Prof. Indro Spinelli
Foundations of Data Science Sapienza University of Rome
29 December 2024

Using Machine Learning Models to Predict Stock Returns for Apple Inc.

Saif Ali

Abstract

This project explores the application of machine learning models to predict quarterly stock returns for Apple Inc., aiming to inform investment decisions. *The research is done from the perspective of an inexperienced investor who prefers a buy and hold approach to build wealth*. The study focuses on determining whether stock returns will be positive or negative in the upcoming quarter, providing actionable signals for "**Buy**" or "**Not Buy**" decisions. Thus, our methodology diverges from conventional approaches that predict stock prices or short-term returns, emphasizing a risk-averse strategy to hedge against inflation. By prioritizing **precision** and **recall** for the "Buy" signal, the framework minimizes false positive predictions, ensuring investments are aligned with a high probability of positive returns.

The analysis incorporates a diverse set of machine learning models—Logistic Regression, Support Vector Machines, XGBoost, and Neural Networks—to evaluate their predictive efficacy. The choice of such a collection of models is motivated by the fact that the financial market is a complex, evolutionary, and non-linear dynamical system. The field of financial forecasting is characterized by data intensity, noise, non-stationary, unstructured nature, high degree of uncertainty, and hidden relationships. Many factors interact in finance including political events, general economic conditions, and traders' expectations. Thus, only linear classifiers, like Logistic Regression and SVM, may not be adequate for the non-linear patterns in the data.

Features for our dataset include historical **quarterly** stock data, company financials, and U.S. macroeconomic indicators, Models are assessed using key classification metrics, with a focus on precision and recall for the positive class. While the dataset's **limited size**, due to quarterly data, constrains the generalizability of results, the study underscores the potential of machine learning to enhance investment strategies. Our findings suggest that precision-driven models can serve as effective tools for inflation-hedged investment decision-making, promoting a balance between risk and return.

Dataset and Features

We utilized Yahoo Finance and Alpha Vantage APIs to compile a dataset spanning from 2009 to the end of 2024 (60 quarters). This range was selected to ensure the availability of complete, high-quality quarterly data without missing values. The dataset encompasses a combination of stock-specific metrics, company financial indicators, and macroeconomic variables, reflecting the multifaceted influences on stock market performance.

The following features were gathered for each stock:

- **Stock Closing Price**: The latest closing price for each quarter, reflecting the market valuation of the stock.

- **Average Volume**: The average number of shares traded during each quarter, indicating the liquidity and investor interest in the stock.
- **Reported Earnings Per Share (EPS)**: A key metric of profitability reported quarterly, showcasing a company's ability to generate earnings.
- **Total Revenue**: A measure of the company's ability to generate sales.
- **Net Income**: Reflecting the company's overall profitability after expenses.
- **US GDP**: Representing the economic growth or contraction of the US economy on a quarterly basis.
- **Federal Funds Rate**: The benchmark interest rate set by the Federal Reserve, influencing borrowing costs and investment flows.
- **Consumer Price Index (CPI)**: An indicator of inflation and purchasing power within the economy.

These features were selected based on the well-established theory that stock market performance is influenced by a combination of company-specific fundamentals and broader economic conditions. Stock price, EPS, and net income provide insights into investor confidence and financial health, while GDP, CPI, and the Federal Funds Rate capture the economic environment's impact on market dynamics.

To facilitate model development, we introduced a derived binary target variable, **Buy**, which indicates whether a stock is expected to deliver positive returns in the subsequent quarter. This variable was engineered based on the underlying features and serves as the primary focus of our predictive analysis. The dataset is imbalanced, for it contains 70% Buy signals and 30% Not Buy signals. However, it reflects a realistic representation of market conditions, while mitigating the risk of bias in the model's predictions.

