Stock Price Prediction Analysis Report

A Comprehensive Data Science Pipeline

Submitted by: Saksham Maurya

Date: March 21, 2025

Submitted to: Invsto Analysis Team

1 Introduction

This report presents a detailed analysis of stock price prediction using advanced data science techniques, focusing on two distinct modeling approaches: the Autoregressive Integrated Moving Average (ARIMA) model for time-series forecasting and the Gradient Boosting method (via XGBoost) for regression-based predictions. The study targets Apple Inc. (AAPL) stock data, aiming to provide actionable insights for the Invsto Analysis Team. The comprehensive workflow includes data preparation, exploratory data analysis (EDA), feature engineering, model development, and thorough evaluation. This effort seeks to identify the most effective model for predicting stock prices, considering both short-term trends and complex market dynamics, to support informed trading strategies as of March 21, 2025.

2 Data Preparation

2.1 Data Source

The foundation of this analysis is high-quality historical stock data sourced from Yahoo Finance using the `yfinance` library. The dataset includes daily price and volume metrics for multiple prominent equities: Apple Inc. (AAPL), Microsoft Corporation (MSFT), Alphabet Inc. (GOOGL), Amazon Inc. (AMZN), and Tesla Inc. (TSLA). Spanning from January 1, 2015, to December 31, 2024, this nearly ten-year period captures significant market events, trends, and volatility, offering a robust basis for modeling and forecasting stock price movements. For this report, the focus is on AAPL data, saved as "Apple_Data.csv" for analysis.

2.2 Libraries Used

This project employs a suite of powerful Python libraries tailored for data science and financial analysis:

- `pandas`: Facilitates efficient data manipulation and structuring, enabling seamless handling of time-series data.
- `numpy`: Provides fast numerical computations, essential for statistical analysis and feature calculations.
- `yfinance`: Enables direct access to Yahoo Finance data, ensuring accurate and up-to-date stock information.
- `statsmodels`: Supports sophisticated time-series modeling, particularly for ARIMA implementation and stationarity tests (e.g., ADF test).
- `scikit-learn`: Offers tools for machine learning, including data splitting (`train_test_split`), hyperparameter tuning (`RandomizedSearchCV`), and performance metrics (e.g., RMSE, MAE).
- `xgboost`: Implements the Gradient Boosting algorithm (XGBRegressor), known for its robustness in predictive modeling.
- `matplotlib` and `seaborn`: Deliver high-quality visualizations (e.g., plots, heatmaps) to illustrate trends and model outcomes.

2.3 Data Cleaning

Ensuring data integrity is critical for reliable predictions. The AAPL dataset was inspected for missing values using `stock_data.isnull().sum()`, revealing no gaps (all counts: 0). As a precautionary measure, a forward fill method (`stock_data.fillna(method='ffill')`) was prepared to propagate the last observed value forward if needed, maintaining a consistent dataset essential for ARIMA's sequential nature and Gradient Boosting's feature-based predictions.

3 Exploratory Data Analysis (EDA)

3.1 Closing Price Trends

The EDA phase began with an in-depth examination of AAPL's historical closing prices using `plt.plot(stock_data['Close'])`. From January 2015 to December 2024, the analysis revealed periods of steady growth interspersed with volatility, reflecting market reactions to economic events, product launches, and broader financial conditions. The mean closing price was \$94.18, with a standard deviation of \$65.67, ranging from \$20.67 to \$258.74. Visualizing these trends (Figure 1) provides a foundational understanding of price behavior, crucial for selecting appropriate modeling techniques.

