# NASDAQ Price Prediction

University of Connecticut – STAT 4255

Julia Mazzola

2024-12-11

# a. Description of the Data

As a double major in Statistical Data Science and Economics, I aim to integrate my dual interests in this project. By using the statistical learning methodologies and tools acquired in this course, I seek to understand key economic questions dervived throughout my economics degree. For an overall approach, I chose to use data from the NASDAQ Composite to analyze and interpret these economic issues. The NASDAQ Composite is a significant indicator of the economy, reflecting the performance of over 3,000 companies, primarily in the technology sector. Its trends can provide insights into broader economic conditions, investor sentiment, and potential future economic activity.

Further, I wanted to conduct my own research on the stock market to gain a deeper understanding of the various factors that influence its returns. As someone who is passionate about financial analysis and data-driven decision-making, this will provide me with valuable insights and help me build a solid foundation for future work in this area.

The NASDAQ Composite Index is one of the most followed in the United Statess. Two similar indices are the S&P 500 Index and the Dow Jones Industrial Average (DJIA); however, for the sake of this project, I am focusing solely on NASDAQ. Using key indicators that influence stock performance, I will run a prediction model to forecast stock price movements based on the historical data and economic variables within the dataset.

The dataset comprises of 13 variables from NASDAQ data spanning from January 4, 2010, to October 25, 2024, with a total of 3,914 daily observations. The names of the variables are Date, Open, High, Low, Close, Volume, InterestRate, ExchangeRate, VIX, TED Spread, EFFR (Effective Federal Funds Rate), Gold, and Oil. For my prediction my response variable will be Close. The dataset was obtained from Kaggle.

## Variables in Original Dataset

| Variable | Description |
|---|---|
| Date | The date of the recorded stock prices (formatted as YYYY-MM-DD). |
| Open | The price at which the stock opened for trading on a given day. |
| High | The highest price reached by the stock during the trading day. |
| Low | The lowest price recorded during the trading day. |
| Close | The price at which the stock closed at the end of the trading day. |
| Volume | The total number of shares traded during the day. |
| Interest Rate | The prevailing interest rate, which influences economic activity and stock performance. |
| Exchange Rate | The exchange rate for the USD against other currencies, reflecting international market influences. |
| VIX | The Volatility Index, a measure of market risk and investor sentiment, often referred to as the "fear index." |
| Gold | The price of gold per ounce, which serves as a traditional safe-haven asset and is often inversely correlated with stock prices. |
| Oil | The price of crude oil, an essential commodity that influences various sectors, especially transportation and manufacturing. |
| TED Spread | The difference between the interest rates on interbank loans and short-term U.S. government debt, indicating credit risk. |
| EFFR | The interest rate at which depository institutions lend reserve balances to other depository institutions overnight, influencing overall economic activity. |

## Main Questions

### What are the main predictors of stock price?

Stock prices can be influenced by a variety of factors, but which variables are the most significant? Identifying key predictors is crucial for building accurate and reliable models.

### How does seasonality affect stock price?

Stock prices often exhibit seasonal patterns, with certain months historically performing better or worse due to factors like economic cycles, consumer demand, and investor behavior.

### Is using a statistical learning model the best approach to predict future prices?

Statistical learning models can be effective for predicting future outcomes. However, are they truly beneficial when applied to time-series data or is there a more efficent approach?

Using these questions, I aim to identify the main predictors of stock prices, understand how seasonality affects stock prices, and evaluate the effectiveness of statistical learning models in predicting future prices. By integrating insights from economics and advanced statistical learning models, I hope to build more accurate and reliable predictive models, ultimately enhancing my ability to make informed investment decisions.