Initial Feature Set and Model Testing vs Engineered Feature Set and Model Testing

Before delving into feature engineering, we first trained and evaluated our models using the initial set of features as described earlier. This approach allowed us to establish a baseline performance and assess how well the original predictors could capture the relationship between the input variables and the target variable, Buy. *This decision was motivated by the understanding that adding more features does not necessarily improve model performance*, as the complex relationships between predictors and the target variable, uncovered during Exploratory Data Analysis (EDA), may diminish model's performance. Moreover, to evaluate the effectiveness of the machine learning models, their performance was compared against the accuracy of a **Random Walk model**. The Random Walk model predicts the next value of a time series as the current value plus a random fluctuation (we chose a standard deviation of 3 and 4 respectively). Serving as a baseline, the Random Walk model achieved an accuracy of approximately 53%, akin to tossing a coin and making investment decisions based on the outcome. This benchmark highlighted whether the machine learning models could outperform random guessing.

After evaluating the initial feature set, we then undertook feature engineering to enhance the dataset and further refine the model's predictive power. We introduced the following feature:

- **PERatio**: The Price-to-Earnings (P/E) ratio was computed and added as an additional feature. The P/E ratio is a widely recognized metric in financial analysis, providing insight into market sentiment and valuation.

Then, we transformed the complete set of existing features. Rather than using absolute values for each feature, we computed the *relative changes from the previous quarter* as inputs to the model. For example, instead of using the absolute quarterly `Net Income`, we calculated the percentage change in Net Income relative to the prior quarter.

This transformation was quite intuitive for reasons such as *normalization* and *trend removal*. More importantly, it aids in capturing *directional movements*. Predicting the sign of future stock price changes (Buy or Not Buy) requires an emphasis on directional movements rather than raw magnitudes, which relative changes effectively highlight.

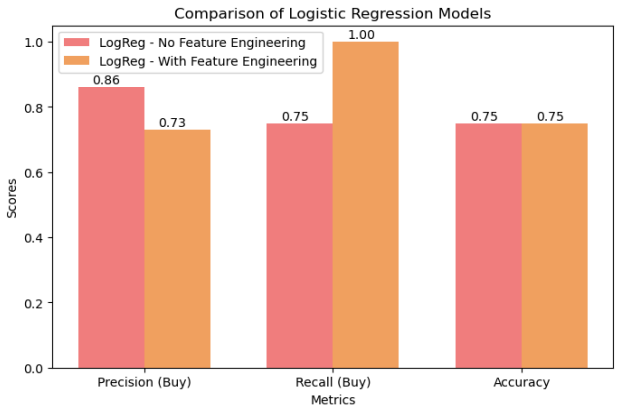
With these enhancements, the models were re-trained and re-evaluated. The results, demonstrated below, show how the engineered features impacted model performance. These findings underscore the iterative nature of machine learning workflows, where initial insights guide successive refinements for enhanced outcomes.

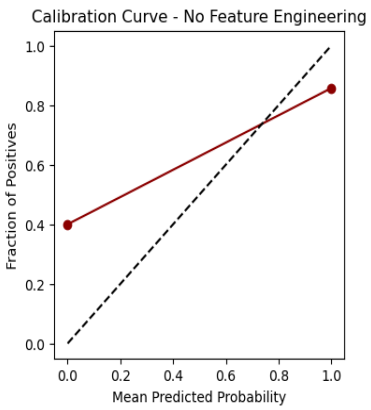
At each stage, the dataset was split into **80% training** and **20% testing** subsets. Models were optimized using **5-fold cross-validation** on the training set, focusing on Precision and Recall to determine the best hyperparameters. This ensured robust evaluation and alignment with the project's performance objectives.

