ML-Based Portfolio Optimization

Final Project Report

Team Bull Traders

Daksh Soni • Rahul Gadhavi • Aiswaryaa Velumani

`Github Repository`

1. Introduction

Financial markets move quickly and are influenced by many factors, making short-term stock behavior difficult to predict. Traditional portfolio strategies often rely on historical patterns that may not capture these fast, non-linear changes.

This project explores whether machine learning models can extract useful signals from historical stock and macroeconomic data to predict next-day stock movement.

We also build a simple trading strategy using the model outputs and compare its performance with the S&P 500 benchmark to evaluate whether ML-driven decisions can improve portfolio returns.

2. Data Description and Preparation

This project uses more than 20 years of historical financial data, collected from large U.S. financial companies and key macroeconomic indicators.

2.1. Data Sources

We gathered daily closing prices for major financial stocks such as JPMorgan Chase, Bank of America, Goldman Sachs, and American Express. The S&P 500 ETF (SPY) was included as a benchmark to compare portfolio performance. To capture broader market conditions, we added several widely used macroeconomic indicators:

- VIX (market volatility index),
- 10-year U.S. Treasury yield,
- Crude oil prices,
- Gold prices,
- U.S. dollar index.

All data was downloaded through the `yfinance` Python library, which provides long-term historical data in a structured format.

2.2. Data Cleaning

Because the data came from different assets with slightly different trading calendars, several cleaning steps were required:

- **Forward-fill missing values:** When one asset was closed and another was trading, we filled the missing values using the previous day's price.
- **Align dates:** We ensured that each row represented the same trading day across all stocks and macro indicators.
- **Remove rows created by lag features:** Creating lagged columns introduced missing values at the start of the dataset. These rows were removed.

2.3. Feature Engineering

To help the ML models learn meaningful patterns, we created several features commonly used in quantitative finance:

- **Daily returns** to capture day-to-day price movement,
- **Lagged returns** to provide context from the previous trading day,
- **Lagged macroeconomic indicators** to reflect market sentiment and stress,
- **Moving averages (MA5, MA10)** to show short-term price trends,
- **Rolling volatility** (5-day window) to indicate recent market risk.

3. Exploratory Data Analysis

Exploratory Data Analysis (EDA) helped us understand how the financial stocks and macroeconomic indicators behaved before building any machine learning models.

3.1. Long-Term Price Trends

The long-term price trend plot shows how the major financial stocks moved over more than two decades.

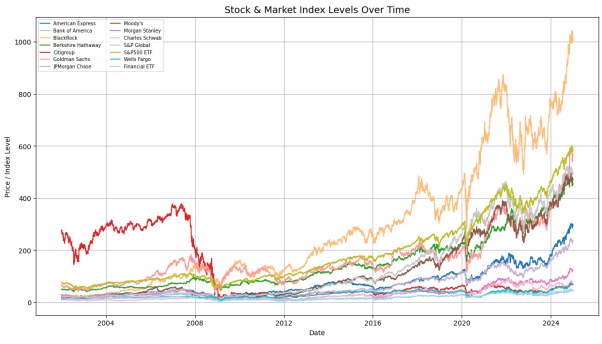

Figure 1: Long-term price trends of major financial stocks (2002–2025). (by Daksh Soni)

3.2.Correlation Between Stocks and Macro Indicators

The correlation heatmap provided insight into how stocks relate to each other and to broader economic factors.

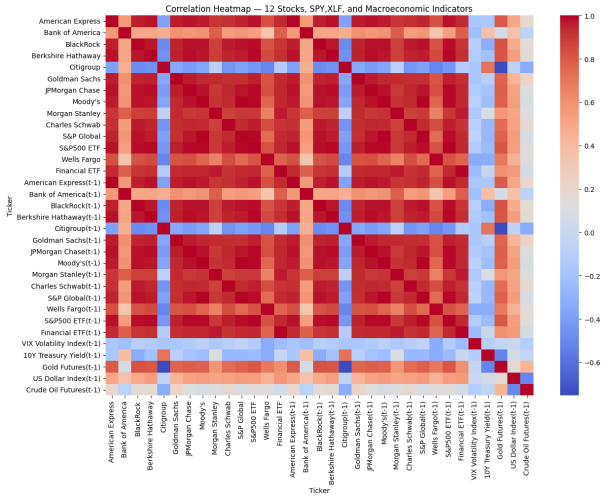

Figure 2: Correlation heatmap showing relationships between financial stocks, SPY, and macro indicators (by Aiswaryaa Velumani)

3.3. Rolling Volatility (20-Day Window)

We examined rolling volatility to understand how market risk changed over time.

