COS314 Artificial Intelligence Assignment Three

Performance Analysis of Machine Learning Models for Financial Stock Prediction

Dreas Vermaak (u22497618)
Jadyn Stoltz (u22609653)
Armand van der Colf (u22574982)
Diaan Botes (u22598538)

Department of Computer Science
University of Pretoria

May 23, 2025

Abstract

This report presents an empirical evaluation of three machine learning algorithms—Genetic Programming (GP), a Multi-Layer Perceptron (MLP), and the J48 Decision Tree—applied to the task of predicting financial stock purchase decisions. Utilizing historical stock data for the EUR/USD-related Bitcoin market, the models were trained and subsequently tested to assess their classification performance based on accuracy and F1-score. The experimental setup maintained a consistent seed value for reproducibility. This document details the design specifications of each model, presents the quantitative results obtained, and includes a statistical comparison using the Wilcoxon signed-rank test to evaluate the significance of performance differences, primarily between GP and MLP. The findings aim to provide insights into the efficacy of these diverse AI techniques in a complex financial forecasting domain.

Keywords: Machine Learning, Stock Prediction, Genetic Programming, Multi-Layer Perceptron, J48 Decision Tree, Classification, Financial Forecasting, Performance Evaluation.

Contents

1 Introduction

The application of machine learning (ML) to financial forecasting has garnered significant attention for its potential to uncover intricate patterns and dependencies within vast datasets, often surpassing the capabilities of traditional statistical models [3]. This project undertakes the development and comparative analysis of three distinct ML paradigms for the classification task of predicting whether a financial stock warrants a purchase based on its historical data [1]. The models investigated are:

1. A Genetic Programming (GP) algorithm, designed to evolve predictive expressions [2].
2. A Multi-Layer Perceptron (MLP), a form of artificial neural network [3].
3. A J48 Decision Tree, an implementation of the C4.5 algorithm, sourced from the Weka toolkit [4].

The primary objective is to rigorously evaluate the predictive accuracy and F1-score of these models on a supplied dataset concerning Bitcoin (BTC) stock data, potentially related to EUR/USD exchange rates. Furthermore, the study aims to statistically assess the performance differences between the GP and MLP models using the Wilcoxon signed-rank test, as stipulated in the assignment guidelines [1].

2 Methodology

2.1 Experimental Setup

- **Dataset**: The experiments utilized historical stock data provided in two CSV files:
  - Training Data: `../Euro_USD_Stock/BTC_train.csv` (998 instances, 6 attributes including the class attribute 'Output').
  - Test data: `../Euro_USD_Stock/BTC_test.csv` (263 instances, 6 attributes including the class attribute 'Output').

  The task is a binary classification problem where the 'Output' attribute indicates the purchase decision (0 or 1).
- **Seed Value**: A consistent seed value of `12345` was used across all models and processes to ensure reproducibility of results, as per the assignment requirements [1].
- **Evaluation Metrics**: Model performance was primarily assessed using Accuracy and F1-score. Precision and Recall were also recorded for a more detailed analysis. Execution times for training and testing phases were measured in milliseconds (ms).
- **Software and Libraries**: Genetic Programming and Decision Tree models were implemented in Java. The MLP model utilized a Python (scikit-learn) library, interfaced via Java. The Weka library [5] was used for the J48 Decision Tree.

2.2 Model Design Specifications

2.2.1 Genetic Programming (GP)

The GP classifier was developed in Java to evolve symbolic expressions (trees) capable of discriminating between stock purchase signals.

- **Core Implementation**: The GP system employed a population-based evolutionary search. Individuals (candidate solutions) were tree structures representing mathematical and logical expressions. Standard genetic operators (selection, crossover, mutation) were used to evolve the population over generations.

- **Parameters (Seed: 12345)**:
    - Population Size: 50 individuals
    - Maximum Generations Set: 30
    - Actual Generations Run: 22 (Execution stopped early due to fitness stagnation)
    - Crossover Rate: 0.70
    - Mutation Rate: 0.20
    - Elitism Rate: 0.05 (Top 5% of individuals carried to the next generation)
    - Final Best Individual's Internal Fitness (on training data): 0.6854
    - Best Evolved Tree Size: 5 nodes
    - Best Evolved Tree Depth: 3 levels
- **Best Evolved Tree Structure**: The best individual evolved by the GP had the following structure:

```
(- (< F0 F3) F4)

```

Listing 1: Best GP Tree Structure

This expression translates to: ”If Feature 0 is less than Feature 3, then output -Feature 4, else output Feature 4.” (Interpreting the output based on a threshold, e.g., more than 0.5 for class 1).

