A Comparative Study on the Use of Classification Algorithms in Financial Forecasting

Fernando E. B. Otero and Michael Kampouridis

School of Computing
University of Kent, Chatham Maritime, UK
{F.E.B.Otero,M.Kampouridis}@kent.ac.uk

Abstract. Financial forecasting is a vital area in computational finance, where several studies have taken place over the years. One way of viewing financial forecasting is as a classification problem, where the goal is to find a model that represents the predictive relationships between predictor attribute values and class attribute values. In this paper we present a comparative study between two bio-inspired classification algorithms, a genetic programming algorithm especially designed for financial forecasting, and an ant colony optimization one, which is designed for classification problems. In addition, we compare the above algorithms with two other state-of-the-art classification algorithms, namely C4.5 and RIPPER. Results show that the ant colony optimization classification algorithm is very successful, significantly outperforming all other algorithms in the given classification problems, which provides insights for improving the design of specific financial forecasting algorithms.

Keywords: financial forecasting, classification, genetic programming, ant colony optimization

1 Introduction

Financial forecasting is a vital area in computational finance [13]. There are numerous works that attempt to forecast the future price movements of a stock; several examples can be found in [1].

In this study, we approach the financial forecasting problem as a classification problem [5, 14]. In a classification problem, the aim is to create a model that places objects (examples) into pre-defined categories (classes). The model is able to determine the category of an object by analysing patterns (attribute-values) between objects of that category. Classification problems can therefore be viewed as optimisation problems, where the goal is to find the best model that represents the predictive relationships in the data. Classification algorithms can be grouped by the type of the model representation that they produce: as 'black-box' models (e.g., the models produced by support vector machines using kernels and artificial neural networks, which are difficult to interpret); and 'white-box' models (e.g., decision tree and classification rule models, which are more readily interpreted). Since 'white-box' models have the advantage of being easier to interpret, they can be used to further understand the data—i.e., they can be used to understand how the predictions are made by the model. This enhanced understanding leads to a greater degree of trust in the models produced, which is crucial in various domains—e.g., medical and financial domains, where the predictions usually need to be validated by doctors/experts.

In this paper, *our goal is to report and analyse the performance of such 'white-box' models, one from the domain of genetic programming (GP) [8], and one from the field of ant colony optimization (ACO) [3].* We also compare the performance of these two algorithms with two other state-of-the-art algorithms, namely C4.5 and RIPPER. For the purposes of the GP, we will be using EDDIE [7], an algorithm especially designed for classification financial forecasting problems. For the purposes of ACO, we will be using the Unordered cAnt-Miner$_{\text{PB}}$ [10], which is a classification algorithm.

The remainder of this paper is organised as follows. In Section 2 we discuss the financial forecasting problem and how it can be viewed as a classification problem. Section 3 presents the research goals of this study. The datasets used in our experiments are described in Section 4, while the algorithms are described in Section 5. Section 6 presents and discusses the computational results. Finally, Section 7 concludes this paper.

2 Problem Description

2.1 Financial Forecasting

One of the most common methods used in the financial forecasting area is technical analysis [4]. This method assumes that patterns exist in historical data and that these patterns will repeat themselves. Consequently, it is worth identifying these patterns, so that we can exploit them in the future and make profit. As part of technical analysis, several indicators are used. These technical analysis indicators are formulas that measure different aspects of a given financial dataset, such as trend, volatility and momentum.

An example of such indicators is the Moving Average (MA), which calculates the averages of a given dataset under sliding windows of a fixed length L. One way of using MAs is to compare a short-term MA with a long-term one; for instance, when the short-term MA goes above the long-term one, this would indicate that the market is in an upward trend and thus it would be a good indication to buy.

The above method has been extensively used in both the literature and in the industry. However, what is evident is that both academics and people who work in the industry tend to use very specific L lengths for the indicators; for instance, 20 days is a common short-length period and 50 days is a common long-term period. In [7], it was argued that this method is not very flexible and cannot guarantee that specific pre-specified indicators are necessarily the best ones. For example, nobody can guarantee that a 20 days MA is definitely more effective than a 25 days MA, under all possible datasets.