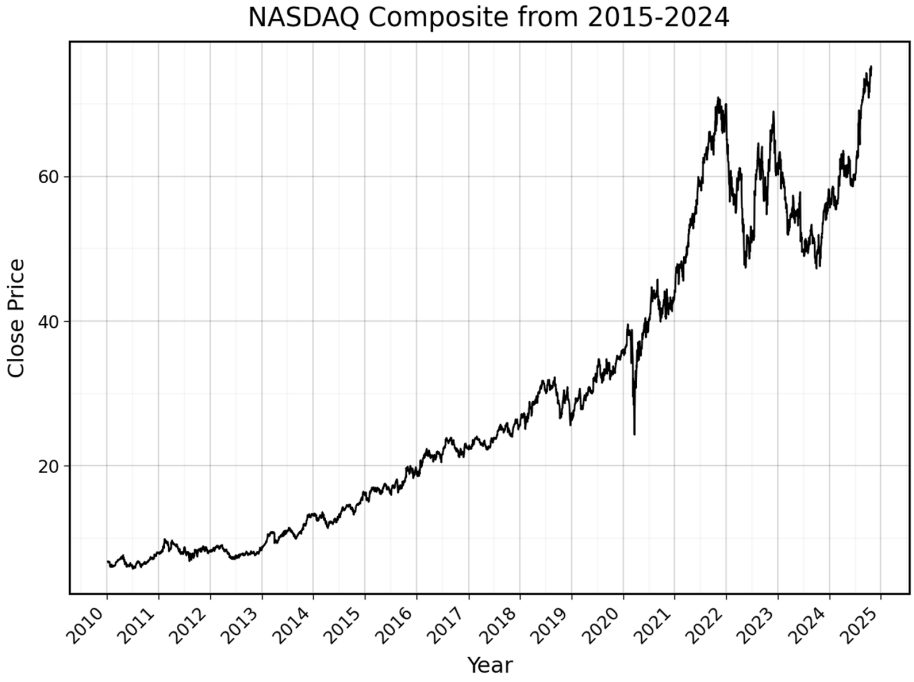


Figure 1: NASDAQ Composite from 2010-2024

The figure above illustrates the stock price trends over time. From 2010 to 2020, there was a steady increase, reflecting the market's growth during this period. However, a sharp decline occurred in 2020, primarily driven by the economic disruptions caused by the COVID-19 pandemic. Following this downturn, the market rebounded and has shown a fluctuating yet upward trajectory, marked by significant peaks and valleys. As of the most recent data, the market is on an upward trend heading into 2025, signaling continued growth despite prior volatility.

## b. Other Approaches

**LASSO Regression:** I chose against a linear regression model as stock price fluctuates greatly, so having a model that can capture these nonlinearities was more benefical for my overall goals.

**Random Forest:** Another potential approach, which is a tree based bagging learning method that builds multiple decision trees to improve prediction accuracy and reduce overfitting. I went against random forest as usually gradient boosting works more effectivly overall.

**AutoRegressive Integrated Moving Average (ARIMA):** While ARIMA can provide accurate short-term forecasts and is interpretable, it assumes that the relationships within the data are linear and stationary. This can limit its performance when dealing with more complex, non-linear stock price movements or when incorporating external factors, like interest rates or exchange rates.

In the future, I hope to utilize approaches such as ARIMA as it can be useful for time-series data. ARIMA's ability to model the autoregressive and moving average aspects of stock price behavior could provide valuable insights, particularly in cases where external variables are less influential. The potential to combine a hybrid model with ARIMA could be beneficial and provide a deeper understanding of how to integrate classical statistical methods with modern machine learning techniques. This approach seems interesting; however, given my current skill set and the time required to learn and implement it, I recognize that I am not yet at that skill level.

## c. Final Approach

**Extreme Gradient Boosting (XGBoost):** For this project, I will use Extreme Gradient Boosting (XGBoost) to predict stock price. I chose this approach due to its simplicity and ability to handle large datasets. XGBoost is an efficient, scalable machine learning algorithm that excels in handling both structured and unstructured data. It is known for its high performance in regression tasks, especially when dealing with time-series data like stock prices. Additionally, XGBoost can effectively model non-linear relationships and capture complex patterns, which are crucial when analyzing stock price fluctuations. By leveraging its regularization capabilities, XGBoost can also help prevent overfitting, ensuring robust predictions even with noisy data.