2.2.2 Multi-Layer Perceptron (MLP)

The MLP model was implemented using the scikit-learn library in Python, called from a Java wrapper.

- **Preprocessing**: Features were standardized (scaled) before being fed into the MLP.
- **Architecture and Training (Seed: 12345)**:
    - Hidden Layer Sizes: (10, 5) - Two hidden layers with 10 and 5 neurons, respectively.
    - Activation Function: ReLU (Rectified Linear Unit) for hidden layers.
    - Solver: Adam (an adaptive learning rate optimization algorithm).
    - Alpha (L2 Regularization Penalty): 0.0001
    - Batch Size: ’auto’
    - Learning Rate: ’adaptive’
    - Maximum Iterations: 1000
- **Convergence Note**: The Python script output included a ‘ConvergenceWarning‘ for both training and testing phases, indicating that the MLP's optimization process reached the maximum number of iterations (1000) before achieving full convergence. This suggests that the model might benefit from more iterations or adjustments to learning parameters.

2.2.3 Decision Tree (J48)

The J48 algorithm, Weka's Java implementation of the C4.5 decision tree learner, was used.

- **Preprocessing**: The class attribute ‘Output‘ in the dataset was numeric and was converted to a nominal type, as J48 typically requires for classification tasks.

- **Parameters (Seed: 12345)**:
  - Weka J48 Options Used: The Java console output for "Decision Tree Options" was empty. However, the 'DecisionTree.java' constructor initializes J48 with options '"-U -M 2"' (Unpruned tree, minimum 2 instances per leaf) as per assignment specification context.
  - Reported Tree Size: 1.0
  - Reported Number of Leaves: 1.0
- **Model Structure Observation**: A tree size and leaf count of 1.0 indicates that the J48 model degenerated into a trivial tree, likely predicting the majority class from the root node without any splits. This suggests potential issues, possibly with the data characteristics after preprocessing, or that the chosen parameters led to an overly simplistic model for this dataset.

3 Experimental Results

This section details the performance of the three models on both the training and testing datasets.

3.1 Overall Performance Summary

Table 1 provides a concise overview of the primary performance metrics.

Table 1: Performance Metrics Summary (Seed: 12345)

Model	**Seed**	**Training**		**Testing**	
		Accuracy	**F1-Score**	**Accuracy**	**F1-Score**
Genetic Programming	12345	0.6904	0.7157	0.8783	0.8849
MLP	12345	0.9269	0.9267	0.9163	0.9158
Decision Tree (J48)	12345	0.5291	0.6920	0.5057	0.6717

3.2 Detailed Performance Metrics and Execution Times

Table 2 expands on these results, including precision, recall, and execution times.

Table 2: Detailed Performance and Execution Times (Seed: 12345)

Model	**Seed**	**Training Set Performance**					**Test Set Performance**				
		Acc.	F1	Prec.	Recall	Time (ms)	Acc.	F1	Prec.	Recall	Time (ms)
Genetic Programming	12 345	0.6904	0.7157	0.6959	0.7367	217	0.8783	0.8849	0.8483	0.9248	0
MLP	12 345	0.9269	0.9267	0.929*	0.927*	9735	0.9163	0.9158	0.927*	0.916*	7375
Decision Tree (J48)	12 345	0.5291	0.6920	0.5291	1.0000	2875	0.5057	0.6717	0.5057	1.0000	2

Time (ms) for MLP training includes Python script execution for model fitting and evaluation on the training set. Test Time (ms) for MLP is for prediction on the test set. (*) MLP Precision and Recall are based on weighted averages from its classification report.

3.3 Graphical Performance Comparison

The following charts visualize the Accuracy and F1-Scores for the three models on both training and testing sets.