In our current work, we will be building on the above idea by allowing all algorithms tested in this paper to use any period lengths within a parameterised length $[MinP, MaxP]$, which is set to $[2, 65]$ days. This will add flexibility to all algorithms and enable them to create new indicators. However, one has to keep in mind that this will also add to the complexity of the problem. While traditionally a financial forecasting algorithm would have to deal with a low number of indicators, e.g. 6 technical indicators with two period lengths—a short (20 days) and a long (50 days) term—leading to a total number of 12 indicators, in our framework each algorithm would have to choose among $6 \times 64 = 384$ indicators. Hence, the problem is much more complex than the traditional classification financial forecasting problems, as the number of combinations for all available indicators is much higher.

2.2 The Classification Problem

Each algorithm tested in this work will be attempting to answer the question 'Will the price of the X stock rise by $r\%$ within the next n days?' Thus, the classes 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 $r\%$. For this set of experiments, n is set to 20 and r to 4%. In other words, the algorithms will be trying to forecast whether the daily closing price is going to increase by 4% within the following 20 days.

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

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

It should also be noted that we chose to use RMC and RF instead of the more 'traditional' classification Recall and Precision measures, because they are metrics related to the financial forecasting problem. Thus, it would make more sense to an investor to know how many times an algorithm returns FP predictions, because this could have a very negative impact on his portfolio.

3 Research Goals

The goal of this study is to analyse the performance of state-of-the-art classification algorithms when applied to financial forecasting, and compare it to two well-known bio-inspired algorithms, namely EDDIE and Unordered *c*Ant-Miner$_{\text{PB}}$. EDDIE is a GP financial forecasting algorithm, as it was designed specifically for the purposes of financial forecasting and that has been previously successfully applied to datasets from different international markets. The Unordered *c*Ant-Miner$_{\text{PB}}$ algorithm is an established ant colony optimization (ACO) [3] classification algorithm which has been found efficient in discovering comprehensible and accurate classification models [10].

As we discussed in Section 2.2, financial forecasting can be modelled as a classification problem. Thus, we can summarise the goals of this study as follows:

- How do EDDIE and Unordered *c*Ant-Miner$_{\text{PB}}$ compare to state-of-the-art classification algorithms in terms of Rate of Correctness, Rate of Missing Chances and Rate of Failure?
- Can we get insights on the differences in the search behaviour between GP and ACO, based on their performance as classification algorithms in the given financial forecasting problem?

4 Data Preparation

The set of data used in this work is composed of three parts: (i) daily closing price of a stock, (ii) a number of attributes, and (iii) signals. Stocks' daily closing prices can be obtained online on websites such as `http://finance.yahoo.com` and also from financial statistics databases like *Datastream*.[1] The attributes are indicators commonly used in technical analysis [4]; which indicators to use depends on the user and his belief of their relevance to the prediction. The technical indicators that are used in this work are: Moving Average (MA), Trade Break Out (TBR), Filter (FLR), Volatility (Vol), Momentum (Mom), and Momentum Moving Average (MomMA).[2] Also, as already explained in Section 2, 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 $r\%$.

As we are approaching the financial forecasting problem as a classification problem, the training set is created using the daily closing price of a stock for a fixed time window. For each training day (each example in our training set), we calculate the aforementioned technical indicators MA, TBR, FLR, Vol, Mom and MomMA for the periods from 2 to 65 days—the predictor attributes values of the problem. There are 384 predictor attribute in total: 6 indicators and 64