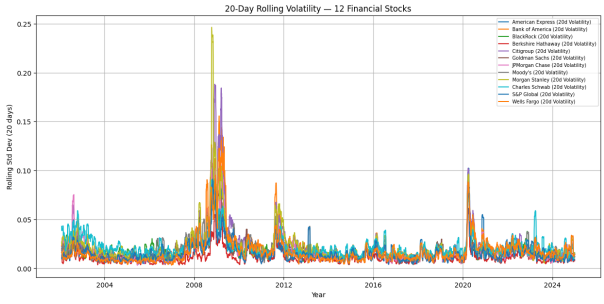

Figure 3: 20-day rolling volatility for selected financial stocks. (by Rahul Gadhavi)

3.4. KDE Distribution of Daily Returns

The KDE plot shows the distribution of daily returns for all financial stocks, highlighting how most returns cluster tightly around zero.

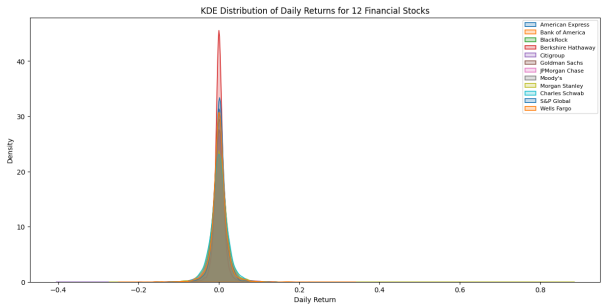

Figure 4: KDE distribution of daily returns for financial stocks. (by Rahul Gadhavi)

3.5. Pairwise Return Relationships

The pairplot illustrates pairwise relationships between daily returns, showing positive correlations and similar movement patterns across financial stocks.

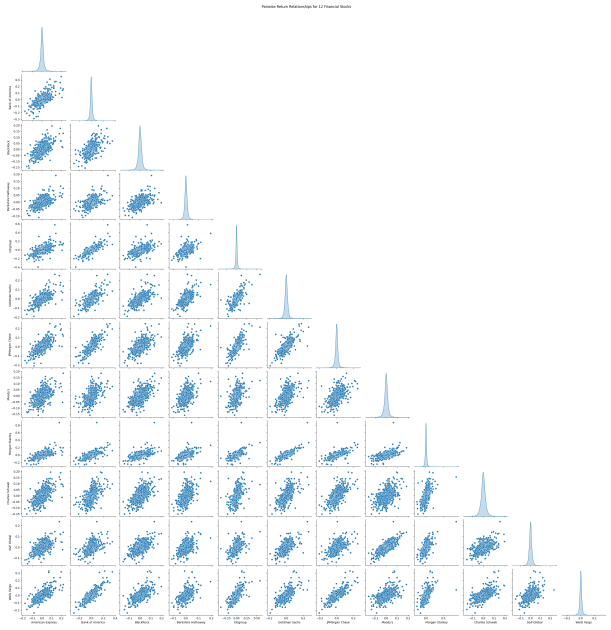

Figure 5: Pairwise scatterplots and distributions of daily returns for financial stocks. (by Aiswaryaa Velumani)

4. Machine Learning Models

To evaluate whether machine learning can help predict next-day stock direction, we trained models using the engineered features from our dataset. We used a 5-fold expanding TimeSeriesSplit, where each fold trains on past data and tests on a future period.

4.1. Majority-Class Baseline (Implemented by Aiswaryaa Velumani)

Before using ML models, we built a simple baseline by checking whether each stock had more "up" days or "down" days in each fold. This baseline predicts the majority class for that fold.

- Accuracy ranged around 50–53%, depending on the fold and stock.
- This showed that even a trivial model performs reasonably well in daily return prediction.

4.2. Random Forest Classifier (Implemented by Rahul Gadhavi)

Random Forest was our first non-linear model. It works by averaging predictions from many decision trees, which helps reduce noise and capture interactions between features.

- Accuracy varied slightly across folds but generally stayed near the baseline.
- The model captured relationships between features such as volatility and lagged returns.

