

# Report

COS 314 Artificial Intelligence – Assignment 3

## **TEAM MEMBERS:**

Hlungwani T.

Makhene T.

Mapheto T.

Motsepe O.J.

# Introduction

The purpose of this assignment was to implement and evaluate Machine Learning models, namely: Genetic Programming (GP), Multi-Layer Perceptron (MLP) and Decision tree (DT) to predict (classification) whether a financial stock should be purchased based on historical data.

The models were each built with Java programming language ensuring no model was built using a superior language.

The models were trained and tested on the same input data, namely BTC\_test and BTC\_train, ensuring fairness across the models.

This report includes a detailed design specification for each model – including specific parameters used, as well as a results table to measure the accuracy (ACC) and F-Score of each model to evaluate their respective performance as well as a detailed Wilcoxon signed-rank test carried out between the GP and MLP to evaluate the significance of the differences in their respective performances.

# Genetic Programming (GP)

The GP model is built using Java with several classes namely:

## **Dataset:**

- Loads data from CSV file into an object where we can retrieve data and labels.

## **GPAlgorithm:**

Core Engine of GP model.

- Builds the GP algorithm using parameters mentioned below.
- Creates runs for Individuals and fills them into a list : populations.
- The algorithm initializes the population and evaluates the fitness using respective functions.
- Selects two parents.
- Applies cross over and mutation on the parents and replaces them with their child individuals in the population.

## **Individual :**

- Built using the Tree Object and the fitness evaluation.

## **GPNode:**

- Creates Nodes for the tree.
- Attributes a function to them.

## **Tree:**

- Generates a random expression tree using the GP nodes and allows for mutation, crossover and evaluation.

## **Main:**

Explained in detail below

# Design Specifications:

The main program executes as follows:

## **Input Collection**

Program asks user for input for:

- Seed value – for reproducibility.
- Absolute paths to training and testing CSV datasets.

## **Data loading**

- Dataset and Buffer Objects are used to load both datasets.

## **Model Configuration**

The GPAlgorithm is configured using the created GPAlgorithm object with the following parameters:

- Train Dataset Object
- Seed value.
- Population Size = 50
- Max Depth = 5
- Generations = 30
- Mutation Rate = 0,3
- Crossover Rate = 0,9
- TournamentSize = 5

### **Model Traininn**

A best Individual is then created from running GPAlgorithm

### **Model Evaluation**

- An Evaluation Function that uses the best individual and test Dataset as paremeters is used to compare the score of the individual and the predicated score.
- Data labels are also extracted from the test Dataset.

Metrics used are:

- **Accuracy:** Percentage of correctly classified instances.
- **F1 Score:** Weighted harmonic mean of precision and recall, especially useful for imbalanced datasets.

### **Results**

Results are outputed to terminal.

# Multi-Layer Perceptron (MLP)

Implemented in Java using the Weka machine learning library.

## Design Specifications:

The model executes briefly in the following steps:

### Input Collection

Program asks user for input for:

- Seed value – for reproducibility.
- Absolute paths to training and testing CSV datasets.

### Data loading

- Instance Objects are used to load data from dataset CSV files.
- The last column is set to the class index and converted to a nominal value

### Model Configuration

The MLP is configured using the Perceptron object with the following parameters:

- Seed value.
- The model sets Hidden layers- which prevents the MLP from modeling a linear relationship between data. The dimension parameters of the matrix are set to 10x10 indicating 10 hidden layer neurons and 10 input vectors to the hidden layer.
- The learning rate ( $\alpha$ ) is the parameter that controls the rate at which the model learns, is set to 0.3, this medium rate is to ensure steady convergence in the model.
- The training time variable which is used to denote the time the model should take to learn and find an optimal solution is set to 500ms.

### Model Training

Model is then trained using the buildClassifier() built in function to test the performance on both training and testing datasets.

### Model Evaluation

An Evaluation Object is created to evaluate the training and test data respectively.

Metrics used are:

- **Accuracy:** Percentage of correctly classified instances.
- **F1 Score:** Weighted harmonic mean of precision and recall, especially useful for imbalanced datasets.

### Results

Results are outputted to terminal.

# Decision Tree (DT)

Implemented in Java using the Weka machine learning library.

## Design Specifications:

The model executes briefly in the following steps:

### Data loading

- Instance Objects from Weka are used to load both dataset CSV files.
- The last column of “convert” variable – used to set format of training data is set to the class index and converted to a nominal value.
- Training data is filtered using the “convert” nominal variable.