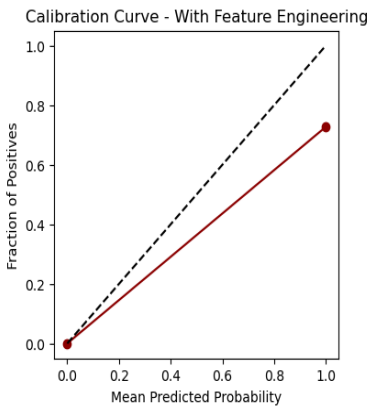
Model 1: Logistic Regression

Logistic Regression, a basic binary classification model, was deployed for evaluating the relationship between the features and the binary target variable, Buy. Initial results using the unengineered feature set revealed strong precision (0.86) but lower recall (0.73), indicating the model's ability to accurately identify positive signals (Buy) while missing some. After applying feature engineering—introducing the P/E ratio and using relative changes—the precision slightly decreased to 0.75, while recall improved significantly to 1.00, reflecting a better balance between precision and recall.

Calibration curves demonstrate an improvement in probability estimates with feature engineering, aligning more closely with the ideal diagonal, signifying better-calibrated predictions. Overall, these results highlight the benefits of feature engineering in improving the model's recall and predictive reliability while maintaining competitive precision.







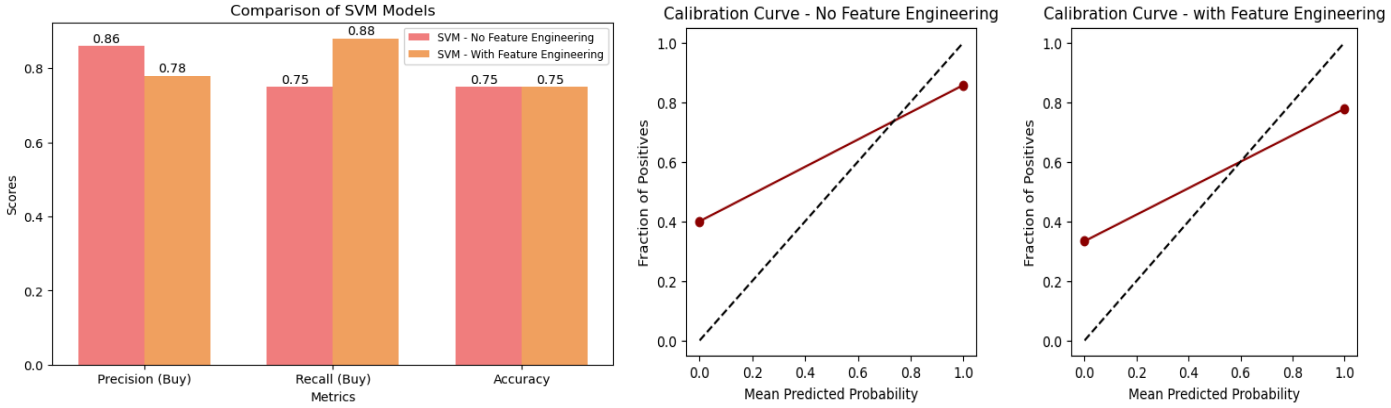
Model 2: Support Vector Machine

The Support Vector Machine (SVM) model was also employed to classify the binary target variable, Buy. Without feature engineering, the SVM achieved a precision of 0.86 and a recall of 0.75, indicating that the

model effectively captured positive signals but could benefit from improved recall. The overall accuracy was 0.75.

After incorporating feature engineering, the precision slightly decreased to 0.78, but recall increased to 0.88, resulting in an unchanged accuracy of 0.75. This indicates that feature engineering did not significantly alter the performance metrics but helped maintain a consistent balance between precision and recall.

Calibration curves suggest that feature engineering slightly improved the model's probability estimation, aligning predictions closer to the ideal diagonal. This ensures that the predicted probabilities better reflect the true likelihood of positive outcomes. Overall, SVM demonstrates robust and reliable classification performance, with calibration benefiting from feature engineering.

