Programming Assignment 1: Research Report

Eli Freedman

October 5, 2025

Problem Description

This assignment focuses on building a predictive model for financial time series data using the daily closing prices of BHP Group Limited (ticker: $BHP). The objective is to forecast whether the next day's stock price will move up or down based on engineered features derived from historical data.

Data Preparation and Pipeline

Daily price and volume data were downloaded from Yahoo! Finance, covering the period from January 2000 through September 2025. The dataset included open, high, low, close, and volume observations from each trading day. After dropping irrelevant variables such as dividends and stock splits, multiple lagged features were constructed to ensure that all predictor variables used only historical information to avoid data leakage.

The feature-engineering process included building 16 new features [1]:

(0) **CloseLag1** Lag-one daily closing price
(1) **CloseLag2** Lag-two daily closing price
(2) **CloseLag3** Lag-three daily closing price
(3) **HMLLag1** Lag-one high minus low daily prices
(4) **HMLLag2** Lag-two high minus low daily prices
(5) **HMLLag3** Lag-three high minus low daily prices
(6) **OMCLag1** Lag-one open minus closing daily prices
(7) **OMCLag2** Lag-two open minus closing daily prices
(8) **OMCLag3** Lag-three open minus closing daily prices
(9) **VolumeLag1** Lag-one daily volume
(10) **VolumeLag2** Lag-two daily volume
(11) **VolumeLag3** Lag-three daily volume
(12) **CloseEMA2** Exponential moving average across two days
(13) **CloseEMA4** Exponential moving average across four days
(14) **CloseEMA8** Exponential moving average across eight days
(15) **CloseSMA5** Rolling 5-day mean of lagged close

(16) **NormVolumeLag1** Lag-one day volume normalized by 5-day mean

Each of these features were then rounded to the nearest three decimal places and normalized using a standard scaler. Standardization of feature values is imperative to prevent the model from weighing features that have larger scales more significantly than other features.

Research Design

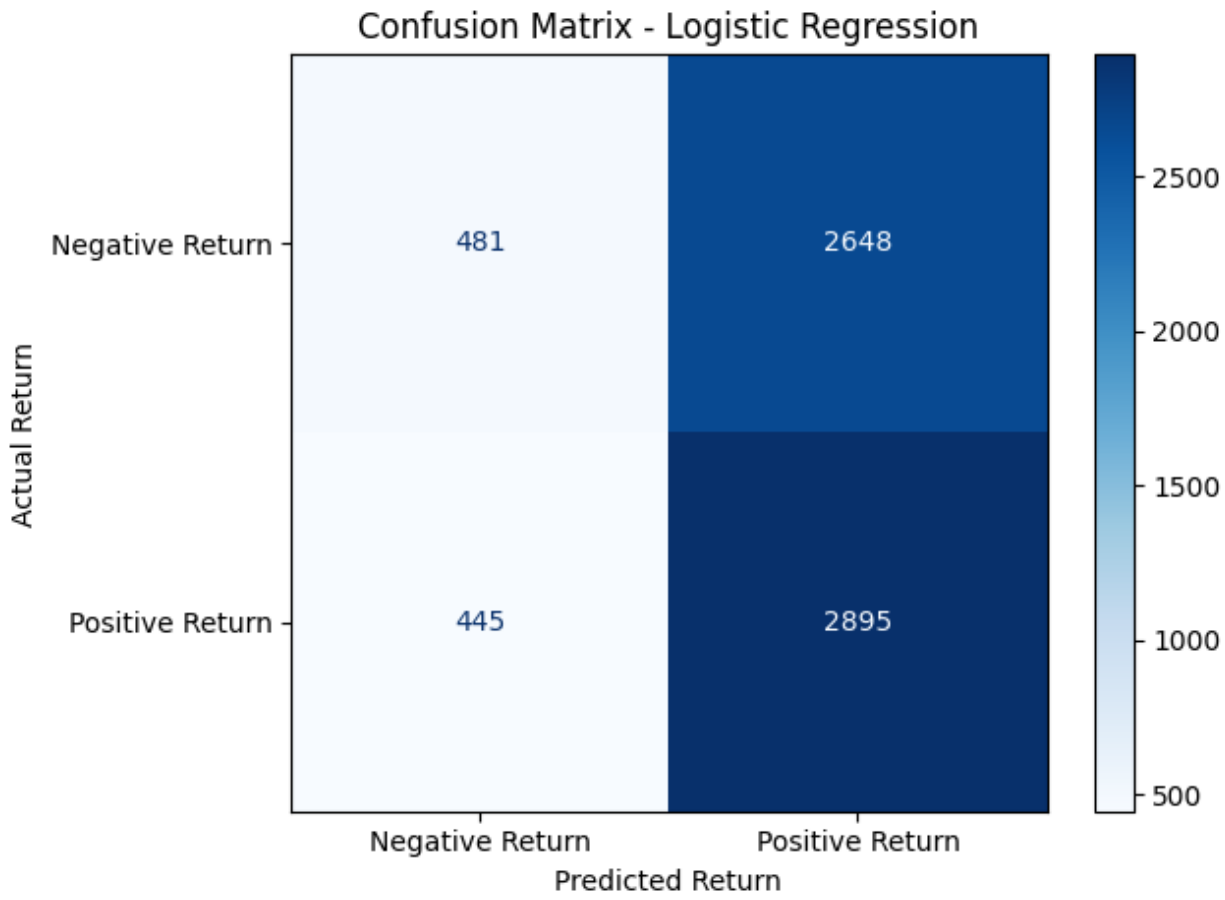
To predict the next day returns for $BHP, I employed two approaches: Logistic Regression and XGBoost.

