Chapter 1

EDDIE FOR FINANCIAL FORECASTING

E.P.K. Tsang
University of Essex
edward@essex.ac.uk

J. Li
University of Essex
jli@essex.ac.uk

Abstract EDDIE is a genetic-programming based system for channelling expert knowledge into forecasting. FGP-2 is an implementation of EDDIE for financial forecasting. The novelty of FGP-2 is that, as a forecasting tool, it provides the user with a handle for tuning the precision against the rate of missing opportunities. This allows the user to pick investment opportunities with greater confidence.

Keywords: Genetic programming, financial forecasting, precision, genetic decision trees

1. Introduction

1.1 The Forecasting Problem and Its Difficulties

In machine forecasting, one is often given a series of observations over a set of monitored variables$\{x_1, x_2, \cdots, x_n\}$, and asked to find the regularity in the data in order to predict the value of a dependent variable y. For example, given three years' of records of the daily closing prices, trade volumes, changes in interest rate, market indices, etc., one may attempt to predict whether a share price will rise or fall in the following week. This prediction task is difficult for many reasons. In our research, we focus on the following two:

1. Variables identification: Can the observed variables explain the dependent variables? In other words, is y determined by a function of the x_i's? Taking more variables into consideration may incur higher observation cost. Finding the right x_i's in a domain is a problem for the *human expert*. Can the expert be helped to do his/her job more efficiently?

2. Variable interactions: Even if we are sure that y is a function of the x_i's, how could this function be found? Many *machine learning techniques* are designed to address this problem. Real life forecasting problems are difficult because the x_i's are rarely independent of each other. For example, a company's share price may be affected by, among many other things, the interest rate and the company's sales volume; the sales volume could be affected by money supply, which is affected by the interest rate. Combinatorial explosion prevents one from examining combinations of all the factors and all possible interactions between them.

1.2 EDDIE - A Forecasting Tool

EDDIE (which stands for Evolutionary Dynamic Data Investment Evaluator) is an interactive tool, designed at University of Essex, to help analysts to search the space of interactions and make financial decisions [Tsang et al 1998]. Given a set of variables, EDDIE attempts to find interactions among variables and discover non-linear functions (addressing point 2 above). By using genetic programming, EDDIE generates decision trees, which can be understood by human experts. Human expertise is channelled into EDDIE through human feedback to the system. In this process, EDDIE helps the human expert to experiment with different variables (x_i's) more easily (addressing point 1 above).

FGP is an implementation of EDDIE for financial forecasting. FGP has been applied to a variety of financial forecasting problems with demonstrated accuracy [Li & Tsang 1999; Tsang et al 2000]. It generates Genetic Decision Trees (GDTs), which enables the program to explain how a forecast was arrived at. This allows the users to judge whether the reasons for the prediction are sound or not.

An example GDT generated for daily valuations of S&P 500 is shown below:

```
(IF (NOT (PMV_50 > −38.974922) )
 (IF (Vol_50 > 26.432180)
    (IF (AND (NOT (OR (TRB_50 > −124.212029) (Vol_50 < 42.150013) ))
             (AND (PMV_12 < 31.534222) (NOT (Filter_63 > 104.841507))));
    Buy ;
    Don't Buy );
 Buy );
 Don't Buy )
```

In this GDT, PMV_50, Vol_50 , PMV_12 and $Filter\_63$ are technical indicators (e.g. Vol_50 is "50 days volatility", measured by standard deviation of the last 50 day's closing prices). Details of these indicators will not be elaborated here as The main aim of this paper is to argue that EDDIE is a useful forecasting tool. Whether the rules generated are good or not depends on the quality of the input indicators, and this GDT was generated using low quality text-book indicators. [1]

In this paper, we focus on a novel feature of FGP-2, in that it provides the user with a handle for tuning the precision against the rate of missing opportunities, as it will be explained below.

2. Forecasting Performance Criteria

Prediction accuracy is naturally important to forecasting. In this section, we define precision and explain why it is important in some forecasting applications. As a forecasting tool, FGP-2 is designed to give the user control over the precision of a forecast.

