	0.65	0.65	0.66	**0.68**	0.61	0.6	0.61	**0.65**	0.24	0.27	0.27	**0.13**	0.30	0.29	**0.28**	0.31
Imp	**0.62**	0.61	**0.62**	**0.62**	0.58	0.57	**0.59**	0.58	0.48	0.45	0.44	**0.34**	**0.27**	0.29	0.28	0.32
MDAX	0.52	0.52	**0.54**	0.53	0.49	0.49	**0.51**	0.49	0.24	0.20	0.18	**0.15**	0.52	0.51	**0.50**	0.51
NASDAQ	0.61	0.60	0.61	**0.63**	0.57	0.55	0.56	**0.59**	0.37	0.42	0.36	**0.23**	**0.30**	0.31	0.32	0.33
Next	0.55	0.51	0.55	**0.60**	0.50	0.47	0.50	**0.55**	0.41	0.48	0.43	**0.30**	0.37	0.39	0.36	**0.35**
NIKEI	0.56	0.54	0.53	**0.58**	0.54	0.52	0.51	**0.55**	0.23	0.30	0.34	**0.16**	0.46	0.48	0.48	**0.45**
NYSE	0.56	0.57	**0.58**	0.56	0.54	**0.55**	**0.55**	0.53	0.26	0.21	**0.20**	**0.20**	0.45	**0.44**	**0.44**	0.46
RBS	0.54	0.56	**0.58**	0.57	0.50	0.53	**0.54**	**0.54**	0.41	0.37	0.34	**0.32**	0.40	0.39	**0.38**	0.39
Schroders	0.59	0.60	**0.62**	0.61	0.55	0.57	**0.59**	0.57	0.36	0.33	0.26	**0.24**	0.37	**0.36**	**0.36**	0.38
Sky	0.57	0.56	0.54	**0.62**	0.53	0.52	0.49	**0.58**	0.43	0.47	0.49	**0.34**	0.34	0.34	0.36	**0.32**
Tesco	0.64	0.65	0.65	**0.66**	0.59	0.61	0.61	**0.63**	0.30	0.29	0.27	**0.22**	0.30	**0.28**	0.29	0.30
Vodafone	0.46	**0.52**	0.49	0.51	0.44	0.50	0.47	**0.48**	0.37	0.30	0.32	**0.21**	0.56	**0.51**	0.54	0.52
Avg. Rank	2.75	2.85	2.35	**2.05**	2.80	2.75	2.35	**2.10**	2.90	3.05	2.75	**1.30**	2.70	2.19	**2.10**	2.99

$\{1/3$ *of the data used for pruning*, *weight of a case in a rule equals to* $2\}$. Since these algorithms are deterministic, they are executed only once on each dataset.

5 Results

This section presents the results from our experiments. We will first present the results from the comparison of our proposed version EDDIE 9 with previous EDDIE versions, namely EDDIE 7, EDDIE 8 and EDDIE 8-ATTR. Then, we will also compare the performance of EDDIE 9 with C4.5 and RIPPER, two state-of-the-art classification algorithms.

5.1 Comparisons with previous EDDIE versions

Table 2 presents the average results, over 50 runs, for all EDDIE algorithms for Fitness, RC, RMC, and RF. When an algorithm has the best value for a given dataset among all other algorithms, then the respective value is in bold fonts. As we can observe, EDDIE 9 has done quite well in terms of Fitness and RC, where it returned the best results in 9 and 10 datasets, respectively. Results for RMC were even better for EDDIE 9, where it returned the best RMC in 17 datasets out of the 20 tested. Lastly, in terms of RF, EDDIE 9 returned the best values in 5 datasets. The last row of the table presents the average rank of each algorithm—the lower the average rank, the better the algorithm's performance—for each metric. As we can observe, EDDIE 9 ranks first in terms of Fitness, RC, and RMC. The average rank was calculated by running the non-parametric Friedman test.

Table 3 Statistical test results according to the non-parametric Friedman test with the Hommel's post-hoc test. Statistically significant differences at the $\alpha = 0.05$ level are in bold.