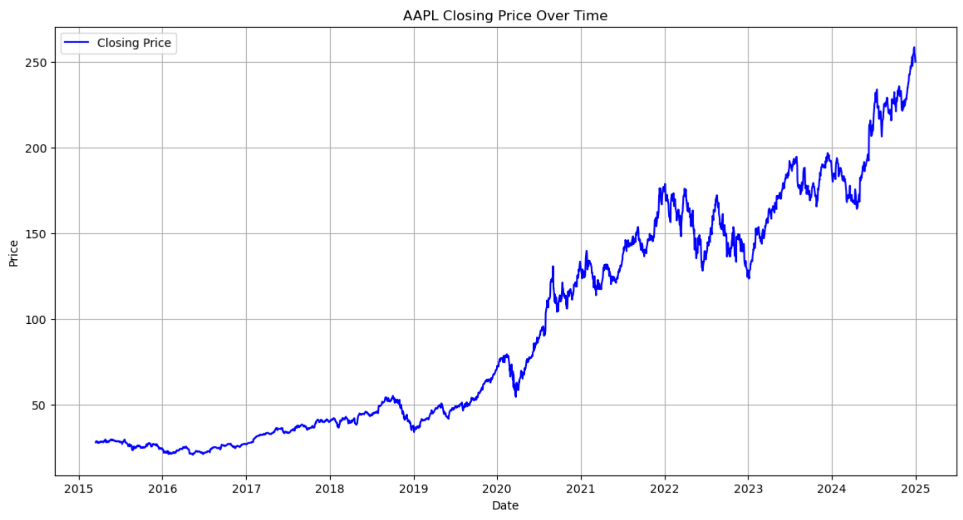

Figure 1: AAPL Closing Price Over Time

3.2 Moving Averages

To distill longer-term trends, 20-day and 50-day moving averages were calculated using `stock_data['Close'].r` and `window=50`. These rolling means smooth out short-term noise, highlighting sustained movements. For AAPL, crossovers between these averages often signal potential buy or sell opportunities. The visualization in Figure 2 juxtaposes closing prices with these averages, aiding in trend stability identification.

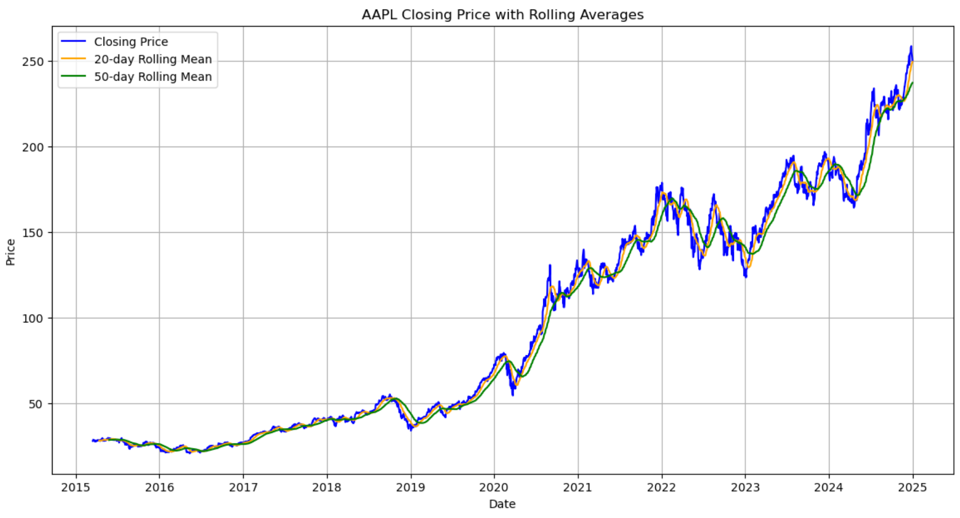

Figure 2: AAPL Closing Price with Rolling Averages

3.3 Trading Volume

Trading volume analysis, plotted via `plt.bar(stock_data.index, stock_data['Volume'])`, complements price trends by offering insights into market activity. For AAPL, volume ranged from 23.23 million to 648.83 million shares, with a mean of 117.09 million. Spikes often aligned with significant price movements, reflecting investor reactions. Figure 3 illustrates this data, enhancing understanding of market sentiment and volatility.

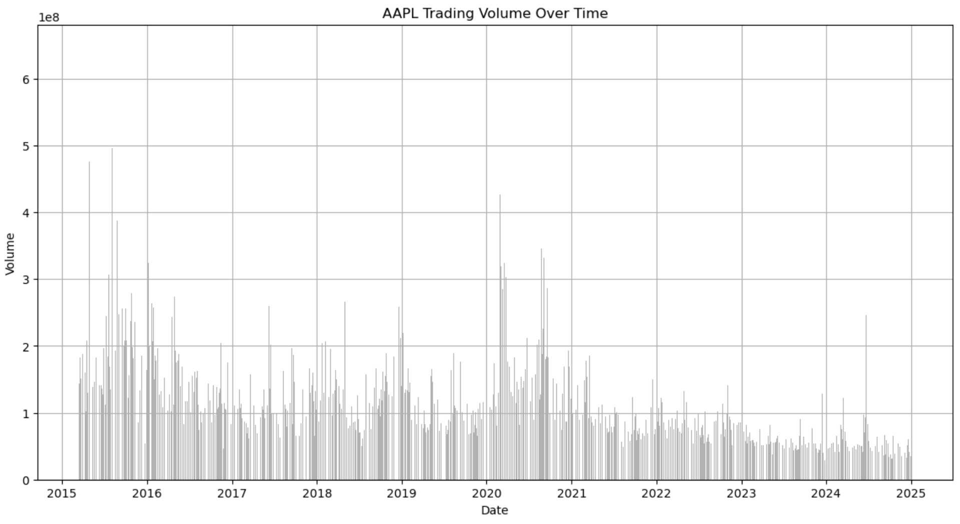

Figure 3: AAPL Trading Volume Over Time

3.4 Correlation Heatmap

A correlation heatmap, generated with `sns.heatmap(stock_data.corr())`, explored relationships between Open, High, Low, Close prices, and Volume. Strong positive correlations (near 1) among price variables were observed due to their daily interdependence, while Volume showed weaker correlations (e.g., 0.1–0.3 with Close), capturing distinct market behavior. Figure 4 aids in feature selection by highlighting multicollinearity.

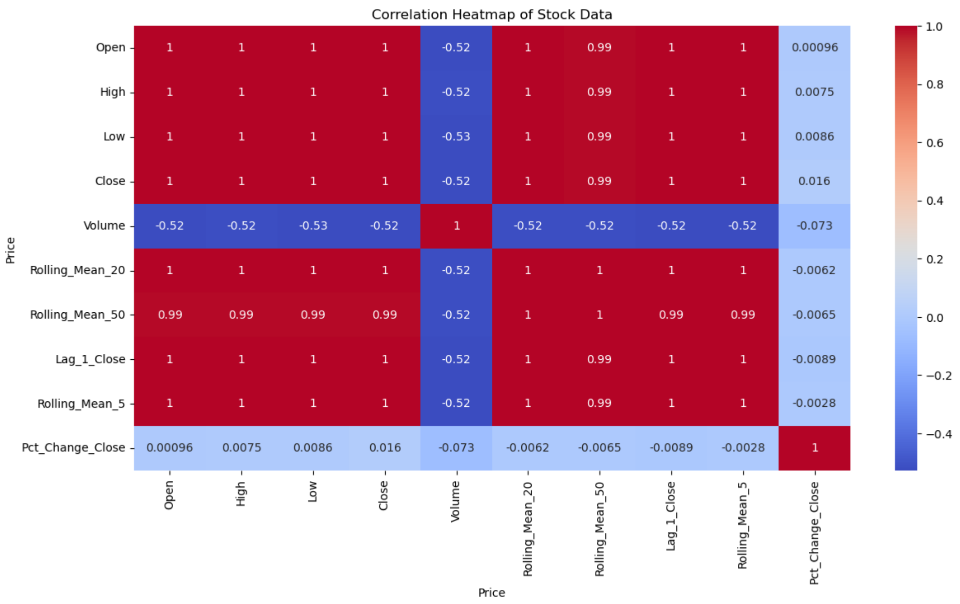

Figure 4: Correlation Heatmap of AAPL Stock Data

3.5 Pair Plots

Pair plots, created using `sns.pairplot(stock_data)`, visualized pairwise relationships. Scatter plots confirmed linear price variable relationships, while histograms showed distributions (e.g., right-skewed Close). Volume interactions appeared non-linear, informing feature engineering. Figure 5 provides a detailed view of these dynamics.

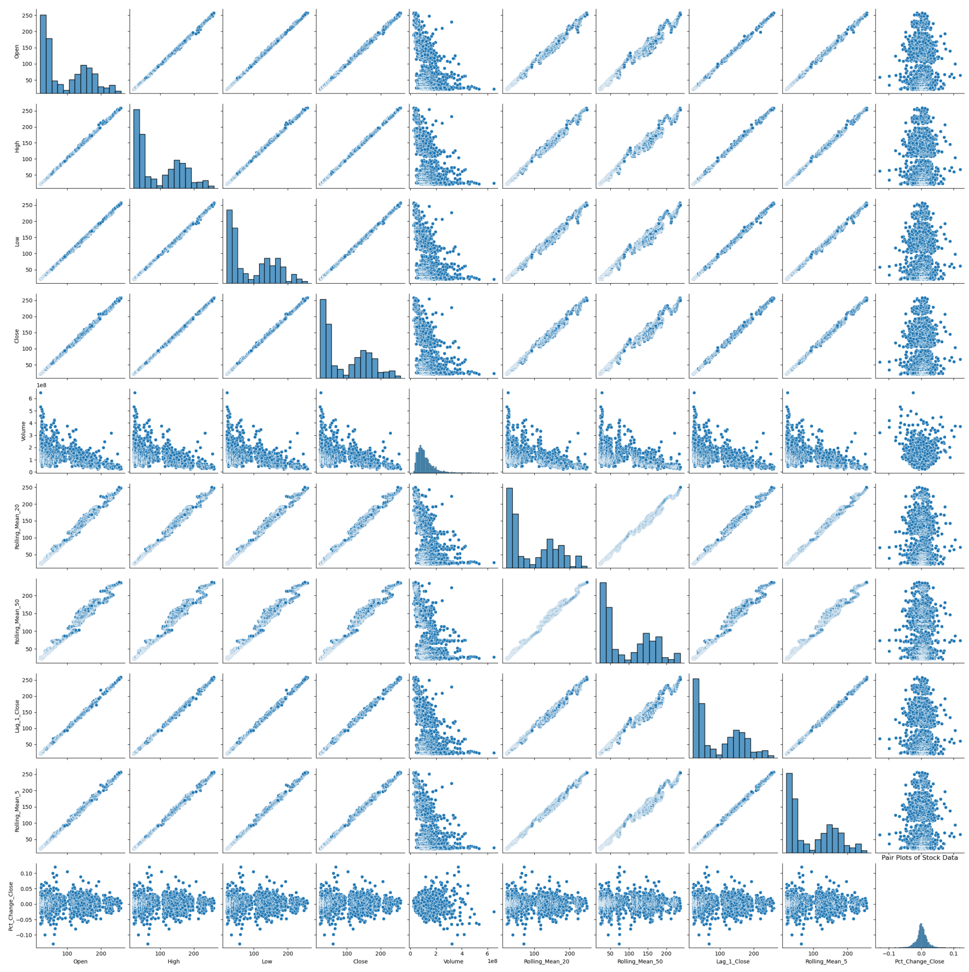

Figure 5: Pair Plots of AAPL Stock Data

3.6 Distribution of Closing Prices

A histogram with kernel density estimate (`sns.histplot(stock_data['Close'], kde=True)`) revealed a right-skewed distribution for AAPL's closing prices, with a median of \$64.61 and occasional high-value outliers (max: \$258.74). This skewness reflects typical stock market behavior. Figure 6 informs model assumptions, suggesting potential transformations.

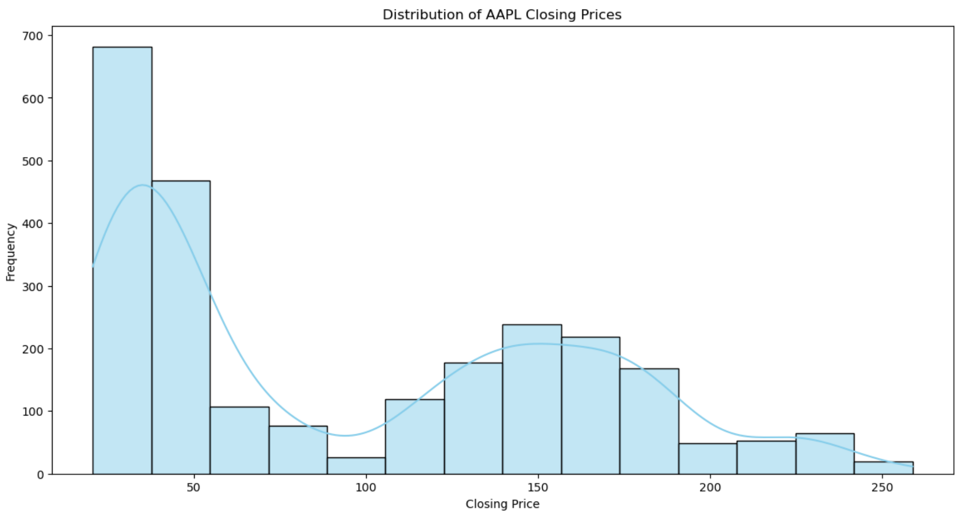

Figure 6: Distribution of AAPL Closing Prices

3.7 Box Plot for Volume

A box plot (`sns.boxplot(y=stock_data['Volume'])`) examined volume spread, showing a median of 100.36 million and outliers above 300 million, likely tied to major events. Figure 7 highlights these deviations, useful for assessing trading activity.

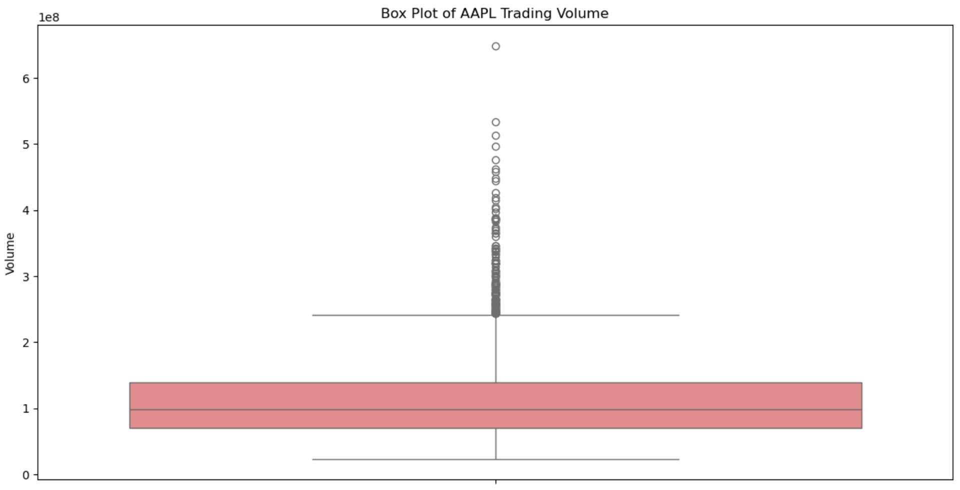

Figure 7: Box Plot of AAPL Trading Volume

3.8 Rolling Statistics (Mean and Standard Deviation)

Rolling 30-day mean and standard deviation (`stock_data['Close'].rolling(window=30).mean()`) assessed stationarity. The ADF test (statistic: 0.797, p-value: 0.992) indicated non-stationarity, prompting differencing (d=1). Figure 8 shows an increasing mean and variable standard deviation, critical for ARIMA and risk analysis.

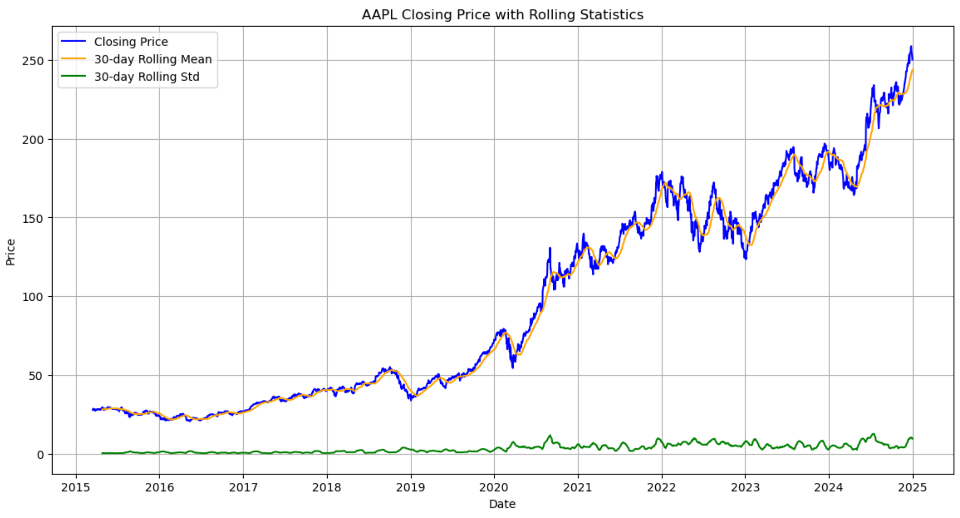

Figure 8: AAPL Closing Price with Rolling Statistics (Mean and Standard Deviation)

4 Feature Engineering

4.1 Feature Creation

New features were derived to enhance model performance:

- **Lagged Features:** `Lag_1_Close` = `stock_data['Close'].shift(1)` captures temporal dependencies.
- **Rolling Mean:** `Rolling_Mean_5` = `stock_data['Close'].rolling(window=5).mean()` encapsulates short-term trends.
- **Percentage Change:** `Pct_Change_Close` = `stock_data['Close'].pct_change()` measures daily returns and volatility.

These features leverage historical context and trends, improving predictive power.

4.2 Handling Missing Data

Feature engineering introduced NaNs (e.g., first 50 rows for Rolling_Mean_50), removed via `stock_data.dropna()`. This ensures only complete observations are modeled, preserving reliability.

5 Modeling

5.1 ARIMA Model

ARIMA modeled time-series properties, with stationarity checked via ADF (p-value: 0.992, non-stationary; differenced with d=1). Hyperparameter tuning used `itertools.product` over p=0–2, d=0–1, q=0–2, selecting (2, 1, 2) with RMSE 62.8092 on test data. A fixed model (p=2, d=1, q=2) yielded RMSE 4.5548 over 10 steps, with MAE 3.7653, MAPE 1.47%, R^2 -0.8008, and MASE 3.1219, indicating reasonable short-term fit but poor generalization.

5.2 Gradient Boosting Model (XGBoost)

XGBoost used features `['Lag_1_Close', 'Rolling_Mean_5', 'Pct_Change_Close']`, split 80/20 via `train_test_split`. RandomizedSearchCV tuned parameters (e.g., learning_rate: 0.1, max_depth: 30, subsample: 0.5), achieving RMSE 0.8938, MAE 0.4847, MAPE 0.57%, R^2 0.9998 on test data. A fixed model confirmed these results, excelling at capturing non-linear dynamics.

6 Model Evaluation

6.1 ARIMA Model

Test performance (over aligned test data) showed:

- **RMSE:** 168.1791, large errors.
- **MAE:** 155.3327, significant deviations.
- **MAPE:** 360.11%, poor relative accuracy.
- **R^2:** -5.7929, no explanatory power.

ARIMA struggled with non-linear market behavior.

6.2 Gradient Boosting Model

Test performance demonstrated:

- **RMSE:** 0.8938, minimal errors.
- **MAE:** 0.4847, tight alignment.
- **MAPE:** 0.57%, high accuracy.
- **R^2:** 0.9998, near-perfect fit.

XGBoost excelled across metrics.

6.3 Performance Comparison

Model	RMSE	MAE	MAPE	R^2
ARIMA	168.1791	155.3327	360.11%	-5.7929
Gradient Boosting	0.8938	0.4847	0.57%	0.9998

Table 1: Model Performance on Test Data

Table 1 highlights Gradient Boosting's superiority.

6.4 Visualization of Predictions

Figure 9 shows ARIMA forecasts diverging from actuals, while Figure 10 shows Gradient Boosting predictions closely tracking actuals, plotted over the last 50 test points.