I started with logistic regression because it is a simple and easily understood model. It is often used as a baseline since it provides a clear view of how each feature affects the target variable. Logistic regression assumes a linear relationship and works well for identifying which features are most useful for predicting next day's returns.

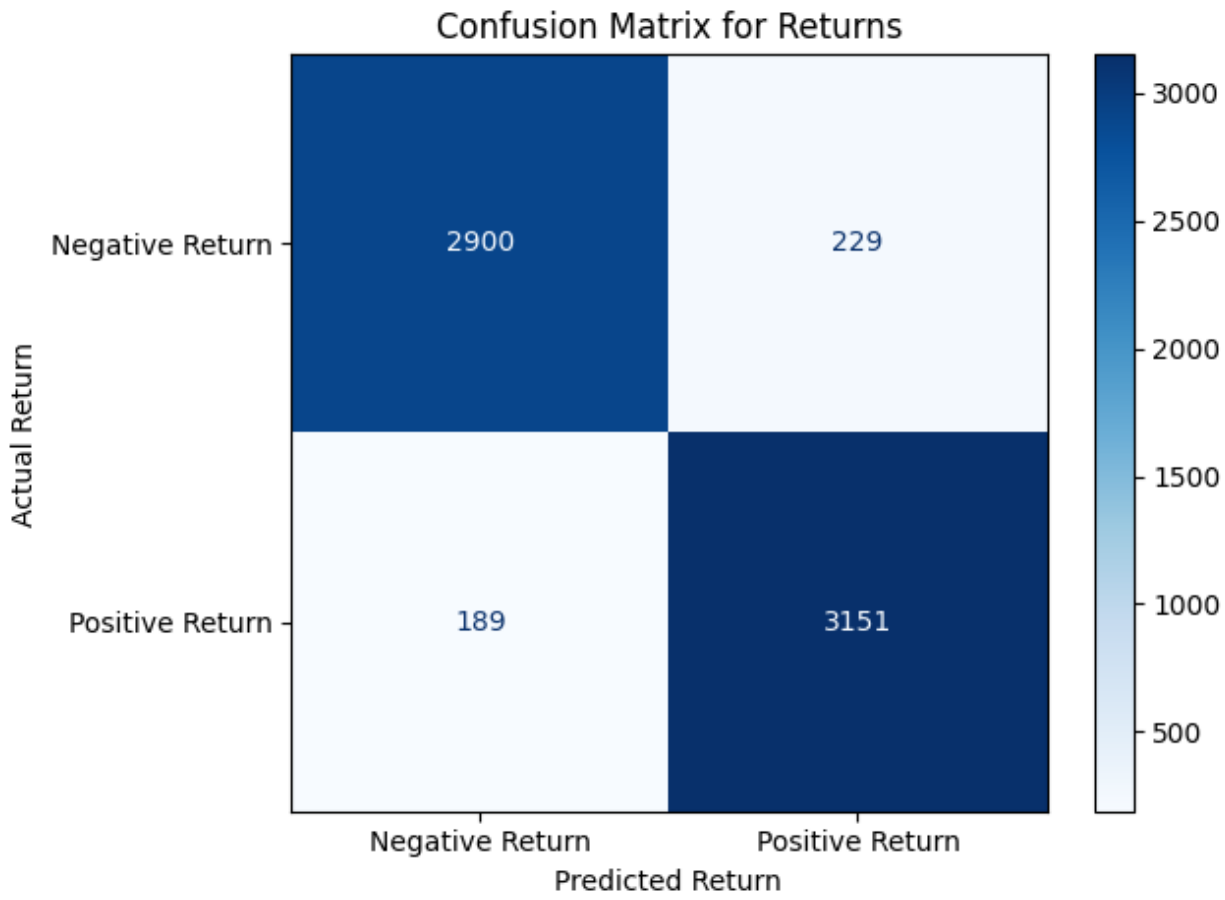
After building the logistic regression model, I added XGBoost to test whether a more advanced model could improve performance. XGBoost is a robust ensemble method that can capture nonlinear relationships and complex patterns that logistic regression might miss. It also includes built-in tools for feature importance, which helps understand which inputs have the most influence on predictions.