Algorithm	Average Rank	Adjusted p_{Homm}
(i) Fitness		
EDDIE 9 (c)	2.05	–
EDDIE 8-ATTR	2.35	0.4624
EDDIE 7	2.75	0.1728
EDDIE 8	2.85	0.1296
(ii) Rate of Correctness (RC)		
EDDIE 9 (c)	2.10	–
EDDIE 8-ATTR	2.35	0.5402
EDDIE 8	2.75	0.2226
EDDIE 7	2.80	0.1728
(iii) Rate of Missing Chances (RMC)		
EDDIE 9 (c)	1.30	–
EDDIE 8-ATTR	**2.75**	**3.82E-4**
EDDIE 7	**2.90**	**1.77E-4**
EDDIE 8	**3.05**	**5.44E-4**
(iv) Rate of Failure (RF)		
EDDIE 8-ATTR (c)	2.10	–
EDDIE 8	2.19	0.0824
EDDIE 7	2.70	0.2832
EDDIE 9	2.99	0.8064

These observations are further supported by the Hommel's post-hoc test (Demšar, 2006; García and Herrera, 2008), and are presented in Table 3. For each algorithm, the table again shows the average rank according to the Friedman test (first column), and the adjusted p-value of the statistical test when that algorithm's average rank is compared to the average rank of the algorithm with the best rank (control algorithm) according to the Hommel's post-hoc test (second column). When statistically significant differences between the average ranks of an algorithm and the control algorithm

Table 4 Average results for comparison with state-of-the-art (C4.5, RIPPER). Results are in the [0, 1] scale—best results are shown in boldface. The last row of the table shows the average raking according to the Friedman statistical test, where the lower the rank the better the algorithms' performance.

	Rate of Correctness (RC)			**Rate of Missing Chances (RMC)**			**Rate of Failure (RF)**		
Algorithm	EDDIE 9	C4.5	RIPPER	EDDIE 9	C4.5	RIPPER	EDDIE 9	C4.5	RIPPER
Amlin	**0.550**	0.453	0.467	**0.330**	0.457	0.531	**0.390**	0.469	0.450
Athens	**0.510**	0.467	0.413	**0.260**	0.276	0.394	**0.550**	0.576	0.621
BP	**0.550**	0.467	0.510	**0.350**	0.474	0.438	**0.350**	0.400	0.363
Cadbury	**0.600**	0.527	0.553	**0.230**	0.350	0.433	0.360	0.403	**0.354**
Carnival	0.440	**0.587**	0.503	**0.140**	0.239	0.337	0.660	**0.593**	0.659
DJIA	0.640	**0.670**	0.667	0.160	**0.062**	0.118	0.300	0.303	**0.287**
Easyjet	0.530	0.540	**0.550**	**0.450**	0.498	0.458	0.290	**0.266**	0.276
Hammerson	0.530	0.507	**0.593**	**0.270**	0.331	0.396	0.430	0.449	**0.350**
HSI	**0.650**	0.573	0.627	**0.130**	0.248	0.150	**0.310**	0.332	0.316
Imp	0.580	**0.593**	0.567	**0.340**	0.355	0.452	0.320	0.318	**0.311**
MDAX	0.490	**0.503**	0.487	**0.150**	0.397	0.493	0.510	**0.508**	0.526
NASDAQ	**0.590**	0.573	0.450	**0.230**	0.396	0.584	0.330	**0.282**	0.359
Next	**0.550**	0.470	0.450	**0.300**	0.482	0.568	**0.350**	0.380	0.377
NIKEI	0.550	**0.617**	0.380	0.160	**0.062**	0.677	0.450	**0.410**	0.597
NYSE	0.530	**0.563**	0.500	**0.200**	0.364	0.525	0.460	**0.411**	0.458
RBS	0.540	**0.567**	0.493	**0.320**	0.332	0.370	0.390	**0.359**	0.420
Schroders	0.570	0.493	**0.643**	**0.240**	0.276	0.309	0.380	0.438	**0.290**
Sky	0.58)	**0.583**	0.427	**0.340**	0.497	0.626	0.320	**0.222**	0.407
Tesco	**0.630**	0.587	0.600	**0.220**	0.429	0.278	0.300	**0.235**	0.299
Vodafone	0.480	**0.633**	0.450	**0.210**	0.261	0.380	0.530	**0.410**	0.558
Avg. Rank	**1.75**	1.90	2.34	**1.20**	2.40	2.40	2.27	1.92	**1.80**

at the 5% level ($p \leq 0.05$) are observed, the line is tabulated in bold face.