### Model Configuration

The is is configured using the J48 Decision Tree Classifier Object with the following parameters:

- Enabled Pruning to remove branches which are not significant in finding an optimal solution.
- Confidence Factor of Pruning- used to control how much pruning is done to tree is set to 0.25f for moderate error-based pruning.
- Minimum instances per leaf – which specifies the number of traing instances a node must have to be split is set to 2.

### Model Training

Model is then trained using the buildClassifier() built in function with paramter of traing data.

### Model Evaluation

An Evaluation Object is created to evaluate the traing and test data respectively.

Metrics used are:

- **Accuracy**: Percentage of correctly classified instances.
- **F1 Score**: Weighted harmonic mean of precision and recall, especially useful for imbalanced datasets.
- **Confusion Matrix** : of evaluation object is created.

### Results

Results are outputted to terminal.

## Results

The results for each model were collected and tabulized in *figure 1*. The table represents the seed value used, accuracy(Acc) an F-score (Fi) for both training and test data for each model.

<b>Model</b>	<b>Seed Value</b>	<b>Training Acc (%)</b>	<b>F1</b>	<b>Test Acc(%)</b>	<b>F2</b>
GP	140233	65,33	0.6239	47.53	0.3551
MLP	141025	70.94	0.6759	95.82	0.9581
DT	140123	58.32	0,5452	57.03	0.5143

*Figure 1*

# Wilcoxon signed-rank test

As mentioned earlier, the test was carried out between the GP and MLP evaluate the significance of the differences in their respective performance (accuracy and f-score) across the same dataset.

This test was automated using the **wsr.py** program which uses python's built wilcoxon library under the the /wilcoxon signed-rank test/ folder

## Inputs:

accuracy\_gp = [0.7094, 0.9582, 0.7294, 0.9782, 0.6914, 0.8612]

accuracy\_mlp = [0.6533, 0.4753, 0.6033, 0.5053, 0.5733, 0.5142]

fscore\_gp = [0.6239, 0.3551, 0.6014, 0.3312, 0.6701, 0.4010]

fscore\_mlp = [0.6759, 0.9581, 0.6169, 0.9010, 0.6099, 0.9011]

**H<sub>0</sub>:** Median difference between GP and MLP performance (accuracy or F-score) is zero - neither method consistently outperforms the other.

**H<sub>1</sub>:** The median difference is **not** zero - there *is* a consistent difference between GP and MLP

## Results:

Accuracy Wilcoxon: stat=0.0, p=0.03125

F-Score Wilcoxon: stat=3.0, p=0.15625

### Accuracy:

- Wilcoxon Statistic: 0,0
- p-value: 0,03125
- Interpretation:  $p < 0,05$ , reject null hypothesis.

### F-Score:

- Wilcoxon statistic: 3.0
- p-value: 0.15625
- Interpretation:  $p > 0.05$ , we cannot reject the null hypothesis.

## Conclusion:

Based on the Wilcoxon signed-rank test, **GP shows significantly better performance than MLP in terms of accuracy**, but **there is no significant difference between the two methods in terms of F-score**. Therefore, while GP may offer better overall



classification accuracy, there is no confirmed statistical advantage in terms of balanced precision and recall (F-score).

# Conclusion

The performance of Genetic Programming (GP), Multi-Layer Perceptron (MLP), and Decision Tree (DT) models was evaluated using both training and test data. In addition to raw performance metrics, a Wilcoxon signed-rank test was conducted to statistically assess the differences between GP and MLP.

The results reveal the following:

- **Wilcoxon test on accuracy** showed a **statistically significant difference** ( $p = 0.03125$ ), indicating that **GP and MLP differ in classification accuracy**. The test accuracy values in Figure 1 support this, showing GP with **47.53%** and MLP with a notably higher **95.82%** accuracy.
- However, the **Wilcoxon test on F-score** did **not show a significant difference** ( $p = 0.15625$ ), suggesting that **GP and MLP have comparable performance in terms of F-score**. This is reflected in the test F-scores (F2) from Figure 1: **GP = 0.3551, MLP = 0.9581**. While the raw values differ, the variation across multiple seeds (as in the Wilcoxon test) was not consistent enough to conclude statistical significance.
- The Decision Tree model performed moderately, with a **test accuracy of 57.03%** and **F-score of 0.5143**, falling between GP and MLP in terms of accuracy, but outperforming GP on the F-score.