Without lost of generality, we shall describe a specific forecasting task. We shall use it to describe the criteria for measuring forecasting success. Suppose one is asked to forecast whether an index will rise by $r\%$ within the next n periods. Each day can be classified into a *positive position*, where the target return will be achieved, or a *negative position*, where the target return will not be achieved. Given a prediction and the reality (in hindsight), one may construct a contingency table as shown in table 1.1. The following analysis applies to all two-class classification prediction problems in general.

Here we define some measures of success in forecasting. The *rate of correctness* (RC) in a prediction is the number of all correct predictions over the total number of predictions. The *rate of failure* (RF) is the proportion of positions that were wrongly predicted positive (FP) over the number of positive predictions (N+). The precision is 1 - RF, i.e. the proportion of positive positions that were correctly predicted. The rate of *missing chances* (RMC) is the number of wrongly predicted negative

Table 1.1. A contingency table for a two-class classification prediction problem

# of True Negative (TN)	# of False Positive (FP)	Actual Negative O_-=TN+FP
# of False Negative (FN)	# of True Positive (TP)	Actual Positive O_+=FN+TP
Predicted Negative N_-=TN+FN	Predicted Positive N_+=FP+TP	Total # of Predictions T=N_++N_-=O_++O_-

(FN) over the number of actual positives (O_+):

$$\mathbf{RC} = \frac{TP+TN}{T}; \quad \mathbf{RF}(1-precsion) = \frac{FP}{N_+}; \quad \mathbf{RMC} = \frac{FN}{O_+}$$

Ideally one would like RC to be 1. How close RC could approach 1 is limited by the nature of the data (e.g. are all the relevant variables used? is there noise?) as well as the quality of the forecasting algorithm.

In financial forecasting, a positive prediction may lead to investment. If this prediction is wrong, the investor will not be able to achieve the return desired. Such mistakes could be costly. Therefore, we assume that the user would want control over RF, if possible. (In other applications, it may be important not to miss any opportunities, hence reduce FN.)

It should be noted that RF may trivially be reduced to 0 if the system makes no recommendation to buy. However, a system that never recommends any "buy" will not be useful to any investor. Therefore, in reality, one would like to reduce RF (the principle goal) without significantly increasing RMC (which is seen as a constraint to the target forecasting system). FGP-2 was built with the following mission: *FPG-2 Mission: to enable the user to reduce RF by increasing RMC, or vice versa, without significantly affecting RC.* In other words, RC will not be used as the objective function here. The objective is to reduce RF. However, RC is considered by EDDIE as a constraint.

3. FGP-2 -Trading Precision with the Rate of Missing Opportunities

3.1 FGP-1: Brittle Results with a Linear Fitness Function

In a forecasting problem, RC is what one would like to improve. In previous papers, we have presented the effectiveness of FGP-1, an early implementation of EDDIE, in achieving reasonably high RC [Li & Tsang 1999; Tsang & Li 2000]. In the preceding section, we explained that RC may not be the only criterion for measuring forecasting performance. In some applications, one may want to reduce RF (or 1 - precision).

FGP-1 used $f_{(1)}$ (Equation (1.1)) as its fitness function. It allows the users to reflect their preference by means of adjusting the weights w_rc, w_rmc and w_rf.

$$f_{(1)} = w\_rc * RC - w\_rmc * RMC - w\_rf * RF \tag{1.1}$$

One possibility of achieving low RF is to assign a high value to w_rf in FGP-1, which we studied thoroughly at an early stage. We fixed w_rc to 1 (to prevent FGP-1 from reducing RF to 0 by generating rules that produce no "buy" signals) and w_rmc to 0. (We have tried an alternative, which is to set w_rmc to 1; no better results were found). We attempted to vary w_rf to a value between 0 and 1. Our experiments show that FGP-1's performance can be very sensitive to the three weights; in other words, performance of FGP-1 was brittle. This is elaborated below.