As we can observe in Table 3, EDDIE 9 ranks first in terms of Fitness, RC, although not significantly at the 5% level. EDDIE 8-ATTR ranks then second, in both cases, and EDDIE 7 and EDDIE 8 take the last two positions. In addition, EDDIE 9 ranks first and significantly outperforms all other EDDIE algorithms for RMC. Lastly, EDDIE 9 ranks in the last position in terms of RF. This appears to happen due to the fact that EDDIE 9 has increased the number of false positive signals (FP), in the expense of true negatives (TN). However, this increase does not affect the other three metrics (Fitness, RC, RMC). Overall, EDDIE 9 has done very well, as it ranked first in three out of the four performance metrics tested. Hence, we believe that this makes it an important addition to the EDDIE family.

5.2 Comparisons with state-of-the-art

This section presents the comparative results between our new algorithm, EDDIE 9, and two state-of-the-art algorithms, C4.5 and RIPPER. Since C4.5 and RIPPER use a different fitness function than the EDDIE algorithm, in this section we will not be comparing this value. We will only be making comparisons for the remaining performance metrics, i.e., RC, RMC, and RF. We first present the average results, over 50 individual runs, for EDDIE 9; C4.5 and RIPPER are deterministic algorithms, hence they were run once per dataset. These results will be presented in Section 5.2.1.

However, while average results are meaningful in Machine Learning as they can give us an idea of the expected performance of a given algorithm, in this section we will

Table 5 Statistical test results for average performance according to the non-parametric Friedman test with the Hommel's post-hoc test. Statistically significant differences at the $\alpha = 0.05$ level are in bold.

Algorithm	Average Rank	Adjusted p_{Homm}
(ii) Rate of Correctness (RC)		
EDDIE 9 (c)	1.75	–
C4.5	1.90	0.6352
RIPPER	2.34	0.1155
(iii) Rate of Missing Chances (RMC)		
EDDIE 9 (c)	1.20	–
C4.5	**2.40**	**1.47E-4**
RIPPER	**2.40**	**1.47E-4**
(iv) Rate of Failure (RF)		
RIPPER (c)	1.80	–
C4.5	1.92	0.6926
EDDIE 9	2.27	0.2661

also be presenting the best results for EDDIE 9. Best results refers to the *best tree* in terms of fitness, out of 50 runs in the training set, which was then applied to the unseen testing set—i.e., a single best tree is selected from the 50 runs. Thus, below in Section 5.2.2 we present the RC, RMC, and RF results for the best tree of each algorithm. This is particularly meaningful in the financial sector, because if an investor was using EDDIE 9 in the stock market, s/he would first run the algorithm multiple times and then select the best performing tree (model) for trading. Therefore, having an algorithm with very good performance in terms of *best tree* is an important aspect in the financial forecasting sector. These results will be presented in Section 5.2.2.

5.2.1 Average results

The good performance of EDDIE 9 continues also when compared to the state-of-the-art. Table 4 presents the aver-

Table 6 Best results for comparison with state-of-the-art (C4.5, RIPPER). Results are in the [0, 1] scale—best results are shown in boldface. The last row of the table shows the average raking according to the Friedman statistical test, where the lower the rank the better the algorithms' performance.