By using both models, I could compare a simple baseline approach with a more flexible and powerful one. This helped me evaluate whether the added complexity of XGBoost provided meaningful improvements in prediction accuracy.

Through reflection of the two models, the logistic regression model performed poorly, achieving an overall accuracy of 52%. The model struggled to distinguish between positive and negative next-day returns. It correctly identified positive returns more often (recall of 0.87) but performed poorly on negative returns (recall of 0.15). This imbalance suggests that the model leaned toward predicting positive outcomes, possibly due to class imbalance or the linear nature of the model. The F1 scores of 0.24 for class 0 and 0.65 for class 1 indicate that the model had limited predictive power overall. As a baseline, however, logistic regression provided a useful reference point for evaluating more advanced models.



In contrast, the XGBoost model achieved much stronger performance, with an accuracy of 94% and nearly identical precision and recall across both classes. The model's precision and recall values (around 0.93–0.94) show that it consistently made correct predictions for both positive and negative returns. This indicates that XGBoost was able to capture more complex relationships in the data, likely due to its ability to handle nonlinear patterns. The balanced performance across both classes also suggests the model generalized well and avoided the bias observed in logistic regression.



Programming

I utilized an assortment of Python packages for this assignment:

NumPy: handles math and arrays [2]
Polars: fast tool for working with tabular data (Pandas replacement) [3]
Matplotlib: basic charts and plots [4]
Seaborn: nicer plots [5]
Scikit-learn: builds and tests machine learning models [6]
XGBoost: specific library for the XGBoost model [7]
yfinance: downloads stock market data from Yahoo! [8]

Exposition

In this project, I built a complete machine learning workflow to predict next-day stock returns for $BHP. I prepared the data by cleaning and transforming daily price information from Yahoo! Finance, then engineered several lagged and rolling features to capture short-term trends while

avoiding data leakage. I implemented and tested both logistic regression and XGBoost models, comparing their performance to understand the impact of model complexity on predictive accuracy.

The logistic regression model achieved an overall accuracy of 52%, with a recall of 0.87 for positive returns and 0.15 for negative returns. These results showed that the model tended to overpredict positive outcomes and struggled to capture market reversals. In contrast, the XGBoost model reached an accuracy of 94%, with precision and recall values around 0.93–0.94 for both classes. This demonstrated that XGBoost was far more effective at identifying patterns in the data and producing balanced predictions. Together, these results highlight the trade-off between interpretability and performance, showing that while logistic regression offers a clear baseline, XGBoost provides a stronger and more reliable approach for forecasting financial time series.

References

[1] Professor Miller's **451_pa1_jumpstart_v001.ipynb** code

[2] NumPy. https://numpy.org/.

[3] "Polars." Polars -. https://pola.rs/.

[4] "Visualization with Python." Matplotlib. https://matplotlib.org/.

[5] "Statistical Data Visualization#." seaborn. https://seaborn.pydata.org/.

[6] "Learn: Machine Learning in Python - Scikit-Learn 0.16.1 Documentation." scikit. https://scikit-learn.org/.

[7] "XGBoost Documentation." XGBoost Documentation - xgboost 3.0.5 documentation. https://xgboost.readthedocs.io/.

[8] "Yahoo Finance - Stock Market Live, Quotes, Business & Finance News." Yahoo! Finance. https://finance.yahoo.com/.