[1] Available at: `http://thomsonreuters.com/datastream-professional/`

[2] We use these indicators because they have been proved to be quite useful in developing decision trees in [9]. Of course, there is no reason not to 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.

```
<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. 1. The Backus Normal Form of EDDIE 8.

possible periods ($6 \times 64 = 384$). We also calculate the signals for each of the training days by 'looking ahead' if the value of the stock increased by $r\%$ or not (0=no increase, 1=increase)—the class attribute values of the problem. The test set is created in a similar fashion, with the difference that the test set is used only to evaluate the performance of the algorithm and the algorithms have no access to the test set during training.

Using the above procedure, we created 25 datasets. These datasets consist of daily closing prices from 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 Scotlland (RBS), Schroders, Sky, Tesco, Vodafone and Xstrata. The 7 indices are: Athens Stock Exchange (Greece), DJIA (USA), HSI (Hong Kong), MDAX (Germany), and NASDAQ (USA), NIKEI (Japan), and NYSE (USA). For each of the datasets, the training set is 1000 days and the test set 300.

5 Algorithms

5.1 EDDIE

EDDIE 8 is a Genetic Programming (GP) [8] financial forecasting algorithm, which learns and extracts knowledge from a set of data. After feeding the data to the system, EDDIE creates and evolves a population of decision trees. Figure 1 presents the Backus Normal Form (BNF) (grammar) of EDDIE 8. 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 *threshold*), or a logic operator (`AND`, `OR`, `NOT`), which can hold multiple boolean conditions. The `Then` and `Else` branches can be a new tree, or a decision, to buy or not-to-buy (denoted by `1` and `0`). The *threshold* is a real number, which is randomly generated within the range of the minimum and maximum values of the respective indicator.

As we can observe from the grammar in Figure 1, there is a function called *'VarConstructor'*, which takes two children. The first one is the indicator, and the

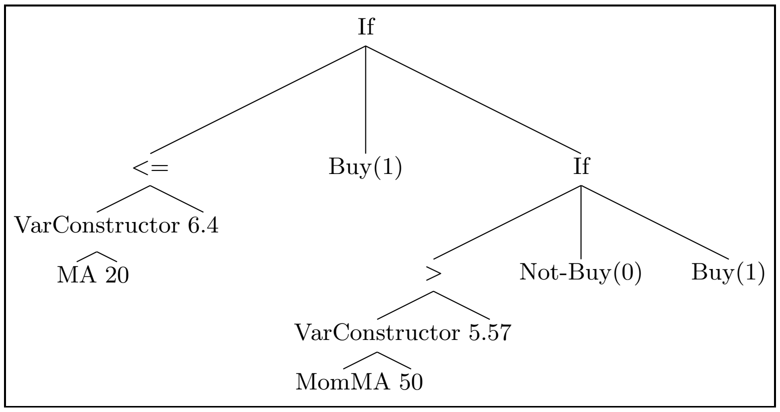

Fig. 2. Sample decision tree generated by EDDIE 8. As we can see, if the *20 days MA* is less than or equal to *6.4*, then the user is advised to buy; otherwise, the user is advised to consult another tree, which is located in the third branch ('else-branch') of the tree. This tree checks if the 50 days Momentum Moving Average is greater than 5.57; if it is, it advises to not-buy, otherwise to buy.

second one is the 'Period'. 'Period' is an integer within the parameterized range [MinP, MaxP] that the user specifies. The advantage of this approach is that it makes the GP more dynamic, as EDDIE 8 is not constrained to pre-specified periods, as is usually the case in literature and industry.As a consequence, it is up to the GP and the evolutionary process to look for the optimal periods values from the period range provided. For instance, if this range is 2 to 65 days, then EDDIE 8 can create Moving Averages with any of these periods, e.g., 20 days MA, 25 days MA, and so on. Furthermore, the periods are leaf nodes and are thus subject to genetic operators, such as crossover and mutation. A sample tree of EDDIE 8 is presented in Figure 2. The periods 20 and 50 of the figure's sample tree are leaf nodes; the advantage of this being that the GP can replace them with more effective periods, which might have come up during the evolutionary process.

EDDIE's fitness function is a weighted formula, combining Equations 1-3:

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

where w_1, w_2 and w_3 are the weights for RC, and the financial-based metrics RMC and RF. These weights are given in order to reflect the preferences of investors. For instance, a conservative investor would want to avoid failure; thus a higher weight for RF should be used. For the experiments in this paper, the focus is on strategies that mainly target correctness and reduced failure.

5.2 Unordered *c*Ant-Miner$_{\text{PB}}$

Unordered *c*Ant-Miner$_{\text{PB}}$ [10] is an ACO classification algorithm that employs an improved sequential covering strategy [11] to search for the best set of classifica-