	Rate of Correctness (RC)			**Rate of Missing Chances (RMC)**			**Rate of Failure (RF)**		
Algorithm	EDDIE 9	C4.5	RIPPER	EDDIE 9	C4.5	RIPPER	EDDIE 9	C4.5	RIPPER
Amlin	**0.590**	0.453	0.467	**0.250**	0.457	0.531	**0.370**	0.469	0.450
Athens	**0.480**	0.467	0.413	**0.160**	0.276	0.394	**0.560**	0.576	0.621
BP	**0.560**	0.467	0.510	**0.360**	0.474	0.438	**0.330**	0.400	0.363
Cadbury	**0.650**	0.527	0.553	**0.030**	0.350	0.433	0.370	0.403	**0.354**
Carnival	0.310	**0.587**	0.503	**0.040**	0.239	0.337	0.700	**0.593**	0.659
DJIA	0.580	**0.670**	0.667	0.350	**0.062**	0.118	**0.280**	0.303	0.287
Easyjet	**0.570**	0.540	0.550	**0.290**	0.498	0.458	0.330	**0.266**	0.276
Hammerson	0.520	0.507	**0.593**	**0.230**	0.331	0.396	0.450	0.449	**0.350**
HSI	0.500	0.573	**0.627**	0.580	0.248	**0.150**	**0.260**	0.332	0.316
Imp	0.570	**0.593**	0.567	**0.170**	0.355	0.452	0.390	0.318	**0.311**
MDAX	**0.510**	0.503	0.487	**0.090**	0.397	0.493	**0.500**	0.508	0.526
NASDAQ	**0.600**	0.573	0.450	**0.200**	0.396	0.584	0.330	**0.282**	0.359
Next	**0.500**	0.470	0.450	**0.420**	0.482	0.568	**0.370**	0.380	0.377
NIKEI	0.560	**0.617**	0.380	**0.010**	0.062	0.677	0.450	**0.410**	0.597
NYSE	0.560	**0.563**	0.500	**0.120**	0.364	0.525	0.440	**0.411**	0.458
RBS	**0.630**	0.567	0.493	**0.020**	0.332	0.370	0.380	**0.359**	0.420
Schroders	0.580	0.493	**0.643**	0.310	**0.276**	0.309	0.360	0.438	**0.290**
Sky	**0.720**	0.583	0.427	**0.040**	0.497	0.626	0.290	**0.222**	0.407
Tesco	**0.670**	0.587	0.600	**0.150**	0.429	0.278	0.280	**0.235**	0.299
Vodafone	0.470	**0.633**	0.450	**0.250**	0.261	0.380	0.530	**0.410**	0.558
Avg. Rank	**1.60**	2.05	2.34	**1.25**	2.05	2.69	**1.85**	1.90	2.24

age results for EDDIE 9, and Table 5 presents the statistical test results. EDDIE 9 ranks again first for RC and RMC, with the latter rank being significant at the 5% level. In addition, EDDIE 9 ranks last in terms of RF. The reason of the poor performance in terms of RF appears again to be a high number of FP signals.

5.2.2 *Best results*

Table 6 presents the best results for EDDIE 9, and Table 7 presents the statistical test results. What we can observe here is that selecting the *best tree*[6] of EDDIE 9 has introduced further improvements in the ranking results. In terms of RC, EDDIE 9 again ranks first, and is also significantly better than the RIPPER algorithm. In terms of RMC, EDDIE 9 maintains its previous excellent performance and significantly outperforms C4.5 and RIPPER. Lastly, in terms of RF, EDDIE 9 ranks first for the first time, although not significantly. Nevertheless, our findings imply that selecting the best tree for trading has only positive effects, with no negatives.

5.3 Computational times

Table 8 presents the computational times in seconds that each of the EDDIE algorithms took to complete a single run. Results are over all datasets. When compared to the computational time for inducing a decision tree of C4.5 and RIPPER, the EDDIE variations are a factor of 20 to 70 slower

[6] Refer to Section 5.2 for the definition of *best tree*.

Table 7 Statistical test results for Best according to the non-parametric Friedman test with the Hommel's post-hoc test. Statistically significant differences at the $\alpha = 0.05$ level are in bold.

Algorithm	Average Rank	Adjusted p_{Homm}
(ii) Rate of Correctness (RC)		
EDDIE 9 (c)	1.60	–
C4.5	2.05	0.1547
RIPPER	**2.34**	**0.0354**
(iii) Rate of Missing Chances (RMC)		
EDDIE 9 (c)	1.25	–
C4.5	**2.05**	**0.0114**
RIPPER	**2.69**	**9.06E-6**
(iv) Rate of Failure (RF)		
EDDIE 9 (c)	1.85	–
C4.5	1.90	0.8743
RIPPER	2.24	0.4111

(C4.5) and a factor of approximately 7 to 23 slower (RIPPER). This is expected, given that both C4.5 and RIPPER create and prune a single candidate decision tree using a deterministic procedure, while the EDDIE variations multiple candidate decision trees are created before finding the best decision tree. With regards to the EDDIE algorithms, EDDIE 9 is the slowest one, and takes about 70 seconds on average to complete a single run. This makes it slower in a factor of 3.5 when compared to the fastest EDDIE algorithm, which is EDDIE 7, and only takes approximately 20.5 seconds on average. Thus, the addition of the heuristic processes introduced in EDDIE 9 (sequential covering and dynamic discretisation) has slowed down the algorithm.

However, it should be noted that in the current financial forecasting application the computational time taken by the algorithms to induce a classification model has a relatively minor importance, since it represents an off-line application (i.e., it involves daily predictions rather than intra-day

Table 8 Average computational times (over all data sets) in seconds to complete a single run for each of the EDDIE algorithms on a 2.53 GHz Intel Xeon computer. The deterministic C4.5 and RIPPER algorithms take on average 1 and 3 seconds, respectively, to complete an individual run.