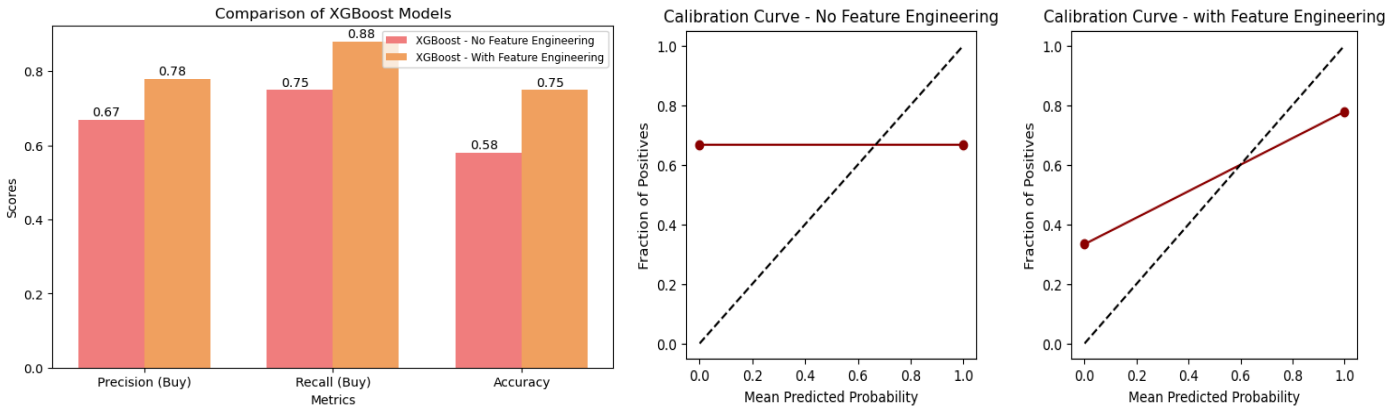


Model 3: XGBoost (Extreme Gradient Boosting)

The XGBoost model demonstrated notable differences in performance depending on whether feature engineering was applied. Without feature engineering, the model achieved a precision of 0.67 and a recall of 0.75, resulting in an accuracy of 0.58. This indicates that while the model was able to capture a reasonable number of positive cases, its overall classification accuracy was suboptimal.

With feature engineering, the model's performance improved significantly. Precision increased to 0.78, recall remained consistent at 0.75, and accuracy rose to 0.75. This demonstrates that feature engineering substantially enhanced the model's ability to balance precision and recall, leading to a more reliable classifier.

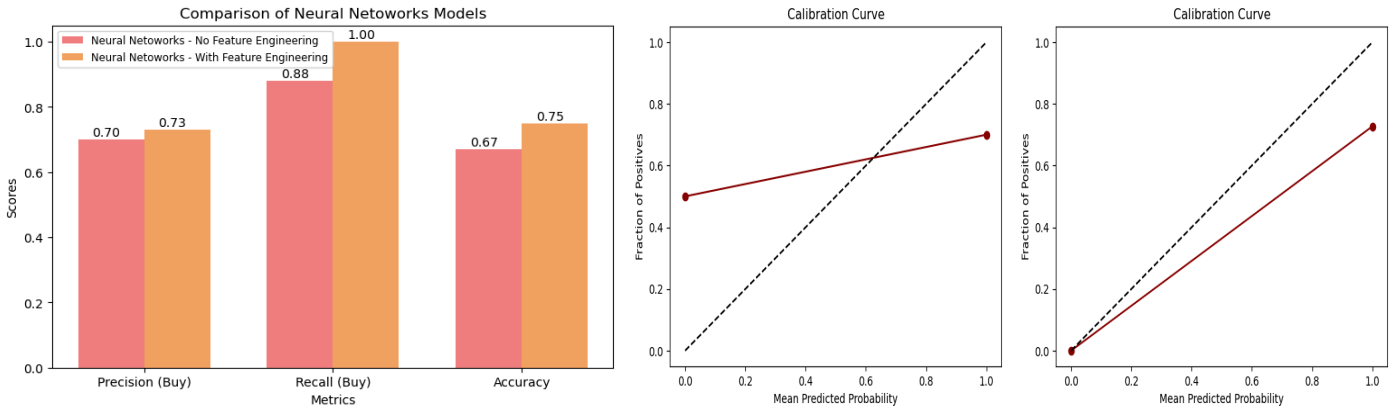
The calibration curve without feature engineering reveals that the model's probability estimates were poorly aligned with the true probabilities, as shown by the horizontal line far from the ideal diagonal. After feature engineering, the calibration improved, with predicted probabilities aligning closer to the diagonal. This highlights the importance of feature engineering in enhancing not only performance metrics but also the interpretability and reliability of probability estimates.



