COS314 Artificial Intelligence: Assignment Three Report

Group Members: Joshua Garner, Christopher Xides, Shravan Seebran, Kgosi Segale
Department of Computer Science
University of Pretoria

May 2025

Abstract

This report presents the implementation and evaluation of three machine learning models—Genetic Programming (GP), Multi-Layer Perceptron (MLP), and Decision Tree (J48)—for predicting whether a financial stock should be purchased based on historical data. The design specifications of each model are detailed, followed by a performance evaluation using accuracy and F1 scores on training and test datasets. A Wilcoxon signed-rank test is conducted to assess the statistical significance of performance differences between the GP and MLP models. Results are summarized in tables, and the report concludes with insights into the models' effectiveness and potential improvements.

Contents

1 Introduction

This report details the implementation of three machine learning models for a stock purchase classification task as part of COS314 Artificial Intelligence at the University of Pretoria. The models include a Genetic Programming (GP) classification algorithm, a Multi-Layer Perceptron (MLP), and a Decision Tree (J48 from the Weka package). The report outlines the design specifications of each model, presents performance results in terms of accuracy and F1 scores, and includes a statistical comparison between GP and MLP using the Wilcoxon signed-rank test. The assignment was completed by a group of four members, with responsibilities divided as follows: Christopher Xides (GP + Report), Shravan Seebran (GP + Report), Joshua Garner (MLP + Report), and Kgosi Segale (Decision Tree + Report).

2 Model Design Specifications

2.1 Genetic Programming (GP)

The Genetic Programming model was designed to evolve classification rules for stock purchase decisions. The implementation follows the principles outlined in the course notes Nyathi [2025]. Key design components include:

- **Population Initialization**: The initial population was set up using a tree-based structure. The arithmetic operators +, -, *, / were selected as the function set. The terminal set included data constants representing stock features provided in the Euro USD Stock files, i.e., Open, High, Low, Close, AdjClose. This was done to ensure that the sets satisfied the closure and sufficiency properties. The population size was a parameter adjustable in the `Parameters.java` file. The grow method was used to construct individuals, which consisted of recursive, randomly generated trees of variable length, constrained by a maximum depth specified in the `Parameters.java` file.

- **Fitness Function**: The F1 fitness score metric was used to evaluate each individual tree. This was done by parsing the tree into its mathematical expression and evaluating it from left to right. The resulting value was normalized using the sigmoid function, with values greater than 0.5 assigned an output of 1, and others assigned an output of 0. This evaluation was applied across all stock data, and the counts of true positives, true negatives, false positives, and false negatives were tallied. These counts were then used to compute the F1 fitness score for each individual.

- **Genetic Operators**: Crossover and mutation were applied in each generation. Crossover was performed by specifying the number of parents to be selected, using the tournament selection method. The left branches of both parents were swapped to produce new offspring. For mutation, both grow and shrink mutation methods were used. The number of parents to undergo mutation was specified in the `Parameters.java` file. The specific mutation method was randomly chosen. For the grow mutation, a leaf was randomly selected and extended with a randomly generated subtree, ensuring the maximum depth was not exceeded. For the shrink

mutation, a random node was selected and replaced with a leaf node.

- **Parameters**: The parameters were defined in the `Parameters.java` file and included population size, number of generations (iterations), a parameter controlling the early generation of leaves, tournament selection size, the number of parents selected for mutation and crossover, and the maximum depth.

- **Implementation Details**: The program was implemented in Java. CSV files were read and processed in the `Train.java` and `Test.java` files, with individual data records stored in the `Data.java` file. A random seed was requested at the start of the program to ensure reproducibility.

2.2 Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron was implemented to classify stock purchase decisions based on historical data, following the concepts in Popescu et al. [2009]. The design includes:

- **Network Architecture**: The MLP consists of three layers: an input layer with 5 nodes (corresponding to the 5 features in the dataset), a hidden layer with 10 nodes, and an output layer with 2 nodes (for binary classification: buy or not buy). The sigmoid activation function, defined as $\sigma(x) = \frac{1}{1+e^{-x}}$, is applied to both the hidden and output layers.

- **Training Process**: The model is trained using backpropagation with stochastic gradient descent (batch size of 1). The training loop iterates over the dataset for 1000 epochs, updating weights and biases based on the error between predicted and target outputs. The error is computed using the derivative of the sigmoid function, and updates are applied with a learning rate of 0.01.

- **Data Preprocessing**: The dataset is loaded from CSV files, with the first 5 columns as features and the last column as the label. No normalization or feature scaling was applied; raw feature values are used directly in the model.

- **Parameters**: The model uses a user-specified seed value for reproducibility, passed to Java's `Random` class to initialize weights with a Gaussian distribution (mean 0, standard deviation 0.01). The learning rate is set to 0.01, and the implicit loss function resembles mean squared error, typical for binary classification with sigmoid outputs. No advanced optimizer (e.g., Adam) is used; a basic gradient descent update rule is applied.