Algorithm	Time (seconds)
EDDIE 7	20.478
EDDIE 8	38.140
EDDIE 8-ATTR	48.112
EDDIE 9	70.358

ones). Hence, the introduced improvements in the performance metrics justify the slower exectution speed of EDDIE 9. In addition, GP algorithms can be easily parallelised since each tree builds and evaluates a candidate solution independent from all other trees in the population. Therefore, a large speed up could be obtained by running a parallel version of EDDIE, as it has actually been shown in (Brookhouse et al, 2014), where speed ups of up to 21 times were observed.

5.4 Average Annualised Rate of Return

Lastly, in addition to the performance metrics mentioned in the previous sections (i.e., fitness, RC, RMC, RF), we also decided to use an additional metric for the comparison of the algorithms. This metric is related to the return the algorithm yields, and is called Average Annualised Rate of Return (AARR). The formula for this metric is presented below. It should be stated that AARR is not part of the fitness function. However, rate of return is a very important investment metric, and that is why we use it as a reference. Therefore, we use an investment performance criterion (AARR), based on the following hypothetical trading behaviour.

***Hypothetical Trading Behaviour**: We assume that when a positive position is predicted by a GDT, one unit of money is invested in a stock reflecting the current closing price. If the closing price does rise by r% or more at day t within the next n trading days, we then sell the portfolio at the closing price of day t. If not, we sell the portfolio on the n_{th} day, regardless of the price.*

Given a positive position predicted, for example, the i_{th} positive position, for simplicity, we ignore transaction cost, and annualise its return by the following formula:

$$ARR_i = \frac{250}{t} * \frac{P_t - P_0}{P_0} \tag{9}$$

where P_0 is the buy price, P_t is the sell price, t is the number of days in markets, 250 is the number of total trading days in one calendar year. Given a GDT that generates N_+ number of positive positions over the period examined, its average ARR is shown in Equation (10):

Table 9 Average Annualised Rate of Return (AARR) for the 20 datasets for EDDIE 9. 13 out of the 20 datasets have achieved a profit ($AARR > 1$).

Dataset	AARR	Dataset	AARR
Amlin	**1.52**	MDAX	0.93
Athens	**1.07**	NASDAQ	**1.42**
BP	**1.07**	Next	**1.18**
Cadbury	0.88	NIKEI	**1.15**
Carnival	0.36	NYSE	0.88
DJIA	**1.07**	RBS	0.75
Easyjet	**2.49**	Schroders	**1.60**
Hammerson	0.71	Sky	**1.24**
HSI	**1.86**	Tesco	**1.94**
Imp	**1.65**	Vodafone	0.62

$$AARR = \frac{1}{N}\sum_{i=1}^{N_+} ARR_i \tag{10}$$

According to Table 9, EDDIE 9 has done quite well. In 13 out of the 20 datasets tested, it achieved a profit ($AARR > 1$). Overall, the algorithm achieved an AARR equal to 1.2205, which indicates that an investor would make an average annual return of 22% by using EDDIE 9.

5.5 Discussion

From the results presented in this section, we can draw several conclusions. First of all, it is apparent that the introduction of discretisation and sequential covering has improved the average performance of the EDDIE algorithm. As we observed in Tables 2 and 3, our new version EDDIE 9 ranked first in Fitness, RC, and RMC. The only metric that EDDIE 9 had a lower performance was RF, although results were not significant at 5% level. On the other hand, the improvements for RMC were statistically significant. This is a very important result, because improving the RMC means that the algorithm is able to identify more buy opportunities. Therefore, thanks to EDDIE 9 an investor increases its chances to make profit by reducing the number of missed buy opportunities.

In addition, EDDIE 9 achieved positive results when compared to the well-known C4.5 and RIPPER algorithms in terms of RC, RMC and RF, as shown in Tables 4 and 5. Although the improvements in RC are not significant, EDDIE 9 outperforms both C4.5 and RIPPER in terms of RMC and the differences are statistically significant at the 5% level. Hence, if an investor needed to choose which algorithm to use, the best option would be EDDIE 9.