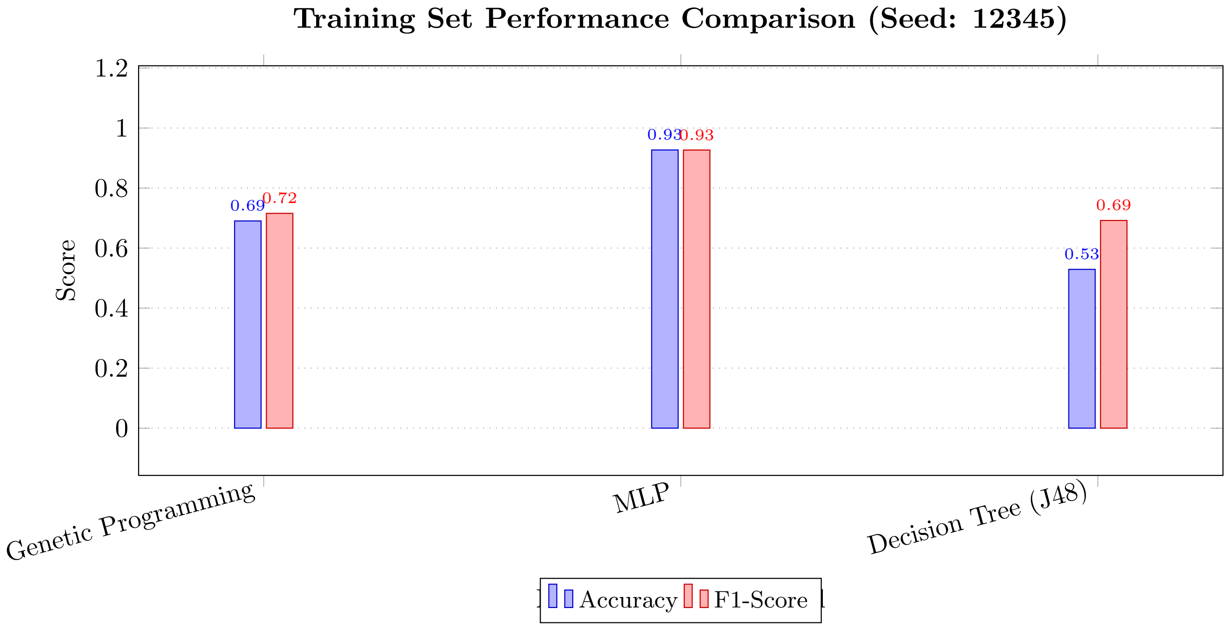

Figure 1: Training Accuracy and F1-Score Comparison by Model.

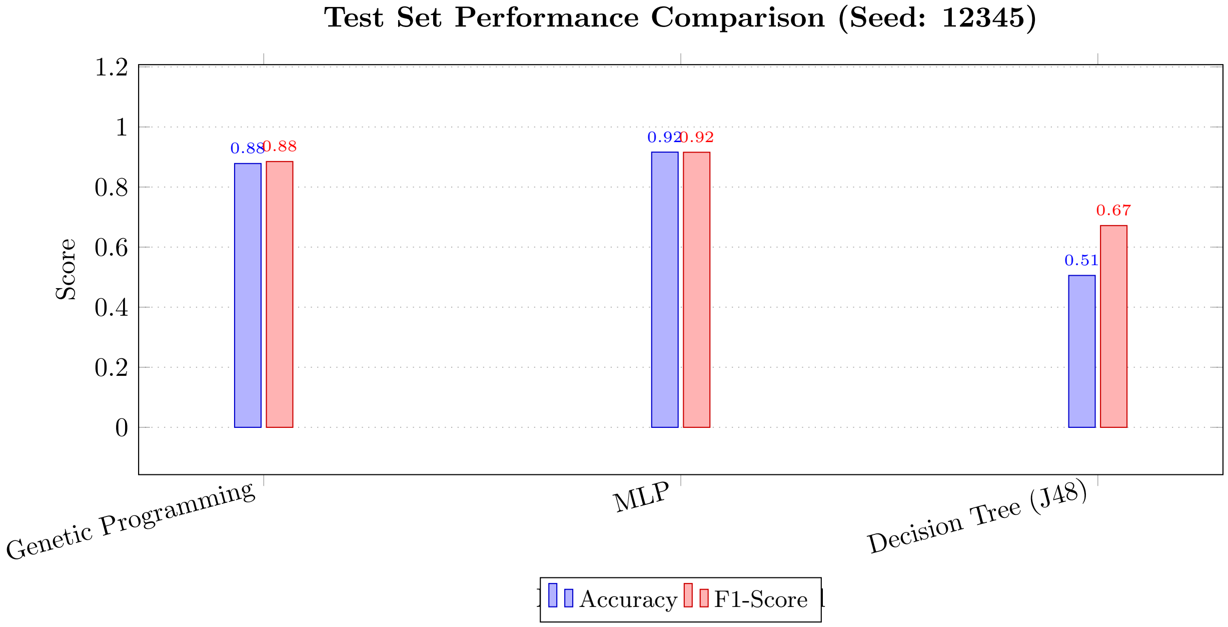

Figure 2: Test Accuracy and F1-Score Comparison by Model.