- **Implementation Details**: The MLP is implemented in Java without external libraries, despite the assignment allowing a Python library. The program prompts the user for the seed value and file paths for the training and test datasets, ensuring reproducibility and flexibility. The implementation includes helper classes for data loading (`DataLoader.java`), utility functions (`Utils.java`), and the MLP itself (`MLPNetwork.java`), all compiled into a JAR file as required.

2.3 Decision Tree (J48)

The Decision Tree model was implemented using the J48 algorithm from the Weka package, as described by Singhal and Jena [2013]. The design specifications are:

- **Algorithm Configuration**: The J48 classifier is configured with a confidence factor of 0.25 for pruning, which controls the extent of tree pruning, and a minimum of 2 instances per leaf to prevent overly specific leaves. These settings balance model complexity and generalization.

- **Data Handling**: The training and test datasets are loaded from CSV files, with the last column set as the class attribute. Preprocessing includes replacing missing values with the mean (for numeric attributes) or mode (for nominal attributes) using Weka's `ReplaceMissingValues` filter. The class attribute is converted from numeric to nominal using the `NumericToNominal` filter, as J48 requires a nominal class. Additionally, the training data is balanced using the `Resample` filter to ensure a uniform class distribution, addressing potential class imbalance.

- **Parameters**: A user-specified seed value is used for reproducibility, applied to both the `Resample` filter and the J48 classifier to control randomness in class balancing and tree construction. The key parameters for J48 are the confidence factor (0.25) and the minimum number of instances per leaf (2).

- **Implementation Details**: The J48 model is implemented in Java using the Weka library. The program prompts the user for the seed value and file paths for the training and test datasets, ensuring reproducibility and flexibility. It is compiled into a JAR file, as required, and does not rely on an IDE for execution. The implementation leverages Weka's built-in functionality for data preprocessing and evaluation, including accuracy and F1 score computation.

3 Results

The performance of the three models was evaluated using accuracy and F1 scores on both training and test datasets across 10 runs with seeds 1234 to 1243. The results are presented in two tables: Table 1 shows the training metrics, and Table 2 shows the testing metrics. Each table reports the mean accuracy and F1 scores, with ranges in parentheses, for each model over the specified seed range. For J48, F1 score averages exclude runs with undefined (NaN) values due to degenerate predictions.

The 10-run analysis in Tables 1 and 2 reveals distinct performance trends across seeds 1234 to 1243. As shown in Table 1, MLP achieved a consistent mean training accuracy of 87.47% and F1 score of 0.8931, demonstrating stable learning across runs. J48 exhibited high variability, with a mean training accuracy of 67.36% (range 50.00%–97.60%) and a mean training F1 of 0.7713 (range 0.4450–0.9760, excluding NaN values), indicating sensitivity to random seeds. GP performed poorly, with a constant training accuracy of 53.21% and F1 of 0.64 across all runs, suggesting limited learning capability or implementation issues.

Table 2 highlights MLP's consistent superiority on the test set, with a mean F1 score of

Table 1: Mean Training Metrics for GP, MLP, and J48 Models (10 Runs, Seeds 1234–1243)

Model	**Seed Range**	**Training Acc**	**Training F1**
Genetic Programming	1234–1243	53.21% (53.21%–53.21%)	0.64 (0.64–0.64)
MLP	1234–1243	87.47% (87.47%–87.47%)	0.8931 (0.8931–0.8931)
Decision Tree	1234–1243	67.36% (50.00%–97.60%)	0.7713* (0.4450–0.9760)

* Excludes 4 runs with NaN F1 scores due to degenerate predictions.

Table 2: Mean Testing Metrics for GP, MLP, and J48 Models (10 Runs, Seeds 1234–1243)

Model	**Seed Range**	**Testing Acc**	**Testing F1**
Genetic Programming	1234–1243	66.12% (55.30%–70.45%)	0.767 (0.65–0.82)
MLP	1234–1243	96.40% (96.20%–96.58%)	0.9628 (0.9609–0.9650)
Decision Tree	1234–1243	57.38% (49.43%–71.86%)	0.5917* (0.3528–0.7186)

* Excludes 4 runs with NaN F1 scores due to degenerate predictions.

0.9628 (range 0.9609–0.9650) and mean test accuracy of 96.40% (range 96.20%–96.58

GP showed an unusual trend of higher test performance than training (mean 66.12% vs. 53.21% accuracy), possibly due to a distribution mismatch or an easier test set. MLP's stable test performance suggests robust generalization, while J48's high variability and degenerate predictions in several runs (e.g., seeds 1235, 1237) indicate overfitting and sensitivity to seed values. For stock prediction, MLP emerges as the most reliable model, supported by statistical evidence. The GP's poor performance and J48's inconsistency highlight areas for improvement, while the test-training discrepancy warrants further dataset analysis.