As explained, the above average results are given as a measure of expected behaviour of the given algorithms. However, in real-world applications such as the one of financial forecasting, it is even more important to have good performance in terms of the best model. As we mentioned earlier, in a real scenario, an investor would decide which model to use after running the algorithm multiple times and then selecting the best performing model in the training datasets. The best results, which were presented in Tables 6 and 7, re-confirmed EDDIE 9's strengths. EDDIE 9 outperformed both C4.5 and RIPPER with statistically significant differences on RMC and also outperform RIPPER with statistically significant differences on RC. An investor, who would use the best model after multiple runs, would be able to get the best performance in all three metrics (RC, RMC, RF) when compared to C4.5 and RIPPER. Hence, in a real scenario, EDDIE 9 would be the best classification algorithm among the well-know C4.5 and RIPPER.

Lastly, our results in Table 9 showed that EDDIE 9 is not simply a good classification algorithm, but also a profitable one. As we mentioned in the previous section, for the 20 datasets tested in this paper, EDDIE 9 would return an annual profit of 22% on average.

6 Conclusion

To conclude, this paper presented work on the application of heuristic processes into the GP financial forecasting algorithm named EDDIE 9. Our proposal included two heuristics: (i) a sequential covering strategy to iteratively build a solution in combination with the GP search, and (ii) the use of an entropy-based dynamic discretisation procedure of numeric values. The sequential covering strategy allowed for the automated decomposition of the original problem into smaller subproblems. In addition, the dynamic discretisation process allowed for the deterministic calculation of threshold values that best fit the data.

Computational experiments showed that EDDIE 9 achieved positive results when compared to all three previous EDDIE versions, as well as C4.5 and RIPPER, in the majority of our test cases. More specifically, EDDIE 9 achieved first rank in terms of average Fitness, Rate of Correctness and Rate of Missing Chances against both the previous EDDIE versions. In addition, EDDIE 9 achieved first rank in all metrics (including Rate of Failure) in terms of best results. Moreover, EDDIE 9 showed remarkable improvements in the Rate of Missing Chances, outperforming both the previous EDDIE versions and well-known C4.5 and RIPPER with statistically significantly differences at the 5% level, which effectively allows investors to identify more buy opportunities, and thus increase their profit opportunities. This profitability was confirmed when we looked into the average annualised return rates across the 20 datasets, where we found that EDDIE 9 was not only proven to be a competitive classification algorithm, but also a profitable one, as it yielded an annualised average return of 22%.

There are several interesting directions for future research. First, it would be interesting to evaluate the use of a pruning procedure to potentially improve the solutions of the GP. This could reduce the size of the overall solutions and also improve their predictive performance. Second, the use of different discretisation procedures can improve the generation of the indicators' tests. Exploring the use of an heuristic to reorder the individual solutions—e.g., rank them based on their quality—to improve the quality of the final solution is a direction worth further exploration.

References

Abdelmalek W, Hamida S, Abid F (2009) Selecting the best forecasting-implied volatility model using genetic programming. Journal of Applied Mathematics and Decision Sciences, vol. 2009, Article ID 179230, 19 pages

Abdou H (2009) Genetic programming for credit scoring: The case of egyptian public sector banks. Expert Systems with Applications 36(9):11,402–11,417

Agapitos A, O'Neill M, Brabazon A (2010) Evolutionary learning of technical trading rules without data-mining bias. In: Schaefer R, Cotta C, Kołodziej J, Rudolph G (eds) Parallel Problem Solving from Nature – PPSN XI, Springer, Lecture Notes in Computer Science, vol 6238, pp 294–303

Allen F, Karjalainen R (1999) Using genetic algorithms to find technical trading rules. Journal of Financial Economics 51:245–271

Austin M, Bates G, Dempster M, Leemans V, Williams S (2004) Adaptive systems for foreign exchange trading. Quantitative Finance 4(4):37–45

Backus J (1959) The syntax and semantics of the proposed international algebraic language of Zurich. In: International Conference on Information Processing, UNESCO, pp 125–132

Binner J, Kendall G, Chen SH (eds) (2004) Applications of Artificial Intelligence in Finance and Economics, Advances in Econometrics, vol 19. Elsevier

Brookhouse J, Otero F, Kampouridis M (2014) Working with OpenCL to speed up a genetic programming financial forecasting algorithm: Initial results. In: Wagner, S. and Affeneller, M. (eds) GECCO 2014 Workshop on Evolutionary Computation Software Systems (EvoSoft), pp.1117–1124

Chen SH (2002) Genetic Algorithms and Genetic Programming in Computational Finance. Springer-Verlag New York, LLC