We found in our experiments that if the value that we choose for w_rf is too close to 1, FGP-1 achieved lower RF by making no positive recommendations at all; we shall refer to this as the "no-positive-prediction problem". If the value that we choose for w_rf is too low, the performance of FGP-1 is no different from FGP-1 that sets w_rf to 0; we shall refer to this as the "no-effect problem".

There is often some constant a (0.62 in our experiments) such that if the value of w_rf deviates slightly from a on either side, one of the above two problems occur. When w_rf was set to the fine-tuned critical value (0.62), FGP-1 did not generate effective decision trees reliably. In our experiments, only two out of ten runs generated a few correct positive positions on the test period; the remaining 8 runs either showed the no-positive-prediction or the no-effect problem.

According to our experience, the weighted fitness function is satisfactory. Firstly, it is likely that the critical value for w_rf varies from one data set to another. We found this value (0.62) for the test data set, but there is no guarantee that it will work for unseen data. Secondly, as explained above, even when w_rf is set to the fine tuned critical value, some of the GDTs generated suffered from the no-positive-prediction and some suffered from the no-effect problem. Therefore, if one picks one of these GDTs and apply it to unseen data, its performance is difficult to predict (because the sensitive w_rf that is good for the test data may or may not be good for the unseen data). Ideally, one would like a system that generates GDTs which performance is not too sensitive to parameter setting. Besides, one would hope that given one set of parameters, the system generates GDTs with similar (reliable) performance.

3.2 FGP-2: Putting Constraints into EDDIE

Our mission is to reduce RF, possibly at the price of RMC. To do so, we introduced a new parameter to FGP, $\Re$= [P_{min}, P_{max}], which defines the minimum and maximum percentage of recommendations that we instruct FGP to make in the training data (like most machine learning methods, the assumption is that the test data exhibits similar characteristics). We call the new fitness function $f_{(2)}$.

If one aims to make accurate forecasts for a given series, then choosing appropriate values for $\Re$ and the weights for $f_{(2)}$ remains a non-trivial task. In this paper, our focus is to first examine whether RF can be reduced by any choice of $\Re$. Then we shall see if RF can be adjusted without affecting the overall forecasting correctness (RC).

Efficacy of the constraint in fitness function is first demonstrated by the following experiment, where we took $\Re = [35\%, 50\%]$, $w\_rmc = 0$ and $w\_rf = 1$. We ran FGP 10 times. Results are showed in Table 1.2. With $f_{(2)}$, FGP-2 does not exhibit the brittleness in FGP-1, as demonstrated by the relatively small standard deviation (STD) in RC. For reference, we have included the AARR (Average Annualised Rate of Return) and RPR (Ratio of Positive Returns) in Table 1.2. RPR measures the proportion of times when FGP-2's recommendation gives a positive return, even when the target $r\%$ has not been achieved. Both AARR and RPR are for reference only, as they are not used to train FGP-2.

Table 1.2. FGP-2 results on test data using the constrained fitness function with $\Re = [35\%, 50\%]$

RULES	RF	RMC	RC	AARR	RPR	Number of Recommendations
GDT 1	40.34%	64.02%	53.92%	60.68%	70.59%	357
GDT 2	41.22%	62.67%	53.66%	63.83%	67.55%	376
GDT 3	40.12%	66.22%	53.66%	61.98%	70.96%	334
GDT 4	40.06%	66.39%	53.66%	62.60%	70.78%	332
GDT 5	40.25%	67.40%	53.39%	64.02%	69.66%	323
GDT 6	41.03%	59.46%	54.27%	58.26%	69.29%	407
GDT 7	41.47%	66.39%	52.95%	62.99%	67.35%	340
GDT 8	39.94%	68.75%	53.30%	63.98%	69.16%	308
GDT 9	39.82%	66.55%	53.74%	63.41%	71.73%	329
GDT 10	36.40%	69.59%	54.63%	72.02%	72.44%	283
MEAN	40.06%	65.74%	53.72%	63.38%	69.95%	338.9
STD	1.41%	2.99%	0.48%	3.53%	1.67%	34.7