4 Statistical Analysis

To evaluate the significance of performance differences between the Genetic Programming (GP) and Multi-Layer Perceptron (MLP) models, a Wilcoxon signed-rank test was performed. The test used the test F1 scores from 10 runs with seeds 1234 to 1243, as the F1 score accounts for both precision and recall, making it suitable for evaluating classification performance.

4.1 GP vs. MLP

- **Methodology**: The Wilcoxon signed-rank test is a non-parametric statistical test used to compare two related samples. It assesses whether the median difference

between paired observations is zero, making it appropriate for comparing the performance of GP and MLP on the same test dataset. The test used the test F1 scores from 10 runs, with seeds 1234 to 1243, to generate paired observations. The differences between MLP and GP F1 scores were calculated, ranked by their absolute values, and the sum of ranks for positive and negative differences was computed. The test was two-tailed, with a significance level of $\alpha = 0.05$. The critical value for 10 pairs is 8.

- **Results**: The test F1 scores were as follows:
  - GP: [0.77, 0.82, 0.80, 0.68, 0.82, 0.78, 0.81, 0.72, 0.65, 0.82]
  - MLP: [0.9650, 0.9609, 0.9609, 0.9650, 0.9650, 0.9609, 0.9650, 0.9650, 0.9650, 0.9650]

  The differences (MLP - GP) were all positive: [0.195, 0.1409, 0.1609, 0.2850, 0.1450, 0.1809, 0.1550, 0.2450, 0.3150, 0.1450]. Ranks were assigned, and the sum of positive ranks (W^+) was 56, with no negative ranks ($W^- = 0$). The test statistic W, the smaller of W^+ and W^-, is 0. Compared to the critical value of 8, $W = 0$ indicates a significant result.

- **Interpretation**: With $W = 0 < 8$, the null hypothesis (that the median difference between GP and MLP F1 scores is zero) is rejected at $\alpha = 0.05$. This suggests a statistically significant difference, with MLP outperforming GP across the 10 runs.

4.2 Summary

MLP demonstrates statistically significant superiority over GP ($p < 0.05$), with a mean test F1 score of 0.9628 compared to 0.767 for GP. These results highlight MLP's effectiveness for this stock prediction task, while J48's inconsistent performance (mean test F1 of 0.5917, excluding NaN values) suggests sensitivity to seed values and potential overfitting.

5 Conclusion

This report presented the design and evaluation of three machine learning models—GP, MLP, and J48—for stock purchase classification. The 10-run analysis (seeds 1234–1243) showed that GP had poor performance, with a mean training accuracy of 53.21% and test accuracy of 66.12% (mean test F1 0.767), indicating limited learning capability. MLP demonstrated strong results, with a mean training accuracy of 87.47% and test accuracy of 96.40% (mean test F1 0.9628), making it the most effective model. J48 showed inconsistent performance, with a mean training accuracy of 67.36% and test accuracy of 57.38% (mean test F1 0.5917, excluding NaN values), reflecting overfitting and sensitivity to random seeds.

A Wilcoxon signed-rank test confirmed MLP's statistically significant superiority over GP ($p < 0.05$), highlighting its robustness and generalization ability. J48's high variability and degenerate predictions in several runs suggest it is less reliable, though no statistical comparison was performed with J48. The unexpected trend of higher test performance

for GP suggests potential dataset distribution issues, limiting the findings' real-world reliability.

Future work could explore: (1) addressing training-test distribution mismatches, (2) optimizing GP and J48 (e.g., tuning J48 pruning or balancing classes to avoid degenerate predictions), (3) ensemble methods, and (4) adding market features to enhance prediction.

6 Group Contributions

The project was completed by the following group members:

- Joshua Garner (u22502883): Responsible for implementing the Multi-Layer Perceptron model, as well as the design specification for the Multi-Layer Perceptron and the Statistical Analysis.
- Christopher Xides (u04979282): Responsible for implementing the Genetic Programming (GP) model, as well as the design specification for the Genetic Programming (GP).
- Shravan Seebran (u23684179): Responsible for implementing the Genetic Programming (GP) model, as well as the design specification for the Genetic Programming (GP).
- Kgosi Segale (u22534718): Responsible for implementing the Decision Tree (J48) model, as well as the design specification for the Decision Tree (J48).

References

T. Nyathi. Artificial intelligence cos314 – metaheuristics and computational intelligence. Lecture notes, 2025. University of Pretoria, Department of Computer Science.

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

S. Singhal and M. Jena. A study on weka tool for data preprocessing, classification and clustering. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 2(6):250–253, 2013.