In order to capture seasonality, I created month variables to account for monthly fluctuations in stock performance. Specific months tend to exhibit stronger or weaker returns based on historical patterns. For instance, February, June, and September have historically been among the weakest months for stock performance, while April, July, and November generally show better returns. By incorporating these month variables into the model, I can better account for these periodic trends and improve the accuracy of predictions.

Further, I also calcuated the 10-day moving average for each day as it could serve as a possible predictor for the model of where the market is trending. The 10-day moving average helps smooth out daily fluctuations and reveals the general direction in which the market is moving.

Finally, for the model to run efficiently, the date variable was transformed into days_since_start to convert the time component into a numerical format. This transformation allowed the model to interpret time as a continuous variable, ensuring better handling of the time-series data.

## d. Summary of Results

The XGBoost model achieved a Mean Squared Error (MSE) of 6.2701, indicating the overall prediction accuracy of the model. The model demonstrated better performance for data points closer to the train-test cutoff, where the predictions were more aligned with the actual values. However, its accuracy declined for data points further from the cutoff, where the model showed greater variability in its predictions. This suggests that the model's predictive power diminishes as it is applied to data points further into the test period, potentially highlighting challenges related to forecasting long-term trends.

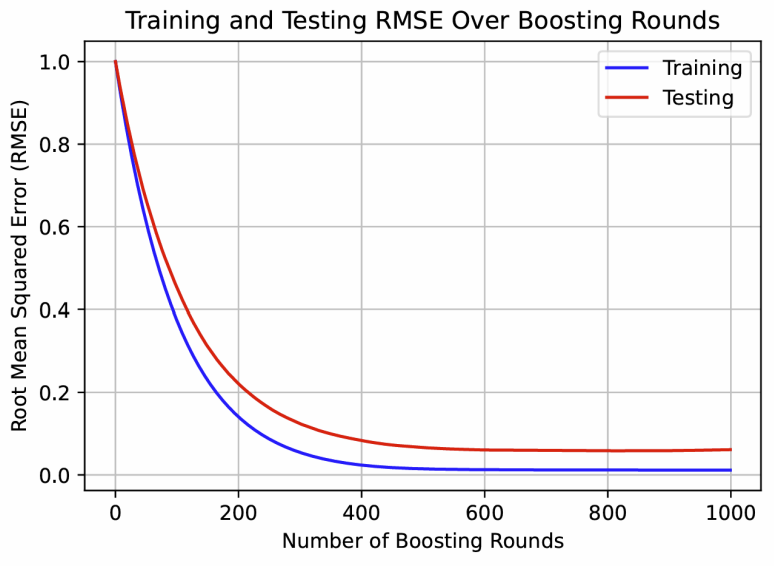


Figure 2: Training and Testing RMSE Over Boosting Rounds

The Root Mean Squared Error (RMSE) firgure depicts the model's predictions throughout the boosting rounds of training. The blue line represents the training RMSE, showing how the error decreases as the model learns from the training data over successive rounds. The red line represents the test RMSE, indicating the model's performance on the unseen test data. Both lines decrease and stabilize, showing similar testing trends to the training RMSE.