Table 1.3. FGP results on test data using the general fitness function ($w\_rc = 1, w\_rmc = w\_rf = 0$)

RULES	RF	RMC	RC	AARR	RPR	Number of Recommendations
GDT 1	41.11%	44.59%	56.56%	57.82%	66.61%	557
GDT 2	43.89%	44.93%	54.10%	52.40%	66.09%	581
GDT 3	42.35%	53.55%	54.27%	55.04%	68.97%	477
GDT 4	45.02%	43.07%	53.22%	54.96%	64.60%	613
GDT 5	44.09%	44.09%	54.01%	52.33%	63.68%	592
GDT 6	44.58%	52.53%	52.69%	54.02%	65.88%	507
GDT 7	43.33%	50.51%	53.92%	54.90%	65.57%	517
GDT 8	43.61%	38.85%	55.07%	60.34%	66.36%	642
GDT 9	43.36%	45.27%	54.54%	53.82%	65.21%	572
GDT 10	43.79%	50.34%	53.57%	55.09%	65.58%	523
MEAN	43.51%	46.77%	54.19%	55.07%	65.86%	558.1
STD	1.11%	4.71%	1.07%	2.42%	1.39%	51.7

To see the effect of the constrained fitness function, we compare the above results with those generated by FGP using RC only as the fitness function (i.e. $f_{(1)}$ with $w\_rmc = w\_rf = 0$). Results are listed in Table 1.3. From Table 1.3, we can see that by using $f_{(2)}$, the mean RF is reduced from 43.51% to 40.06%. For reference, the mean AARR rises from 55.07% (Table 1.3) to 63.38% (Table 1.2) and the mean RPR rises from 65.86% to 69.95%. The price to pay for a lower RF is that more opportunities were missed: the mean RMC rises from 46.77% to 65.74%. The mean RC only slightly decreases from 54.19% to 53.72%. To determine whether result differences are statistically significant, the two-tailed paired t-test was applied on the null hypothesis that the mean performances of two groups were not statistically different under each of the five criteria. Shown in Table 1.4 are t-values and their corresponding p-values under each criterion. The results indicate that by using the constrained fitness function with $\Re$= [35%, 50%], FGP-2 generates decision trees with statistically better RF, AARR and RPR at a significant level of $\alpha = 0.001$, though they have statistically worse RMC. These decision trees do not show any statistically different for RC (p-value is 0.2612). That is, RC has not been compromised as RF is reduced.

Table 1.4. t-statistics for comparing mean performances of two groups (Using RC only versus using the constrained fitness function with $\Re = [35\%, 50\%]$)

Criteria	For RF	For RMC	For RC	For AARR	For RPR
t values	-4.64	6.33	-1.16	4.69	4.71
p values($\alpha = 0.001$)	0.0002.5	0.000005	0.261247	0.000182	0.000175

4. Empirical evaluation of the constrained function

4.1 Objective of the experiments

To test FGP-2's usefulness as a tool for tuning precision against missing opportunities, we tested it on historic data. We should re-iterate that FGP-2's prediction accuracy is limited by whether or not the predicted value is a function of the observed variables (as we pointed out in Section 1.1, point 1). When no such function exists, neither FGP-2 nor any other comparable algorithms would be able to make accurate predictions. The primary objective of the test results presented below is not to demonstrate that FGP-2 can predict DJIA accurately. (In fact, the results presented below do not represent the most accurate predictions that FGP-2 has ever made.) Instead, the primary objective is to observe whether FGP-2 can be used to trade precision with the rate of missing opportunities. We would only conclude that FGP-2 achieves what it is designed to achieve if one could instruct it to improve precision at the cost of increasing the rate of missing opportunities, or vice versa.