Model 4: Neural Network Model

Neural networks were employed to capture non-linear relationships between the target variable and features. For unengineered data, a deeper architecture with four hidden layers (128-64-32-16 neurons) was used, but performance was limited, with precision of 0.67, recall of 0.75, and accuracy of 0.58. The calibration curve, almost being a horizontal line, showed poor probability estimates, with predictions far from the ideal diagonal.

With feature engineering, a more compact three-layer architecture (64-32-16 neurons) achieved substantial improvements: precision rose to 0.73, recall reached 1.00, and accuracy improved to 0.75. The calibration curve showed better alignment, particularly in the lower probability ranges, highlighting more reliable probability estimates. Batch normalization and dropout layers were used in both setups to manage overfitting and improve generalization. This demonstrates the impact of feature engineering and model simplification in improving predictive performance and calibration.



CONCLUSION

The analysis compared four machine learning models (Logistic Regression, SVM, XGBoost, and Neural Networks) before and after applying feature engineering. Feature engineering markedly enhanced the performance of XGBoost and Neural Networks, highlighting the importance of preprocessing in improving model outcomes. A clear pattern emerged: while all models demonstrated very high recall on the engineered dataset, precision—the primary metric of interest—showed mixed results. Some models, like Logistic

Regression and SVM, even experienced a decline in precision, underscoring the trade-offs between recall and precision in certain scenarios.

Among all models, XGBoost on the engineered dataset stood out, achieving an impressive precision of nearly 80% and a recall of almost 90%. This combination makes XGBoost particularly effective for identifying positive return opportunities, striking a balance between precision and recall. Neural Networks also showed substantial improvement but fell slightly short of XGBoost's performance in terms of precision.

Overall, the results affirm the potential of machine learning models, particularly XGBoost, in predicting next-quarter returns with strong precision and reliability. An inexperienced investor, who prefers a buy and hold approach to build wealth, can certainly be confident with using XGBoost and historical data for his investment decisions. However, the dataset's limited size—spanning only 60 quarters—restricts the statistical generalizability of these findings. To ensure robustness and scalability, future research should focus on expanding the dataset and exploring more diverse market conditions. This would not only validate the current findings but also enable the development of even more effective predictive models for financial forecasting.

Remark: This report only extracts the insights about the models we used. For a detailed review, including EDA and preprocessing, model architectures and hyper parameters, kindly refer to the jupyter notebook linked in the resources section.

References, Resources, and Related Work

- **Yahoo Finance**: Data sourced using the Yahoo Finance API via the Python package yfinance.
- **Alpha Vantage**: Supplementary data collected from Alpha Vantage.
- **Stanford's CS229**: Special thanks to Stanford University and Anand Avati for providing practical insights through the CS229 Machine Learning course.
- **Research Papers and Articles**:
    - *"A machine learning-based stock trading framework using technical and economic analysis"*, Naveed Zaman et al., 2017, Stanford.
    - David R. Harper, *"Forces That Move Stock Prices"*, Investopedia.
    - W. Huang et al., *"Forecasting stock market movement direction with support vector machine"*, *Computers & Operations Research*, 32(10), 2005, pp. 2513–2522.
    - Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, Ruben Gryp, *"Evaluating multiple classifiers for stock price direction prediction"*, *Expert Systems with Applications*, Volume 42, Issue 20, 2015, Pages 7046-7056.
- **Jupyter Notebook**: Access the complete analysis and code implementation through the provided link: View Jupyter Notebook.