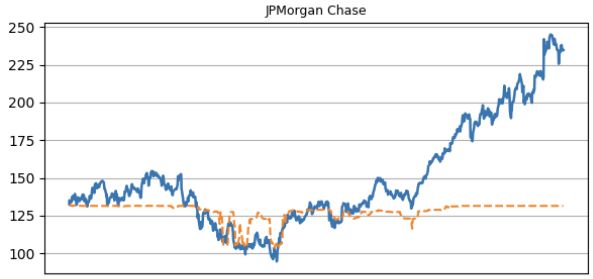

Figure 6: Random Forest predictions for JPMorgan Chase (by Rahul Gadhavi)

4.3. XGBoost Classifier (Implemented by Aiswaryaa Velumani)

XGBoost builds trees sequentially, correcting errors at each step, which makes it effective for noisy financial data.

- Captured non-linear interactions more effectively.
- Regularization helped prevent overfitting across folds.

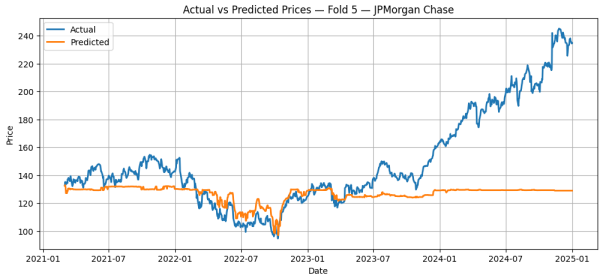

Figure 7: XGBoost predictions for JPMorgan Chase (by Aiswaryaa Velumani)

4.4. Linear Regression (Implemented by Daksh Soni)

Linear Regression produced stable numeric predictions that were easy to translate into buy/sell signals for portfolio testing.

- Provided consistent outputs suitable for generating trading signals.
- Less sensitive to noise compared to tree-based models.

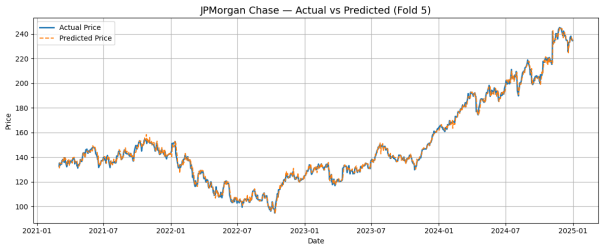

Figure 8: Linear Regression predictions for JPMorgan Chase (by Daksh Soni)

5. Results

This section presents the performance of the machine learning models and the outcome of the portfolio backtesting experiment.

5.1. Backtesting Method

We used the predictions from the Linear Regression model to simulate a simple daily trading strategy. The model generated a predicted return for each stock, which we used to create trading signals.

Trading rules:

- Buy a stock if the predicted next-day return was positive.
- Avoid holding the stock if the predicted return was negative.
- Allocate capital equally across the top 5 stocks with the highest positive predictions.
- Begin with an initial portfolio value of $100.

5.2. Portfolio Performance

We compared the ML-driven portfolio to a buy-and-hold S&P 500 (SPY) benchmark.

Final values:

- **ML portfolio:** $176.85
- **SPY benchmark:** $155.60

The ML portfolio achieved a higher cumulative return than the benchmark in the evaluated fold.

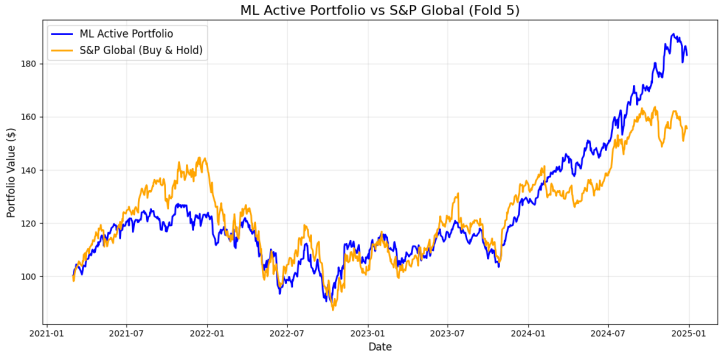

Figure 9: Cumulative returns for the ML-driven portfolio compared to the SPY benchmark. (by Daksh Soni)

Although predictive accuracy remained close to chance due to market noise, Linear Regression provided the most stable output for portfolio simulation.

6. Conclusion

Overall, this project showed that predicting daily stock movements is extremely challenging, but even small and inconsistent signals from machine learning models can still improve portfolio performance when used within a simple trading framework. The Linear Regression–based strategy demonstrated that structured decision rules can sometimes outperform a passive benchmark. These results highlight both the limitations and the practical potential of ML-based portfolio optimization.