4.2 Experimental Data

In this section, we present a typical set of test results by FGP-2, based on daily closing prices of the Dow Jones Industrial Average (DJIA) Index. Other indices and share prices have been used with similar results. Experiments presented in this paper were carried out on DJIA daily closing index from 07/04/1969 to 09/04/1981, a total of 3,035 trading days, as illustrated in Figure 1.1. We took the data from 07/04/1969 to 11/10/1976 (1,900 trading days) as training data, and the period from 12/10/1976 to 09/04/1981 (1135 trading days) as testing data. For the purpose of analysis, we chose $r = 2.2$ and $n = 21$ days, which give roughly 50% of positive positions in both the training and test periods. Details of the experiemental setup are shown in table 1.5

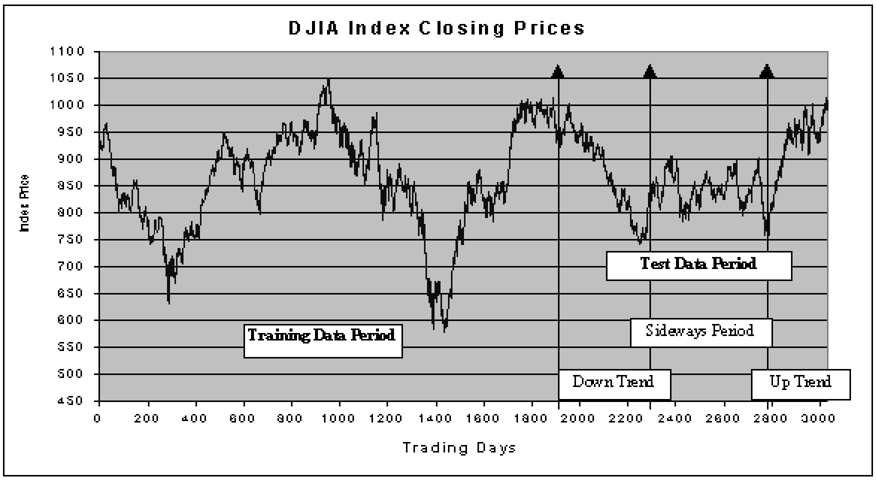

Figure 1.1. Dow Jones Industrial Average (DJIA) index daily closing prices from 07/04/1969 to 09/04/1980 (3035 trading days)

Table 1.5. Parameters used in FGP-2 for experiments

Target	To find GDTs with low RF, with return of 2.2% with 22 days
Input terminals	Six technical indicators plus Real as thresholds
Prediction terminals	$\{0, 1\}$: 1 representing "Positive"; 0 representing "Negative".
Non-terminals	If-then-else , And, Or, Not, $>, \geq, <, \leq, =$.
Crossover rate	0.9
Mutation rate	0.01
Population size	1,200
Maximum no. of generations	30
Termination criterion	Generation limit or Time limit, whichever reached first
Selection strategy	Tournament selection, Size = 4
Max depth of individual programs	17
Max depth of initial individual programs	4
Maximum run time (hours)	2
Hardware and operating system	Pentium PC 200MHz running Windows 95 with 64M RAM.
Software	Borland C++ (version 4.5)

4.3 Experimental Results

To further explore the impact of the constraint $\Re$ on reducing RF, we took five additional non-overlapping $\Re$s in the fitness function respectively. The five mutually exclusive $\Re$s are [5%, 10%], [10%, 15%], [15%, 20%], [20%, 35%] and [50%, 60%]. For each $\Re$, we run FGP-2 ten times using the parameters $w\_rc = w\_rf = 1$ and $w\_rmc = 0$. We calculated the mean performances on test data with respect to RF, RMC, RC, RPR, AARR and the mean number of positive recommendations. All experimental results are showed in Table 1.6. The results are visualised in Figure 1.2.