```
IF MA_20 <= 6.4 THEN Buy(1)
IF MA_20 > 6.4 AND MM_50 > 5.57 THEN Buy(0)
IF MA_20 > 6.4 AND MM_50 <= 5.57 THEN Buy(1)
```

Fig. 3. Sample of a set of classification rules representing the same conditions of EDDIE's tree from Figure 2.

tion rules. The use of this improved strategy avoids the potential problem of rule interaction arising from the greedy nature of the sequential covering commonly used by rule induction algorithms. Instead of creating a rule and then determine its consequent (the prediction of the rule) based on the majority class value of the covered training examples, as the majority of ACO classification algorithms, an ant creates rules for each class value in turn using as negative examples all the examples associated with different class values. The advantage of discovering a set of classification rules (unordered rules) is that the order in which the rules are discovered is not important to the interpretation of the individual rules, in contrast to rule induction algorithms that discover a list of classification rules (ordered rules). A sample of a set of classification rules is presented in Figure 3.

In summary, the Unordered cAnt-Miner$_{\mathrm{PB}}$ works as follows. Each ant starts with an empty set of rules and iteratively adds a new rule to this set. In order to create a rule, an ants adds one term at a time to the rule antecedent by choosing terms to be added to the current partial rule based on their values of the amount of pheromone (τ) and a problem-dependent heuristic information (η). After a rule is created and pruned, it is added to current set of rules and the training examples correctly covered by the rule are removed. At the end of an iteration, when all ants have created a set of rules, the best set of rules (determined by the highest predictive accuracy on the training data) is used to update pheromone values, providing a positive feedback on the terms present in the rules—the higher the pheromone value of a term, the more likely it will be chosen to create a rule. This iterative process is repeated until a maximum number of iterations is reached or until the algorithm stagnates.

Since the consequent of a rule is fixed during its creation, the Unordered cAnt-Miner$_{\mathrm{PB}}$ uses a class-specific heuristic information[3] and dynamic discretisation procedure of continuous values. Hence, when a particular indicator value is selected (e.g., *20 days MA*), the algorithm determines the threshold value by looking at the training data and selecting the one that has the highest frequency of training cases of the current selected class, instead of choosing them in a random way as EDDIE does. Therefore, the selection of threshold values in the Unordered cAnt-Miner$_{\mathrm{PB}}$ algorithm is tailored for the current available data—only suitable values are considered as candidate threshold values.

[3] The heuristic information represents a priori information about the quality of the candidate components of the antecedent of a rule and it is used in the stochastic rule construction process. The Unordered cAnt-Miner$_{\mathrm{PB}}$ uses the frequency of training cases with respect to a specific class value as the heuristic information.

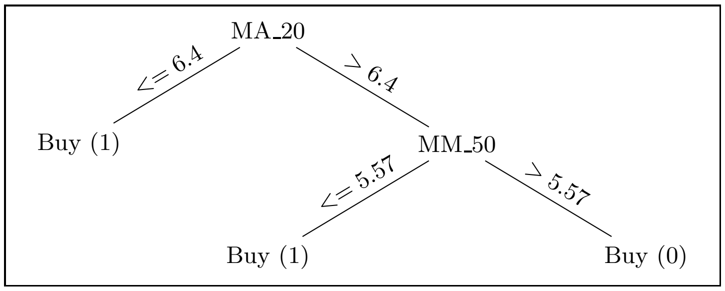

Fig. 4. Sample decision tree generated by C4.5 representing the same conditions of EDDIE's tree from Figure 2.

5.3 C4.5 (J48)

J48 is Weka's implementation [14] of the well-known C4.5 algorithm [12]. The C4.5 algorithm, probably the most known decision tree induction algorithm, employs an entropy-based criterion in order to select the best attribute to create a node. C4.5 has been successfully applied to a wide range of classification problems and it is usually used on evaluative comparisons of new classification algorithms.

5.4 RIPPER (JRip)