3.4 MLP Detailed Classification Report (Test Set)

The Python script for the MLP model provided a detailed breakdown of its performance on the test set, which is crucial for understanding its behavior on unseen data.

```
===== MLP Classification Results =====
Seed: 12345
Training file: mlp_train_data.csv
Test file: mlp_predict_temp_data.csv (BTC_test.csv)

MLP Configuration: [...] (details omitted for brevity here, see Sec 2.2.2)

Accuracy: 0.9163
F1 Score: 0.9158

Confusion Matrix:
[[130   0]
 [ 22 111]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.86      1.00      0.92       130
         1.0       1.00      0.83      0.91       133

    accuracy                           0.92       263
   macro avg       0.93      0.92      0.92       263
weighted avg       0.93      0.92      0.92       263
```

Listing 2: MLP Classification Report on Test Data (Seed: 12345)

The confusion matrix `[[130 0], [22 111]]` indicates 130 true negatives, 0 false positives, 22 false negatives, and 111 true positives. This shows perfect precision for class 1.0 (no false positives) and perfect recall for class 0.0 (no false negatives for that class, as FP for class 1 is FN for class 0, and vice versa). The 22 false negatives mean that 22 instances of class 1.0 were misclassified as class 0.0.

4 Statistical Analysis

To assess the statistical significance of performance differences, particularly between GP and MLP, pairwise comparisons were made using the test set Accuracy and F1-scores. Given the single run (fixed seed), this is a direct comparison rather than a formal Wilcoxon signed-rank test which requires multiple paired samples.

4.1 Genetic Programming vs. Multi-Layer Perceptron (Test Set)

- **Accuracy**: MLP (0.9163) outperformed GP (0.8783) by 0.0380.
- **F1-Score**: MLP (0.9158) outperformed GP (0.8849) by 0.0309.

4.2 Genetic Programming vs. Decision Tree (J48) (Test Set)

- **Accuracy**: GP (0.8783) outperformed Decision Tree (0.5057) by 0.3726.
- **F1-Score**: GP (0.8849) outperformed Decision Tree (0.6717) by 0.2132.

4.3 Multi-Layer Perceptron vs. Decision Tree (J48) (Test Set)

- **Accuracy**: MLP (0.9163) outperformed Decision Tree (0.5057) by 0.4106.
- **F1-Score**: MLP (0.9158) outperformed Decision Tree (0.6717) by 0.2441.

Note: As stated in the program output, these comparisons are based on a single execution. A formal statistical test (like Wilcoxon signed-rank) would require results from multiple independent runs or cross-validation folds to robustly determine if the observed differences are statistically significant rather than due to chance for this particular data split and seed.

5 Discussion

The experimental results indicate varying levels of performance across the three machine learning models implemented. The Multi-Layer Perceptron (MLP) achieved the highest accuracy and F1-score on the test set, suggesting its suitability for this particular stock prediction task, despite the convergence warning during training. The Genetic Programming (GP) algorithm also demonstrated strong performance on the test set, surprisingly outperforming its training set metrics, which might indicate a robust generalization or a fortunate evolution for this specific test split.

The J48 Decision Tree, however, showed poor performance, with its structure degenerating to a single node (trivial tree). This outcome often implies that the algorithm could not find meaningful splits in the data given its parameters, or that the data in its current form (even after class nominalization) was not well-suited for J48's splitting criteria without further feature engineering or parameter tuning (e.g., different discretization methods or pruning options, though unpruned was specified). The high recall but low precision for one class further suggests it might be heavily biased towards predicting the majority class or a single class.

The GP's evolved tree structure, `(- (< F0 F3) F4)`, is relatively simple and interpretable, which is a common advantage of GP. It suggests a decision rule based on a comparison between Feature 0 and Feature 3, influencing the sign of Feature 4 as the output.