Table 1.6. The effect of the constraint $\Re$ on the mean performances of FGP-2

$\Re$ [% %]		RF	RMC	RC	AARR	RPR	Number of Recommendation
[5, 10]	Mean	13.48%	99.14%	48.19%	224.03%	92.22%	6.2
	SD	14.85%	0.63%	0.26%	229.24%	10.86%	4.8
[10,15]	Mean	28.60%	94.05%	49.70%	136.81%	82.95%	49.3
	SD	6.22%	1.65%	0.76%	30.52%	4.40%	13.1
[15,20]	Mean	31.02%	85.69%	51.74%	99.58%	79.02%	125.1
	SD	5.21%	6.41%	1.67%	25.50%	5.47%	62.7
[20,35]	Mean	36.00%	75.25%	53.41%	75.68%	73.61%	229.8
	SD	2.59%	5.50%	1.19%	9.55%	3.41%	55.1
[35,50]	Mean	40.06%	65.74%	53.72%	63.38%	69.95%	338.9
	SD	1.41%	2.99%	0.48%	3.53%	1.67%	34.7
[50.65]	Mean	46.73%	45.47%	51.31%	52.26%	62.57%	606.2
	SD	1.37%	10.40%	1.64%	1.63%	1.67%	115.4

Figure 1.2 shows that RF decreases gradually as $\Re$ is reduced. The lowest RF (13.48%) is obtained by using the smallest $\Re$ [5%, 10%] whereas the highest RF (46.73%) is obtained by using the biggest $\Re$ [50%, 65%]. The six mean RFs in the graph suggest that tightening the constraint ($\Re$) in the fitness function may lead to a lower RF. Reduction in RF obviously benefited RPR and AARR in the test data. RPR rises from 57.16% to 92.85%. AARR increases dramatically from 40.32% to 300.33%. The results obtained by using the tightest constraints [5%, 10%] provide the most reliable recommendations, with a failure rate of 13.48%. The price to pay for using a constraint of this tightness is that it makes fewer positive recommendations, which leads to higher rate of missing chances (RMC). If we reduce $\Re$ beyond a certain point, no positive recommendations will be made by FGP-2. Results in this experiment suggest that $\Re$ is a useful handle for tuning RF against RMC in FGP.

We also tested FGP-2 on different market conditions, namely, (a) down-trend period from 12/10/1976 to 12/04/78 (378 trading days); (b) side-way-trend period from 13/04/1978 to 27/03/1980 (496 trading days); and (c) up-trend period from 28/03/1980 to 09/04/81 (261 trad-

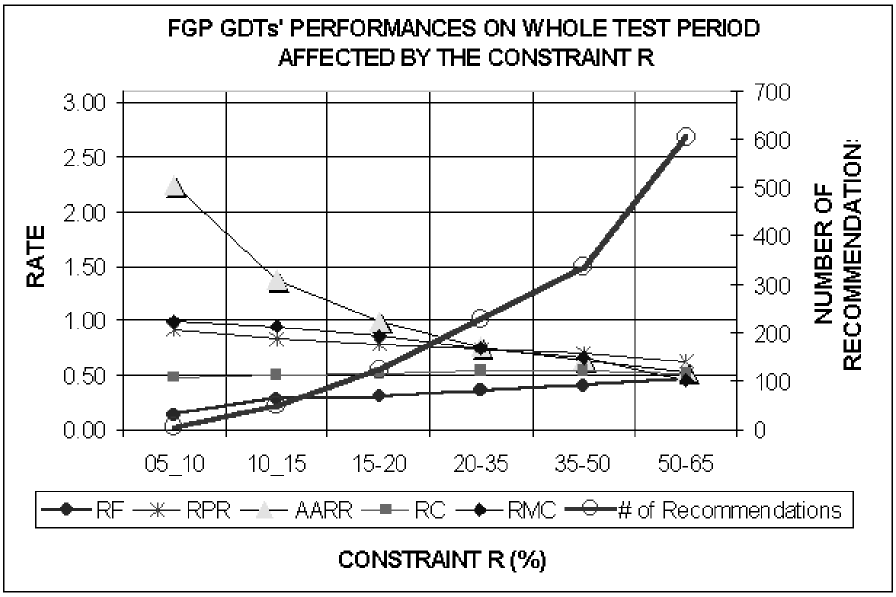

Figure 1.2. Visualization of the effect of the constraint $\Re$ on the mean performances of FGP-2

ing days), as illustrated in Figure 1.1. Results obtained were consistent with those shown above. FGP-2 was tested extensively in other data sets. For simplicity, details of values are not presented here.