JRip is Weka's implementation [14] of the RIPPER algorithm [2]. RIPPER sequentially creates a set of rules that is subject to a global post-processing step by implementing a rule induction procedure with a reduced error pruning strategy [12]. The final set of rules created by JRip has a similar structure than the one created by the Unordered cAnt-Miner$_{\text{PB}}$ algorithm (Figure 3).

6 Results

The experiments were carried out using 25 datasets created using the procedure described in Section 4. The parameters used for both stochastic algorithms are: EDDIE 8 {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, $w_1 = 0.6$, $w_2 = 0.1$, $w_3 = 0.3$} and Unordered cAnt-Miner$_{\text{PB}}$ {colony size = 5, evaporation factor = 0.90, minimum number of examples = 10}. The other algorithms were used with the default values proposed by their correspondent authors, which typically represent robust values that work well across different datasets. We did not attempt to optimise the parameters to individual datasets. Since both EDDIE 8 and Unordered cAnt-Miner$_{\text{PB}}$ are stochastic algorithms, each algorithm is run 50 times for each dataset; C4.5 and RIPPER are deterministic algorithms, hence they are run once per dataset.

Table 1. Summary of the results concerning the rate of correctness (RC). The best RC value for a given dataset is in bold.

	EDDIE 8	U-cAnt-Miner$_{\text{PB}}$	C4.5	RIPPER
Aggreko	0.549	**0.594**	0.573	0.450
Amlin	0.518	**0.523**	0.453	0.467
Athens	**0.535**	0.515	0.467	0.413
Barclays	**0.549**	0.533	0.503	0.470
BP	0.533	**0.534**	0.467	0.510
Cadbury	**0.654**	0.625	0.527	0.553
Carnival	0.506	0.500	**0.587**	0.503
DJI	0.653	**0.696**	0.670	0.667
Easyjet	0.451	**0.564**	0.540	0.550
First	0.509	0.506	**0.593**	0.577
Hammerson	0.533	0.524	0.507	**0.593**
HIS	0.603	**0.687**	0.573	0.627
Imperial Tobacco	0.566	**0.630**	0.593	0.567
Marks & Spencer	0.500	0.572	0.527	**0.660**
MDAX	0.492	0.499	**0.503**	0.487
NASDAQ	0.549	**0.594**	0.573	0.450
Next	0.465	**0.595**	0.470	0.450
NIKEI	0.516	0.556	**0.617**	0.380
NYSE	0.546	**0.564**	0.563	0.500
RBS	0.529	0.569	0.567	**0.493**
Schroders	0.570	0.602	0.493	**0.643**
Sky	0.516	**0.599**	0.583	0.427
Tesco	0.613	**0.621**	0.587	0.600
Vodafone	0.501	0.555	**0.633**	0.450
Xstrata	0.607	**0.685**	0.677	0.590

The results of our experiments are presented in Table 1 for Rate of Correctness; Table 2 for Rate of Missing Chances; and Table 3 for Rate of Failure. For EDDIE 8 and Unordered cAnt-Miner$_{\text{PB}}$ algorithms, a value on those tables corresponds to the average value measured over the 50 runs of the algorithm. Table 4 presents the results of the non-parametric Friedman statistical test with the post-hoc Hommel's test [6]. The information presented in Table 4 corresponds to the average rank (first column), where the lower the rank the better the algorithm's performance, and the adjusted p_{Homm} value. Statistically significant differences amongst the algorithm with the best rank (the control '(c)' algorithm) are determined by the p_{Homm} value: if the p value is less than 0.05, the difference in the rank is statistically significant at the $\alpha = 0.05$ level—i.e., the algorithm with the best rank significantly outperforms the other algorithm.

Our results showed that the Unordered cAnt-Miner$_{\text{PB}}$ algorithm achieved the best average rank in terms of both Rate of Correctness (RC) and Rate of Missing Chances (RMC), outperforming all the other algorithms with statistically significant differences. It achieved the best RC value in 13 out of the 25 datasets and the best RMC value in 23 out of 25 datasets. Both C4.5 and EDDIE 8 achieved similar results for RC and RMC, while RIPPER was the worst performing algorithm. The results for Rate of Failure (RF) did not show any significant differences between the algorithms; all algorithms achieved similar average ranks.