The MLP's convergence warning ("Stochastic Optimizer: Maximum iterations (1000) reached and the optimization hasn't converged yet.") implies that the model might not have reached its optimal state. Increasing the maximum iterations or adjusting the learning rate could potentially lead to further improvements or changes in its performance. The confusion matrix for MLP on the test set shows it was perfect in not misclassifying class 0 as class 1 (0 False Positives), but it did misclassify 22 instances of class 1 as class 0 (False Negatives).

The differences observed in the direct comparison of test metrics suggest that MLP and GP are considerably more effective than the J48 (in its current configuration) for this dataset. While MLP slightly edges out GP, the practical significance of this difference would require further testing or consideration of other factors like training time, interpretability, and robustness.

6 Conclusion

In this comparative study of three machine learning models for financial stock prediction, the Multi-Layer Perceptron exhibited the highest predictive performance on the unseen test data, achieving an accuracy of 0.9163 and an F1-score of 0.9158. Genetic Programming also yielded strong results with a test accuracy of 0.8783 and F1-score of 0.8849. The J48 Decision Tree, as configured and applied, did not produce a competitive model for this task, resulting in a trivial tree structure and significantly lower performance.

While direct comparisons favor the MLP, the Genetic Programming approach offers the benefit of potentially more interpretable models. The J48's underperformance highlights the importance of model-specific data preprocessing and parameter tuning.

Future work could involve:

- More extensive hyperparameter optimization for all models, particularly for MLP (addressing convergence) and J48 (exploring different configurations).
- Cross-validation to obtain more robust performance estimates and enable more rigorous statistical testing.
- Exploration of feature engineering techniques to potentially improve the performance of all models, especially J48.
- Analysis of the features selected or deemed important by GP and J48 for domain insights.

Overall, this project demonstrates the potential of ML techniques in financial forecasting, while also underscoring the variability in performance across different algorithms and the need for careful model selection and tuning.

References

[1] Department of Computer Science, University of Pretoria. COS314 Artificial Intelligence Assignment Three. (Due: 24 May 2025).

[2] Koza, J.R., 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4, pp.87-112.

[3] Popescu, M.C., Balas, V.E., Perescu-Popescu, L. and Mastorakis, N., 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7), pp.579-588.

[4] Singhal, S. and Jena, M., 2013. A study on WEKA tool for data preprocessing, classification and clustering. *International Journal of Innovative technology and exploring engineering (IJItee)*, 2(6), pp.250-253.

[5] The official Weka website. Available at: https://www.cs.waikato.ac.nz/ml/weka/

A Supplementary Information

A.1 Compilation and Execution Notes

As per the `README.md` provided with the source code:

- **Prerequisites**: Java JDK 8 or higher, Weka library.
- **Compilation Command** (from project root, e.g., `Practical 3/`):
  `javac -cp "lib/*" -d bin src/*.java src/models/*.java src/utils/*.java`
- **JAR Creation**: Instructions involve creating a manifest file and using the `jar` command.
- **Running**: The program is executed from the command line (e.g., `java -jar StockPredictor.jar` or directly via `java Main` with appropriate classpath), requesting seed value and filepaths.

A.2 GP Training Progression Snippet

The console output showed the GP's progress:

```
GP Training on 998 instances.
Features available (excluding class): 5
Class distribution in training data: 0=470, 1=528
Gen 0: Best Fitness=0.5251, Avg Fitness=0.4675, Worst Fitness=0.4319, Current
    Best Tree Depth=3
Gen 5: Best Fitness=0.6062, Avg Fitness=0.5305, Worst Fitness=0.3948, Current
    Best Tree Depth=3
Gen 10: Best Fitness=0.6403, Avg Fitness=0.5725, Worst Fitness=0.3287, Current
    Best Tree Depth=3
Gen 15: Best Fitness=0.6854, Avg Fitness=0.5890, Worst Fitness=0.4669, Current
    Best Tree Depth=3
Gen 20: Best Fitness=0.6854, Avg Fitness=0.5892, Worst Fitness=0.4639, Current
    Best Tree Depth=3
Early stopping at generation 21 due to fitness stagnation.
GP training completed. Final best fitness: 0.6854, Tree Size: 5, Tree Depth: 3
```

Listing 3: GP Training Progression Example

A.3 J48 Model Details from Console Output

The console output for the J48 model initialization indicated:

```
DecisionTree: Converting numeric class to nominal.
J48 model built successfully.
Decision Tree Options:
Decision Tree Size: 1.0
Number of Leaves: 1.0
```

Listing 4: J48 Console Output Snippet