5. COMPARISON STUDY

Up to this point, we only tested FGP-2 with the constrained fitness function on a financial index. Should it still be effective and applicable to individual stock data? How does FGP-2 compare with other methods? To partially answer these questions, we referred to Saad et al (1998) in which three specially developed Neural Networks, (i.e. Time Delay (TDNN), Recurrent (RNN) and Probabilistic (PNN)), and a linear classifier were employed to address a similar prediction problem. They also have the goal of achieving low false alarm.

We compared performances based on predictions with $r = 2\%$ and $n = 22$; i.e. daily predictions on whether a return of 2% or more can be achievable within the next 22 trading days. We tested the above algorithms on ten stocks:

- Apple (AAPL), IBM(IBM), Motorola(MOT), Microsoft (MSFT): representing the technology group which generally has high volatility

- American Express(AXP), Well Fargo (WFC): representing the banks
- Walt Disney Co. (DIS), McDonald (MCD): representing consumer stocks
- Public Svc New Mexico (PNM), Energras (V.EEG): representing cyclical stocks

These data series vary in their starting dates, but all ended by 06/03/1997. Following [Saad et. al 1998], the last 100 days were chosen as the test data for each stock.

In the experiments, we ran FGP-2 10 times for each data set. For each run, we took 500 trading days just before the final 100 days as training data, and took a constraint $\Re$ = [20%, 30%] for most data sets except for AAPL, PNM and V.EEG, for which we took a constraint $\Re$ = [10%, 20%]. The $\Re$ values were chosen to reflect the percentage of positive positions in the data. The termination condition was set to 50 generations. Since FGP-2 is a probabilistic technique, it was run ten times for each share. For each share, we picked the best decision tree generated in FGP-2's ten runs for the purpose of comparison, as the same was done for the three different neural networks reported in [Saad et. al 1998].

Table 1.7 lists the performance of the three different NNs, a linear classifier and FGP-2 on 10 stocks. The "Total" column summarises the total number of predicted positive positions on all 10 stocks. The last column, "Ave.", reports the average rate of failure over 10 stocks. On average, the NNs out-performed the linear classifier in RF (7.56%, 3.05% and 3.61% as opposed to 18.62%). The average RF for all the GDTs generated was 5.08% (much better than the RF achieved by the linear classifier, 18.62%). The average RF by the best GDT was 1.29%, which was lower than any of the NNs [2]. The best-found GDT found 385 positive signals totally, which is slightly more than 372 found by the linear classifier. This shows that RMC has not been compromised by FGP-2 in its attempt to reduce RF. On individual shares, the RF by the best GDT found by FGP-2 was at least as good as the RF found by the NNs in 8 out of the 10 shares.

6. Conclusion and future research

How accurate a forecasting program can be is limited not only by the algorithm that it uses, but also by the quality of the data. In previous papers, we have reported the capability of FGP-2 in finding patterns in historical data when patterns exist. In this paper, we argue that, depending on the application, a user may want a forecasting program to

Table 1.7. Performance comparisons among NNs., a linear classifier and FGP-2 in terms of RF and N_+ (the total number of predicted positive positions)