The results provide interesting insights of the strengths the Unordered cAnt-Miner$_{\text{PB}}$ algorithm when compared to EDDIE. Both algorithms are stochastic

Table 2. Summary of the results concerning the rate of missing changes (RMC). The best RMC value for a given dataset is in bold.

	EDDIE 8	U-*c*Ant-Miner$_{\text{PB}}$	C4.5	RIPPER
Aggreko	0.283	**0.164**	0.426	0.427
Amlin	0.429	**0.294**	0.457	0.531
Athens	0.252	**0.159**	0.276	0.394
Barclays	0.444	**0.355**	0.500	0.464
BP	0.421	**0.346**	0.474	0.438
Cadbury	0.166	**0.145**	0.350	0.433
Carnival	0.153	**0.142**	0.239	0.337
DJI	0.161	**0.013**	0.062	0.118
Easyjet	0.458	0.668	**0.404**	0.498
First	0.530	0.498	**0.403**	0.461
Hammerson	0.382	**0.233**	0.331	0.396
HIS	0.267	**0.002**	0.248	0.150
Imperial Tobacco	0.449	**0.182**	0.355	0.452
Marks & Spencer	0.474	0.263	0.435	**0.253**
MDAX	0.204	**0.031**	0.397	0.493
NASDAQ	0.416	**0.267**	0.396	0.584
Next	0.477	**0.206**	0.482	0.568
NIKEI	0.300	**0.008**	0.062	0.677
NYSE	0.207	**0.157**	0.364	0.525
RBS	0.373	**0.281**	0.332	0.370
Schroders	0.328	**0.065**	0.276	0.309
Sky	0.469	**0.317**	0.497	0.626
Tesco	0.287	**0.231**	0.429	0.278
Vodafone	0.303	**0.231**	0.261	0.380
Xstrata	0.259	**0.044**	0.122	0.244

Table 3. Summary of the results concerning the rate of failure (RF). The best RF value for a given dataset is in bold.

	EDDIE 8	U-*c*Ant-Miner$_{\text{PB}}$	C4.5	RIPPER
Aggreko	0.514	0.522	**0.509**	0.508
Amlin	**0.405**	0.427	0.469	0.450
Athens	**0.533**	0.539	0.576	0.621
Barclays	**0.397**	0.431	0.443	0.480
BP	**0.337**	0.363	0.400	0.363
Cadbury	**0.331**	0.359	0.403	0.354
Carnival	0.631	0.631	**0.593**	0.659
DJI	**0.285**	0.298	0.303	0.287
Easyjet	0.305	0.284	**0.266**	0.276
First	0.285	0.305	0.241	**0.224**
Hammerson	0.427	0.444	0.449	**0.350**
HIS	**0.293**	0.312	0.332	0.316
Imperial Tobacco	**0.293**	0.335	0.318	0.311
Marks & Spencer	0.388	0.365	0.367	**0.284**
MDAX	0.513	**0.507**	0.508	0.526
NASDAQ	0.305	0.314	**0.282**	0.359
Next	0.392	**0.337**	0.380	0.377
NIKEI	0.452	**0.410**	0.597	0.477
NYSE	0.445	0.435	**0.411**	0.458
RBS	0.386	0.369	**0.359**	0.420
Schroders	0.365	0.388	0.438	**0.290**
Sky	0.335	0.304	**0.222**	0.407
Tesco	0.284	0.295	**0.235**	0.299
Vodafone	0.514	0.480	**0.410**	0.558
Xstrata	0.281	0.295	**0.275**	0.303

Table 4. 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) Rate of Correctness		
U-*c*Ant-Miner$_{\mathrm{PB}}$ (c)	1.76	–
C4.5	**2.52**	**0.0374**
EDDIE 8	**2.72**	**0.0171**
RIPPER	**3.00**	**0.0020**
(ii) Rate of Missing Chances		
U-*c*Ant-Miner$_{\mathrm{PB}}$ (c)	1.12	–
C4.5	**2.68**	**5.1E-9**
EDDIE 8	**2.88**	**2.8E-6**
RIPPER	**3.32**	**1.9E-5**
(iii) Rate of Failure		
EDDIE 8 (c)	2.26	–
C4.5	2.34	0.8265
U-*c*Ant-Miner$_{\mathrm{PB}}$	2.52	0.8265
RIPPER	2.88	0.2685

algorithms, which can perform a more global search than the deterministic C4.5 and RIPPER algorithms. While EDDIE is a GP algorithm tailored for financial forecasting, it does not use heuristics to guide the search. In this sense, the search of the GP is 'blind', using the training examples just as an oracle: it can query the quality of a solution on the training data, but it does not 'look' into the training data to shape the solutions. The Unordered *c*Ant-Miner$_{\mathrm{PB}}$ is an ACO algorithm, which uses the class frequency of the predictor attributes as heuristic information in the rule construction process. It also incorporates a dynamic discretisation procedure to find threshold values to create the indicator conditions (e.g., `MA_20 < 6.4`); and the consequent (decision) of the rules are fixed during creation, so the conditions selected to create the antecedent are relevant for this particular consequent. In summary, it uses more information available from the training data to guide the search.

Given our findings, we hypothesise that in order to improve EDDIE's performance, there is a need to incorporate more background information to the GP search. While the creation of the conditions can be left to the GP, the choice of the value of the decision can be determined using the training data. A similar procedure can be applied to the selection of threshold values, where instead of selecting values at random, the values can be selected based on the training data. In this way, the GP search is focused in finding a good structure to represent the conditions, and there is less pressure for the GP to find the correct values to be used as the decision and threshold.

7 Conclusion

In this paper we presented a comparative study of different classification algorithms in financial forecasting. Our first research goal was to analyse the performance of state-of-the-art classification algorithms and compare their performance to two well-known bio-inspired algorithms: EDDIE, a GP algorithm designed specifically for financial forecasting; and Unordered *c*Ant-Miner$_{\mathrm{PB}}$ (U-*c*Ant-Miner$_{\mathrm{PB}}$), an ACO algorithm designed for classification. Our results show that the U-*c*Ant-Miner$_{\mathrm{PB}}$ algorithm was significantly better than all the other algorithms tested. Regarding our second research goal, we identified potential strengths of the U-*c*Ant-Miner$_{\mathrm{PB}}$ search, when compared to EDDIE: the use of a data-driven procedures to determine the prediction of the classification rules and to determine threshold values for the indicator conditions.

As a future research direction, we aim to investigate whether heuristics procedures, such as the ones used by U-*c*Ant-Miner$_{\mathrm{PB}}$, can be applied to GP algorithms, like EDDIE, and lead to improved solutions. Our hope is that the GP's search can benefit from using such data-driven processes, and select the thresholds and decision values in a more sophisticated manner.

References

1. Chen, S.H.: Genetic Algorithms and Genetic Programming in Computational Finance. Springer-Verlag New York, LLC (2002)
2. Cohen, W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning. pp. 115–123. Morgan Kaufmann (1995)
3. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press (2004), 328 pages
4. Edwards, R., Magee, J.: Technical analysis of stock trends. NYIF (1992)
5. Fayyad, U., Piatetsky-Shapiro, G., Smith, P.: From data mining to knowledge discovery: an overview. In: Advances in Knowledge Discovery & Data Mining. pp. 1–34. MIT Press (1996)
6. García, S., Herrera, F.: An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. Journal of Machine Learning Research 9, 2677–2694 (2008)
7. Kampouridis, M., Tsang, E.: EDDIE for investment opportunities forecasting: Extending the search space of the GP. In: Proceedings of the IEEE World Congress on Computational Intelligence. pp. 2019–2026. Barcelona, Spain (2010)
8. Koza, J.: Genetic Programming: On the programming of computers by means of natural selection. Cambridge, MA: MIT Press (1992)
9. Martinez-Jaramillo, S.: Artificial Financial Markets: An agent-based Approach to Reproduce Stylized Facts and to study the Red Queen Effect. Ph.D. thesis, CF-FEA, University of Essex (2007)
10. Otero, F., Freitas, A.: Improving the Interpretability of Classification Rules Discovered by an Ant Colony Algorithm. In: Proceedings of the 2013 Genetic and Evolutionary Computation Conference. pp. 73–80. ACM Press (July 2013)
11. Otero, F., Freitas, A., Johnson, C.: A New Sequential Covering Strategy for Inducing Classification Rules With Ant Colony Algorithms. IEEE Transactions on Evolutionary Computation 17(1), 64–76 (2013)
12. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
13. Tsang, E., Martinez-Jaramillo, S.: Computational finance. IEEE Computational Intelligence Society Newsletter pp. 3–8 (2004)
14. Witten, H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2nd edn. (2005)