Cohen W (1995) Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning, Morgan Kaufmann, pp 115–123

Demšar J (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7:1–30

Edwards R, Magee J (1992) Technical analysis of stock trends. New York Institute of Finance

Fayyad U, Piatetsky-Shapiro G, Smith P (1996) From data mining to knowledge discovery: an overview. In: Advances in Knowledge Discovery & Data Mining, MIT Press, pp 1–34

García S, Herrera F (2008) An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. Journal of Machine Learning Research 9:2677–2694

Giacobini M, Provero P, Vanneschi L, Mauri G (2014) Towards the use of genetic programming for the prediction of survival in cancer. In: Cagnoni S, Mirolli M, Villani M (eds) Evolution, Complexity and Artificial Life, Springer Berlin Heidelberg, pp 177–192

Hu Y (1998) Constructive induction: Covering attribute spectrum. Feature Extraction Construction and Selection pp 257–272

Kampouridis M, Otero F (2013) Using attribute construction to improve the predictability of a GP financial forecasting algorithm. In: Proceedings of the Conference on Technologies and Applications of Artificial Intelligence, IEEE Xplore, pp 55–60

Kampouridis M, Tsang E (2010) EDDIE for investment opportunities forecasting: Extending the search space of the GP. In: Proceedings of the IEEE World Congress on Computational Intelligence, Barcelona, Spain, pp 2019–2026

Kampouridis M, Tsang E (2012) Investment opportunities forecasting: Extending the grammar of a gp-based tool. International Journal of Computational Intelligence Systems 5(3):530–541

Koza J (1992) Genetic Programming: On the programming of computers by means of natural selection. Cambridge, MA: MIT Press

Krawiec K (2002) Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genetic Programming and Evolvable Machines 3(4):329–343

Li J (2001) FGP: A genetic programming-ased financial forecasting tool. PhD thesis, Department of Computer Science, University of Essex

Martinez-Jaramillo S (2007) Artificial financial markets: An agent-based approach to reproduce stylized facts and to study the red queen effect. PhD thesis, CFFEA, University of Essex

Otero F, Silva M, Freitas A, Nievola J (2003) Genetic programming for attribute construction in data mining. In: Proc. of EuroGP, LNCS 2610, pp 384–393

Otero F, Freitas A, Johnson C (2008) cAnt-Miner: an ant colony classification algorithm to cope with continuous attributes. In: Ant Colony Optimization and Swarm Intelligence (Proc. ANTS 2008), pp 48–59

Otero F, Freitas A, Johnson C (2013) A New Sequential Covering Strategy for Inducing Classification Rules With Ant Colony Algorithms. IEEE Transactions on Evolutionary Computation 17(1):64–76

Otero FEB, Johnson CG (2013) Automated problem decomposition for the boolean domain with genetic programming. In: Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013, Vienna, Austria, pp 169–180

Phua C, Lee V, Smith K, Gayler R (2010) A Comprehensive Survey of Data Mining-based Fraud Detection Research. http://www.bsys.monash.edu.au/people/cphua/

Piatetsky-Shapiro G, Frawley W (1991) Knowledge Discovery in Databases. AAAI Press

Poli R, Langdon W, McPhee N (2008) A Field Guide to Genetic Programming. Lulu.com

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA

Dos Santos J, Ferreira C, Da S Torres R, Gonçalves M, Lamparelli R (2011) A relevance feedback method based on genetic programming for classification of remote sensing images. Information Sciences 181(13):2671 – 2684

Tsang E, Martinez-Jaramillo S (2004) Computational finance. IEEE Computational Intelligence Society Newsletter pp 3–8

Tsang E, Li J, Markose S, Er H, Salhi A, Iori G (2000) EDDIE in financial decision making. Journal of Management and Economics 4(4) (online)

Tsang E, Markose S, Er H (2005) Chance discovery in stock index option and future arbitrage. New Mathematics and Natural Computation, World Scientific 1(3):435–447

Wang P, Tsang E, Weise T, Tang K, Yao X (2010) Using GP to evolve decision rules for classification in financial data sets. In: Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on, pp 720 –727

Wilson G, Banzhaf W (2010) Fast and effective predictability filters for stock price series using linear genetic programming. In: Evolutionary Computation (CEC), 2010 IEEE Congress on, pp 1–8, DOI 10.1109/CEC.2010.5586297

Witten H, Frank E (2005) Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann