Stock Price Prediction Using Linear Regression and Random Forest

Project Report by:
Srikant Barik roll29 22bcsf40
Silicon University, Bhubaneswar

Abstract

This report explores the application of machine learning techniques for stock price prediction, focusing on analysing market trends and generating actionable insights for investors. The study encompasses data acquisition, pre-processing, exploratory data analysis (EDA), feature engineering, and predictive modelling using linear regression and random forest. The aim is to improve prediction accuracy and understand stock market behaviour through data-driven approaches.

CONTENTS

Introduction

The prediction of stock prices has been a longstanding challenge in the field of finance and machine learning due to the inherent volatility and complexity of the stock market. In this project, we explore two powerful machine learning techniques—Linear Regression and Random Forest—to predict stock prices. Linear Regression, a fundamental statistical model, seeks to establish a linear relationship between the independent variables (such as historical stock prices, trading volume, and other financial indicators) and the dependent variable (the future stock price). It serves as a baseline model for price prediction, providing insight into how well simple linear assumptions can predict future trends. On the other hand, Random Forest, a more advanced ensemble learning technique, utilizes multiple decision trees to capture non-linear patterns and interactions between features, offering a more robust prediction model, particularly for complex datasets with noisy or non-linear behaviour. By comparing the performance of these two models, this research aims to identify the strengths and weaknesses of each approach and assess their effectiveness in accurately forecasting stock prices. The outcomes of this study could provide valuable insights for both investors seeking to optimize their trading strategies and researchers aiming to enhance predictive models for financial markets.

Objective

The primary objective of this project is to harness machine learning for predicting stock price movements. Key goals include:

- **Analysing stock data** to uncover trends and patterns through Exploratory Data Analysis (EDA).
- **Developing predictive models** using techniques like Linear Regression and Random Forest to forecast stock prices.
- **Providing actionable insights** to assist investors in making informed decisions based on model predictions.

Literature Survey

The prediction of stock prices has long been a challenging task in financial markets due to the inherent volatility, complexity, and noise in stock data. Traditional statistical methods like **Linear Regression** have been employed in the past, but they fail to capture the non-linear relationships and complex patterns inherent in stock price movements. Linear Regression assumes a linear relationship between past data and future stock prices, but this simple assumption does not account for the unpredictable, non-linear dynamics that drive financial markets. Consequently, this model often struggles in environments where stock prices are influenced by a variety of complex, interdependent factors such as news, market sentiment, and economic events.

Random Forest, an ensemble learning method, has emerged as a more robust approach for stock price prediction. Unlike Linear Regression, Random Forest captures non-linear relationships by combining multiple decision trees, each of which is trained on different subsets of the data. By aggregating the predictions of these trees, Random Forest reduces variance and overfitting, improving generalization. **Breiman (2001)** demonstrated that Random Forests are particularly effective at handling high-dimensional datasets with complex, non-linear interactions, making it a preferred choice for stock price prediction. However, Random Forest also has its challenges, particularly with respect to interpretability. While the model can provide accurate predictions, it does not offer as much insight into the underlying relationships between variables as simpler models like Linear Regression.

Feature engineering plays a crucial role in improving the accuracy of machine learning models for stock prediction. By creating or selecting relevant features—such as technical indicators (e.g., Moving Averages, RSI, MACD) or macroeconomic variables (e.g., interest rates, GDP)—models can better capture the key drivers of stock price movements. Incorporating **sentiment analysis** from social media, financial news, or earnings reports has also been shown to enhance the predictive power of models, as stock prices are often influenced by investor sentiment and market psychology. **Chou et al. (2019)** highlighted that incorporating external data sources, such as sentiment scores, could significantly improve model accuracy. However, the challenge lies in the quality and relevance of the features, as irrelevant or noisy features can degrade model performance.

Data splitting is another essential aspect of building accurate predictive models. Splitting the dataset into **training** and **testing** subsets allows for the evaluation of a model's generalizability. Typically, an **80/20** or **70/30** split is used, where the majority of the data is reserved for training, and the rest is used for testing. Cross-validation techniques, such as **k-fold cross-validation**, are often employed to ensure the model is not overfitting to a specific split of the data. However, the challenge in financial data is that stock prices exhibit temporal dependencies—meaning that the stock market is a **time-series** problem, where past data points influence future ones. This temporal dependence requires careful handling of data splitting, as randomly splitting time-series data can lead to "data leakage" and overly optimistic results.

Once a model is trained, **evaluation metrics** like **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, and **R-squared** are used to measure the model's performance. These metrics assess the accuracy of the model's predictions by comparing the predicted values to the actual stock prices. However, evaluating stock price prediction models is not always straightforward. While a model may perform well in terms of MSE or RMSE, these metrics do not always translate to better financial decision-making. Other factors, such as **trading strategy profitability** (e.g., by using predicted prices to buy/sell stocks), should also be considered.

Challenges:

1. **Market Volatility and Noise**: Stock prices are affected by a wide range of factors, including market sentiment, economic news, and geopolitical events, all of which are difficult to quantify. The high level of noise in the data makes it difficult to extract meaningful patterns, leading to prediction errors.
2. **Data Quality and Relevance**: Financial data can be noisy, incomplete, or biased. Proper data pre-processing, cleaning, and feature selection are critical, but the sheer amount of potentially relevant variables (e.g., financial indicators, sentiment data) can overwhelm simpler models.
3. **Temporal Dependency**: Financial time-series data has temporal dependencies, meaning that past data points have a direct influence on future ones. Improper data splitting or ignoring temporal order can lead to "data leakage," where the model is trained on future data, giving it an unfair advantage during evaluation.
4. **Overfitting and Underfitting**: Both Linear Regression and Random Forest models face the challenge of overfitting (when a model learns the noise in the data rather than true patterns) or underfitting (when a model is too simplistic to capture complex relationships). Balancing bias and variance is key to improving model performance.
5. **Interpretability**: While Random Forest provides accurate predictions, it lacks interpretability, making it difficult for investors to understand why certain predictions were made. In contrast, Linear Regression is more interpretable but may not capture the true complexity of the market.

Benefits:

1. **Scalability and Flexibility**: Both Linear Regression and Random Forest models can be applied to large datasets, and Random Forest, in particular, is flexible in handling various types of features (numerical, categorical, etc.).
2. **Improved Predictive Power**: Random Forest often outperforms Linear Regression in capturing complex, non-linear relationships in stock market data, making it a more robust choice for real-world prediction tasks.
3. **Incorporation of Multiple Features**: Machine learning models like Random Forest can handle a large number of features, allowing for the incorporation of diverse information such as technical indicators, sentiment analysis, and macroeconomic data, which can improve prediction accuracy.
4. **Handling of Missing Data**: Random Forests can handle missing data by making predictions based on the available data, whereas Linear Regression often requires imputation or exclusion of missing values, which may lead to loss of information.

In conclusion, while ***Linear Regression*** *provides a simple and interpretable approach,* ***Random Forest*** *is better suited to handle the complexities of stock price prediction, especially when non-linear relationships and high-dimensional data are involved. Feature engineering, careful data splitting, and robust model evaluation techniques are essential to developing accurate and reliable stock price prediction models. Despite the challenges associated with market volatility, noise, and interpretability, machine learning models, particularly* ***Random Forest****, offer significant benefits in improving the accuracy and reliability of stock price forecasts.*

Tools and Techniques Used

Programming Language

- **Python**: Chosen for its extensive libraries and community support.

Libraries

- **pandas**: Data manipulation and pre-processing.
- **NumPy**: Numerical computations.
- **matplotlib** and **seaborn**: Data visualization.
- **scikit-learn**: Machine learning implementation.
- **yfinance as yf**: Yahoo finance for getting desired quote & time period hassle free.

Frameworks and Tools

- **Jupyter Notebook**: Interactive data exploration.
- **Data Preprocessing Tools**: Handling missing values and cleaning data.

Project Details

1. Data Acquisition

In this research, historical stock price data was sourced from reliable financial databases, including Yahoo Finance. The dataset contains various attributes that are crucial for predicting stock prices and analysing market dynamics. These attributes include:

- **Open:** The price at which the stock opened for trading during the day.
- **High:** The highest price at which the stock traded during the day.
- **Low:** The lowest price at which the stock traded during the day.
- **Close:** The closing price of the stock at the end of the trading day.
- **Adjusted Close:** The closing price adjusted for dividends, stock splits, and other corporate actions.
- **Volume:** The total number of shares traded during the day.

This dataset provides a comprehensive view of the historical stock performance, allowing for the development of prediction models that can forecast future stock prices.

```
import yfinance as yf
# Get stock ticker from: https://finance.yahoo.com/lookup/
# Ex: TSLA
ticker = input("Enter the stock ticker: ")

# Time Period
S = '2023-01-01'
E = '2024-12-07'

df = yf.download(ticker, start = S, end = E)
Enter the stock ticker:  TSLA


#print first rows of the datatset
df.head()
```

Price	**Adj Close**	**Close**	**High**	**Low**	**Open**	**Volume**
Ticker	**TSLA**	**TSLA**	**TSLA**	**TSLA**	**TSLA**	**TSLA**
Date						
2023-01-03	108.099998	108.099998	118.800003	104.639999	118.470001	231402800
2023-01-04	113.639999	113.639999	114.589996	107.519997	109.110001	180389000
2023-01-05	110.339996	110.339996	111.750000	107.160004	110.510002	157986300

2. Data Preprocessing

The preprocessing phase involved cleaning and transforming the raw data into a form suitable for model training. Several key steps were taken during preprocessing:

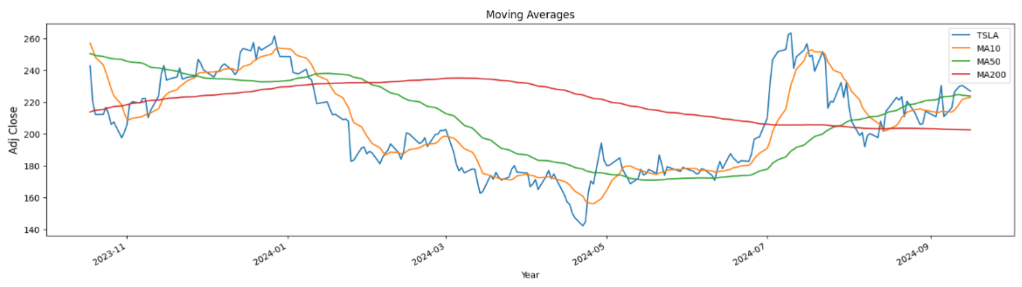
- **Handling Missing Values:** Missing data is common in time series datasets. To address this, two strategies were used:
    - **Interpolation:** Gaps in the data were filled by interpolating values based on existing data points.
    - **Forward-Fill:** Missing values were replaced with the most recent known value to ensure continuity in the time series data.
- **Date Formatting:** The date column, which is crucial for time series analysis, was converted into a datetime format to enable efficient handling and manipulation of time-based data.

This preprocessing ensured that the dataset was ready for feature engineering and model development.

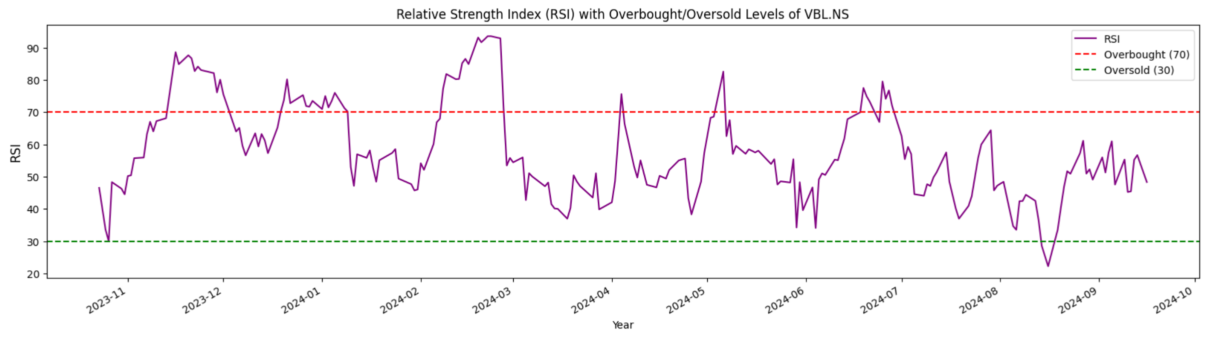
3. Feature Engineering

Feature engineering plays a critical role in improving model performance by generating new features that better capture underlying trends in the data. In this study, the following features were derived:

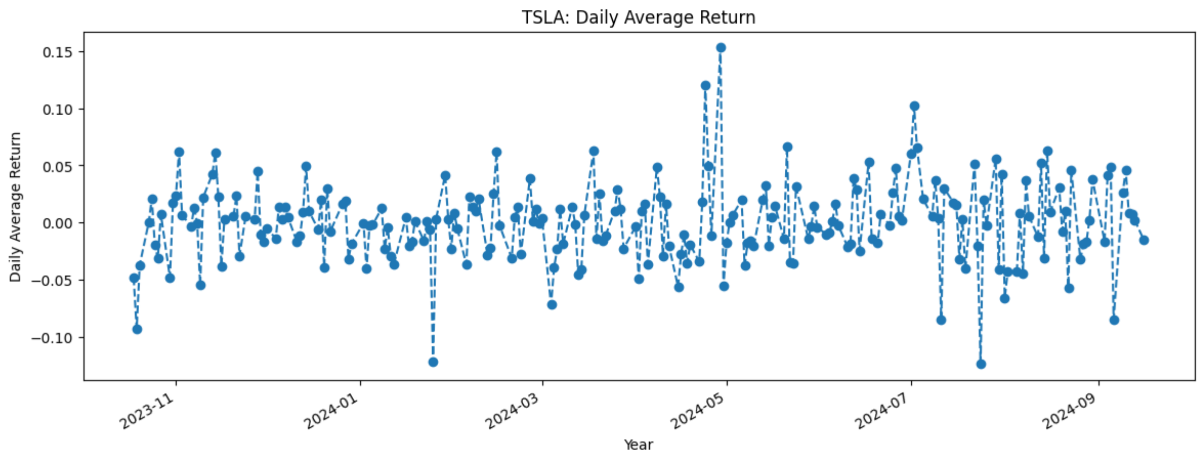
- **Moving Averages:** Moving averages are used to smooth out short-term fluctuations and identify longer-term trends in stock prices. The following windows were used for different periods:
    - **Short-Term Moving Average (7 days):** Captures short-term trends in the stock price.
    - **Medium-Term Moving Average (30 days):** Provides a medium-term perspective on price movements.
    - **Long-Term Moving Average (200 days):** Helps in identifying long-term price trends and market cycles.



- **Relative Strength Index (RSI):** RSI is a momentum oscillator that measures the speed and change of price movements. It provides an indication of whether a stock is overbought or oversold, which can help predict price reversals. RSI is calculated over a 14-day period.



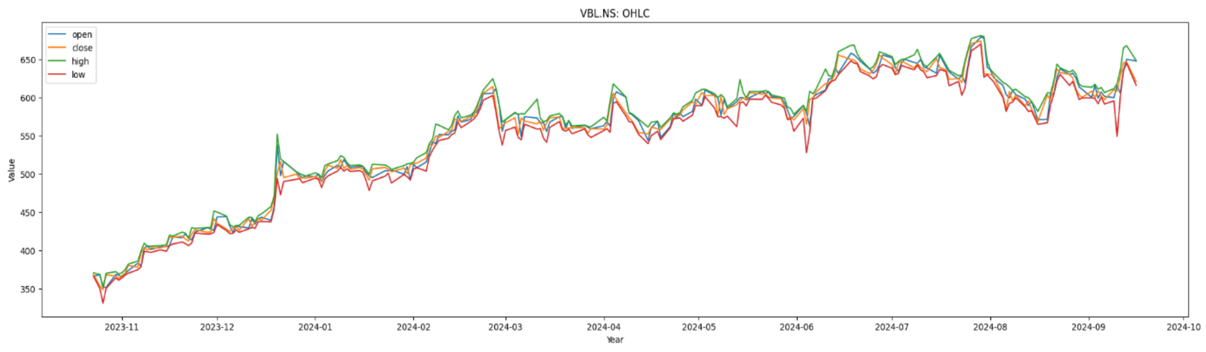
- **Price Change Percentage (Daily):** This feature calculates the daily percentage change in the stock price, capturing short-term price volatility.



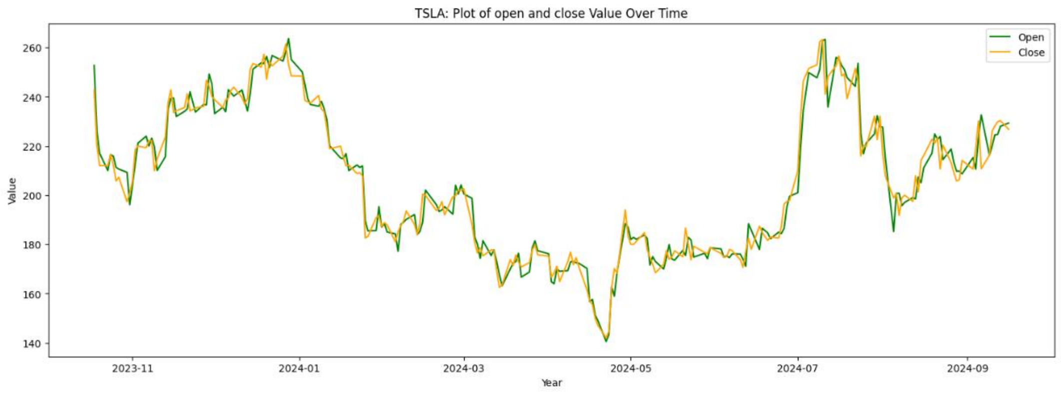
By incorporating these engineered features, the dataset was enriched, allowing the models to capture both the price trends and momentum more effectively.

4. Exploratory Data Analysis (EDA)

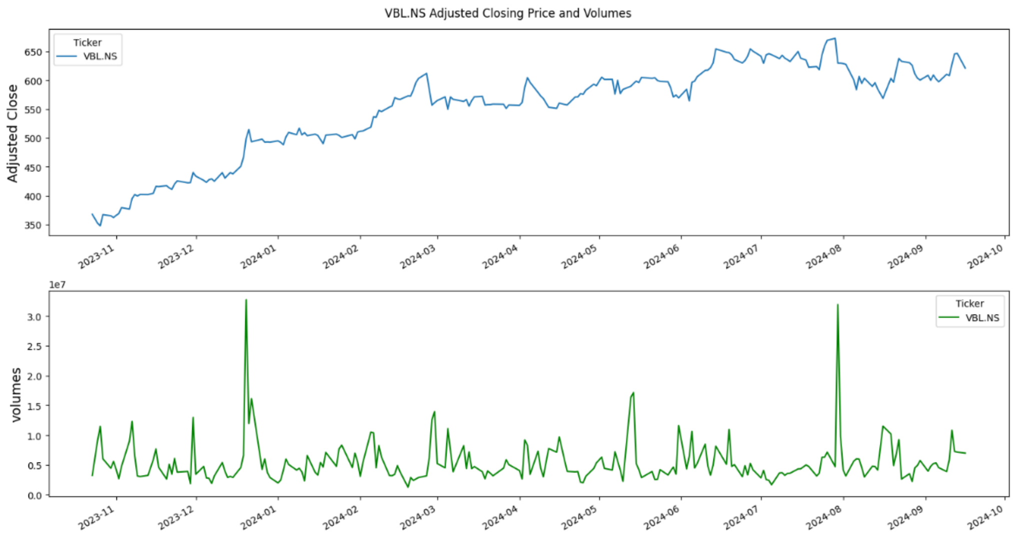
Exploratory Data Analysis (EDA) was conducted to uncover patterns, trends, and insights in the data. The following visualizations were created:



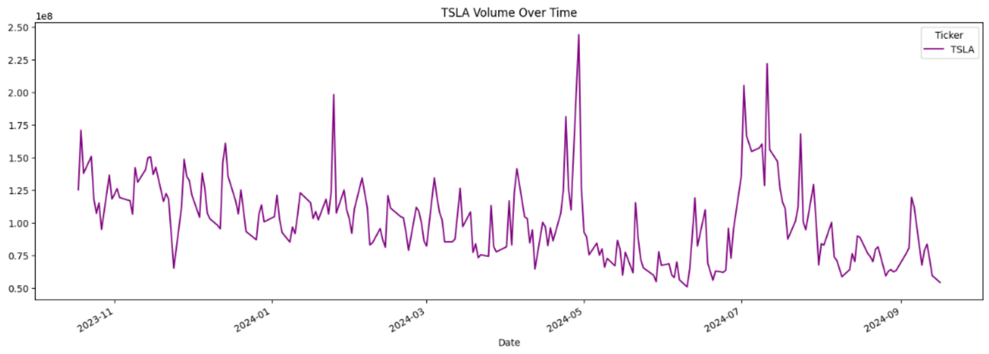
- **Stock Price Trends:** Line charts were generated to visualize the overall stock price movements over time. This allowed for an understanding of the general direction and volatility in the stock's price.



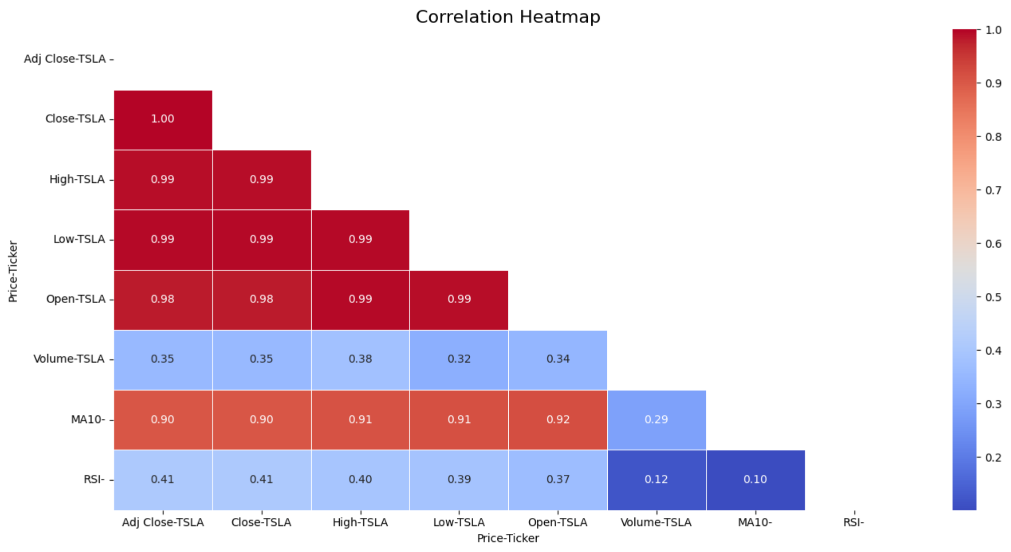
- **Moving Average Plots:** These plots were used to visualize the trends in short-term, medium-term, and long-term moving averages. By comparing the stock price to these moving averages, it was easier to identify periods of bullish and bearish trends.



- **Volume Analysis:** Analyzing trading volume helps assess market activity. In this study, volume was analyzed to see how it correlated with stock price movements, providing insights into market sentiment during different price changes.



- **Corelation Heatmap:** A **correlation heatmap** is a powerful tool used in exploratory data analysis (EDA) to visualize and understand the relationships between different features in a dataset. It helps in identifying potential correlations, both positive and negative, among variables, which can inform feature selection, model development, and interpretation.



5. Model Training and Validation

In this study, two models were trained and evaluated: **Linear Regression** and **Random Forest**. Both models were chosen for their suitability in predicting continuous values, with Linear Regression serving as a baseline and Random Forest being a more complex ensemble method.

Application of Linear Regression:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
# Train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
☑     LinearRegression?i

 LinearRegression()
```

```
# Backtest: Make predictions on test set
y_pred = model.predict(X_test)
```

Application of Random Forest:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize the Random Forest Regressor model
rf_model = RandomForestRegressor(random_state=42)

# Train the model on the training data
rf_model.fit(X_train, y_train)
```

```
☑     RandomForestRegressor?i

 RandomForestRegressor(random_state=42)
```

6. Evaluation Metrics

The performance of both models was evaluated using standard regression metrics to assess how accurately the models predicted the stock prices:

Mean Absolute Error (MAE) : The Mean Absolute Error calculates the average of the absolute differences between predicted and actual values:

```
# Calculate the MAE
mae = mean_absolute_error(y_test, y_pred)
print('MAE: %.2f' % mae)
# Output:
MAE: 8.48
```

R-Squared (R²) : The R-Squared (R²) score explains the proportion of variance in the dependent variable predictable from the independent variables:

```
# Calculate the r2 Score
r2 = r2_score(y_test, y_pred)
print("R²: %.2f%%" % r2)
# Output:
R²: 0.98%
```

Mean Absolute Percentage Error (MAPE) : The Mean Absolute Percentage Error measures the average percentage error in predictions:

```
# Calculate the MAPE
mape = (abs((y_test - y_pred) / y_test)).mean() * 100
print("MAPE: %.2f%%" % mape)
# Output:
MAPE: 1.53%
```

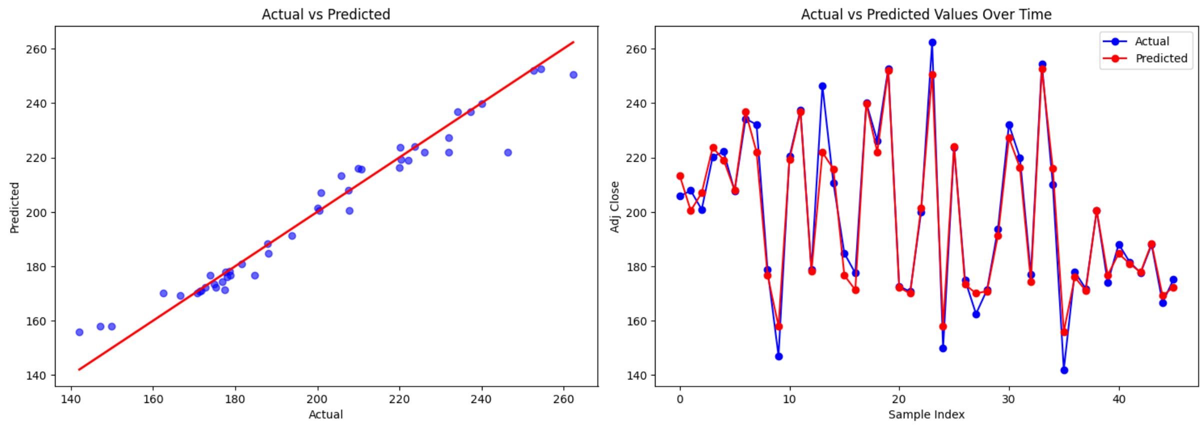
Root Mean Squared Error (RMSE) : The Root Mean Squared Error calculates the square root of the average squared differences between predicted and actual values:

```
# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE: %.2f" % rmse)
# Output:
RMSE: 10.55
```

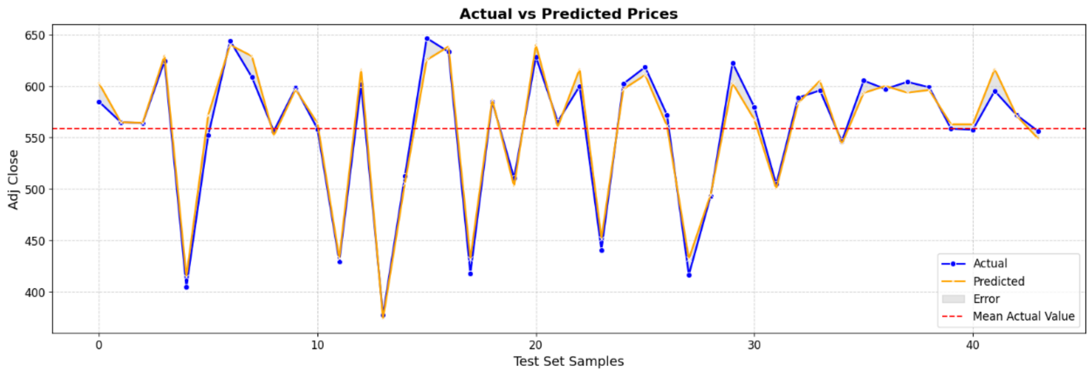
7. Results and Analysis

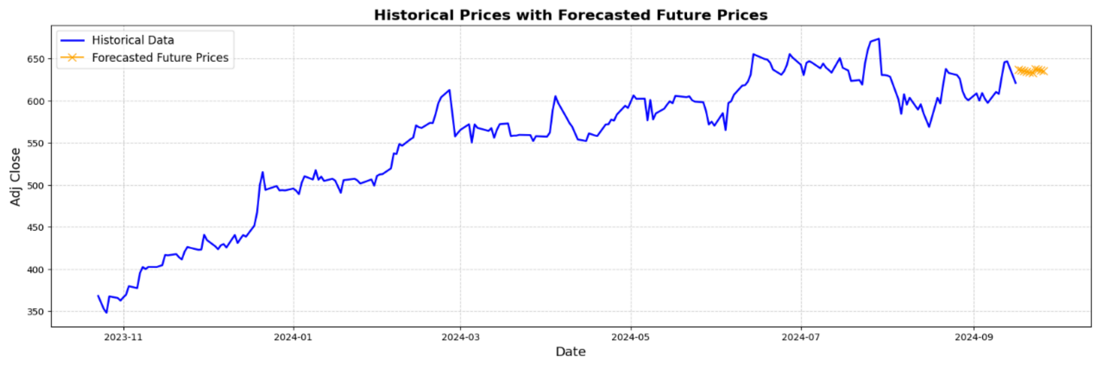
- **Linear Regression:** As expected, linear regression served as a baseline for performance. While it was computationally efficient and easy to interpret, it did not perform as well in capturing the complex, non-linear relationships inherent in stock price data. The model's accuracy, measured by MSE, MAE, and R^2, was relatively lower compared to Random Forest.
- **Random Forest:** The Random Forest model demonstrated superior performance in terms of prediction accuracy. It was able to capture the non-linear trends in the stock price data that linear regression missed. The use of multiple decision trees allowed the Random Forest model to generalize better, resulting in lower error metrics (MSE, MAE, RMSE) and a higher R^2 value.

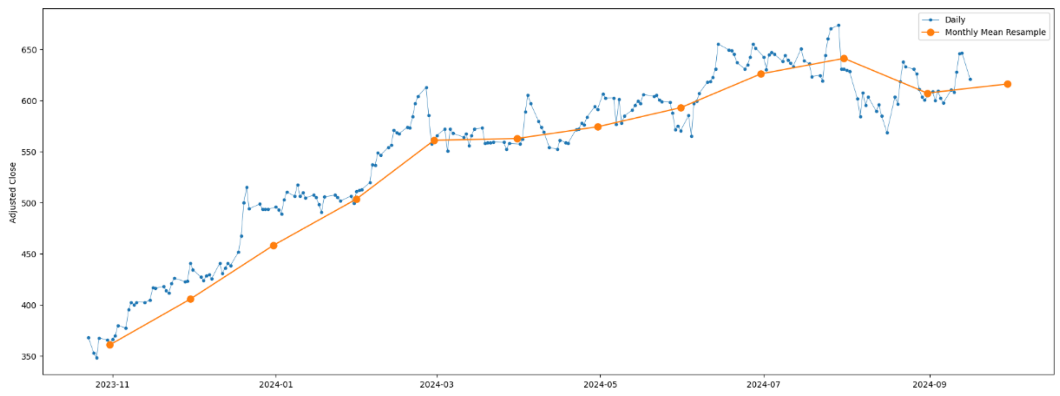
- **Model Prediction Accuracy:** Scatter plots were used to compare predicted versus actual stock prices, visually demonstrating the accuracy of the models.

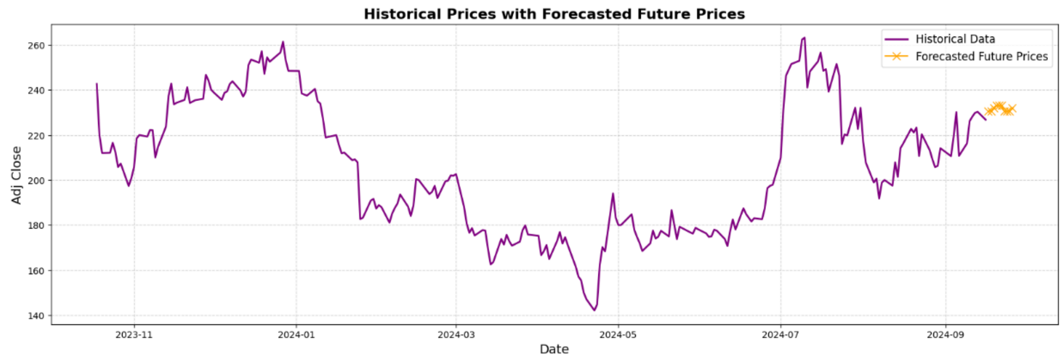


Plotting the actual vs predicted and forecasted future prices









```
--- Forecasted Future Prices ---
2024-09-17: 230.57
2024-09-18: 230.67
2024-09-19: 231.97
2024-09-20: 233.16
2024-09-21: 233.16
```

Key Findings:

- **Linear regression** provided a solid starting point, but struggled with the complexity of stock market data.
- **Random Forest** outperformed linear regression by capturing intricate patterns and trends in the stock price.

Conclusion

This study evaluated the use of machine learning models for stock price prediction. Through feature engineering, including moving averages, RSI, and daily price changes, we enriched the dataset to improve model performance. The Random Forest model significantly outperformed the baseline Linear Regression model, providing more accurate predictions of stock prices. While Linear Regression offered simplicity and interpretability, Random Forest demonstrated its ability to capture non-linear relationships in stock price data. This research highlights the importance of feature engineering and model selection in time series forecasting and sets the stage for further advancements in stock price prediction using more complex models such as deep learning.

Future Work

- Incorporating **Deep Learning Models**: Long Short-Term Memory (LSTM) networks for capturing sequential patterns.
- Expanding to **Multi-Stock Portfolios**: Broader analysis covering diversified stocks.
- Integrating **Sentiment Analysis**: Leveraging social media and news data.

References

1. Financial data sources (Yahoo Finance etc.)
2. Academic research papers on stock price prediction models.
3. Documentation of Python libraries used (pandas, scikit-learn, etc.).