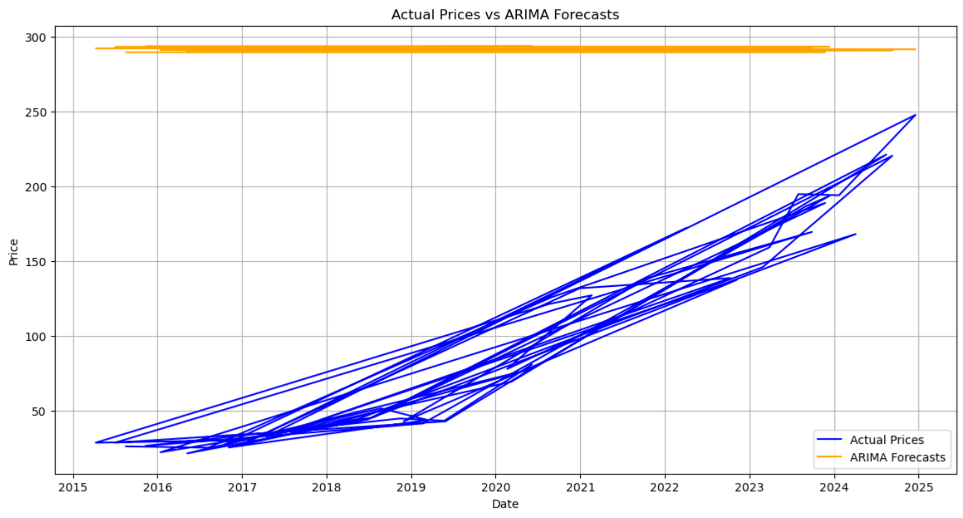

Figure 9: Actual Prices vs ARIMA Forecasts

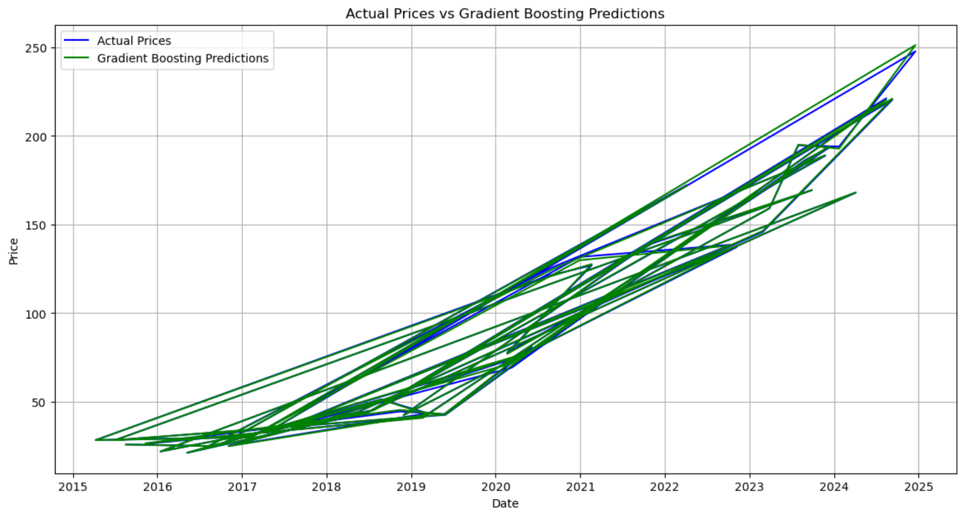

Figure 10: Actual Prices vs Gradient Boosting Predictions

7 Conclusions and Recommendations

7.1 Conclusions

- **ARIMA Model:** Suitable for short-term trends (MAPE 1.47% over 10 steps), but test errors (RMSE 168.1791, MAPE 360.11%) show limited applicability in volatile markets due to linear assumptions.
- **Gradient Boosting Model:** Excels with engineered features (RMSE 0.8938, MAPE 0.57%, R^2 0.9998), effectively handling non-linear relationships and fluctuations.

7.2 Recommendations

- **Prioritize Gradient Boosting:** Use for trading strategies due to its precision, supporting daily decisions.
- **Enhance Feature Engineering:** Add technical indicators (RSI, MACD) and sentiment from StockTwits to capture market psychology.

- **Model Optimization:** Apply regularization (L1, L2) to Gradient Boosting and explore Bayesian optimization for tuning, enhancing robustness.

These strategies maximize accuracy for trading needs as of March 20, 2025.