Stocks			AAPL	IBM	MOT	MSFT	AXP	WFC	DIS	MCD	PNM	V.EEG	Total	Ave.
Profit Opp. (r=2%;n=22)			62	72	81	87	92	85	74	73	50	70	746	74.6
PPN	Total N+		51	25	48	49	20	45	19	4	63	14	338	
	RF (%)		7.84	4.00	18.75	4.08	0.00	4.44	0.00	0.00	36.50	0.00		7.56
TDNN	Total N+		10	9	27	61	17	19	7	6	22	8	186	
	RF (%)		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	18.00	12.50		3.05
RNN	Total N+		16	22	33	46	49	29	48	53	35	37	368	
	RF (%)		0.00	0.00	3.03	2.17	0.00	0.00	0.00	5.66	17.14	8.11		3.61
The Linear Classifier	Total N+		82	24	87	17	10	22	2	32	20	76	372	
	RF (%)		31.71	20.83	18.39	0.00	0.00	13.64	0.00	21.88	60.00	19.74		18.62
Mean and STD of 10 GDTs	Total N+	Mean	18.5	68.7	20.7	26.8	38.3	66.6	20.1	40.2	23.4	49.4	373	
		STD	9.9	3.9	5.1	6.2	9.9	11.1	3.1	1.8	5.9	9.6		
	RF (%)	Mean	9.16	10.15	1.33	3.10	3.72	8.20	0.40	0.00	13.07	4.83		5.08
		STD	5.66	1.13	2.82	2.47	3.10	2.33	1.30	0.00	12.30	3.90		
The Best GDT	Total N+		4	70	28	33	39	69	22	43	28	49	385	
	RF (%)		0.00	8.57	0.00	0.00	0.00	4.35	0.00	0.00	0.00	0.00		1.29

sacrifice investment opportunities for high precision, or vice versa. Our mission is to develop a forecasting tool to help users achieve their preferred performance. FGP-2 allows us to favour precision or investment opportunities through adjusting the tightness of a constraint in the objective function. The effectiveness of FGP-2 in influencing the precision is supported by our experiments.

Many other issues are relevant to the practicality of FGP-2. First of all, it must be established that in the series to be predicted, past patterns will repeat themselves in the future. Secondly, predictions only improve one's odds statistically. One needs to know how to use the predictions to invest one's money, so as to reduce risk. Moreover, if EDDIE were to be asked to recommend buying cautiously, one must have a viable policy for investing one's capital when it is idle. These issues will be left to future research.

Acknowledgments

Full credit should be given to James Butler who invented the acronym EDDIE. He implemented EDDIE-1 and tested it on horse racing. Many colleagues and students at University of Essex have contributed to the EDDIE project through discussion. In particular, we would like to thank Paul Scott, Jeff Reynolds, Tung-Leng Lau, Robert Pugh, Tony Lawson (MSc student), Sheri Markose, Abdel Salhi, Hakan Er, Xin Yao and Qingfu Zhang. This project is partly supported by two Research Promotion Funds by the University of Essex. Jin Li was supported by the

Overseas Research Scholarship and the University of Essex Scholarship. The DJIA index data was kindly supplied by Professor Blake LeBaron of University of Wisconsin-Madison. Ten stock data were kindly supplied by Emad W. Saad of the Texas Tech University.

Notes

1. For the record, this rule recommended 33 "buy" over 500 trading days between 22 May 1998 and 21 April 2000. All 33 occasions resulted in a gain of 4the target return during the training over the preceding 1000 days.
2. The favourable results by FGP may be partly due to the rather bullish market over test period in which over 50% of the position are positive for all the shares; e.g. 87% of the positions were positive for MSFT and 92% for AXP.

References

Li, J. & Tsang, E.P.K, Improving technical analysis predictions: an application of genetic programming, Proceedings of The 12th International Florida AI Research Society Conference, Orlando, Florida, 1999, 108-112.

Li, J. & Tsang, E.P.K, Investment decision making using FGP: a case study, Proceedings of the 1999 Congress on Evolutionary Computation, IEEE Press, Washington DC, USA, 1999, 1253-1259.

Saad, E.W., Prokhorov, D. V. & Wunsch II, D. C., Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks, IEEE Trans. Neural Networks, Vol. 9, No. 6, 1998, 1456-1470

Tsang, E.P.K., Li, J. & Butler, J.M., EDDIE beats the bookies, International Journal of Software, Practice & Experience, Wiley, Vol.28 (10), 1998, 1033-1043

Tsang, E.P.K. & Li, J., Combining Ordinal Financial Predictions With Genetic Programming, In: Leung K.S., Chan L.-W., Meng H. (Eds.), Intelligent Data Engineering and Automated Learning- IDEAL 2000: Data Mining, Financial Engineering, and Intelligent Agents, Lecture Notes in Computer Sciences 1983. Springer, 532–537