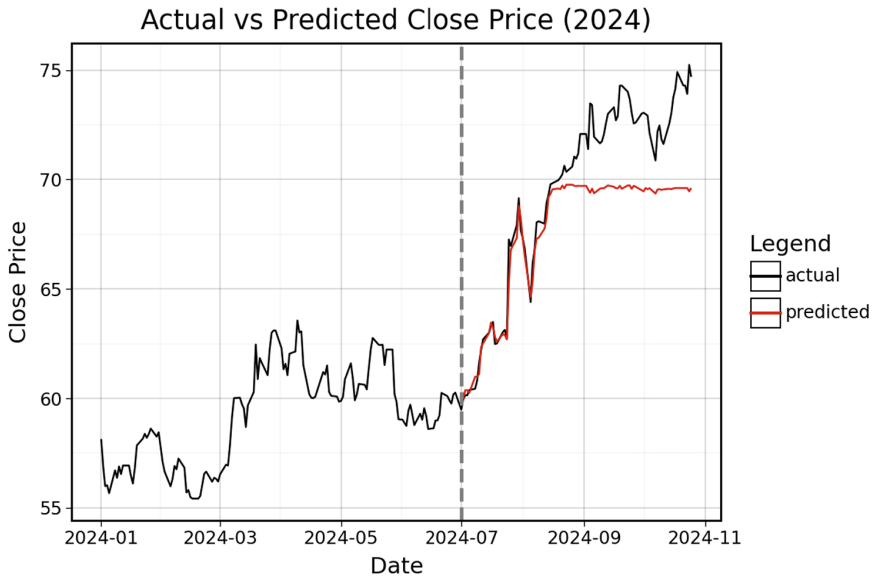


Figure 3: Actual vs Predicted Close Price (2024)

The features that were most influential in the model's predictions included low, high, vix, 10_day_ma, and days_since_start. These variables contributed significantly to the model's ability to forecast the stock price. On the other hand, some of the least impactful predictors were month, effr, interest rate, and gold. These

features had minimal effect on the model's performance, indicating that they may not provide substantial predictive value for stock price movements in this specific context.

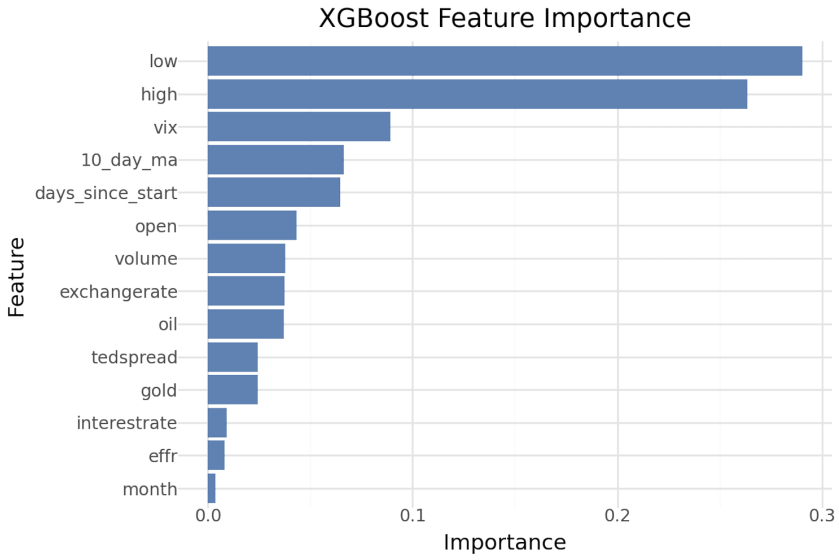


Figure 4: XGBoost Feature Importance

**Answering Main Questions:**

**What are the main predictors of stock price?**

The main predictors of stock price low, the lowest price recorded during the trading day, high, the highest price reached by the stock during the trading day, and the Volatility Index. Other contributers were the 10-day moving average and date.

**How does seasonality affect stock price?**

Incorporating the month variable was intended to capture seasonality effects; however, it proved to be one of the weakest predictors. Utilizing an alternative seasonality measure, such as the quarter, may yield more meaningful insights.

**Is using a statistical learning model the best approach to predict future prices?**

While a statistical learning model may not be ideal for long-term predictions, it performed well in the short term. Incorporating a hybrid model better suited for time-series analysis could enhance predictive accuracy and adaptability.

# e. Conclusions

In conclusion, the model demonstrated moderate success in predicting the NASDAQ Composite, showing reasonable accuracy in general. However, it faced challenges when handling data further from the cutoff date, indicating that its predictive accuracy diminishes over time. For future work, exploring models specifically designed for time-series forecasting, such as ARIMA or LSTM networks, may improve the model's ability to capture long-term trends and seasonality, potentially yielding more accurate predictions for stock prices beyond the immediate test period. LLearning to implement these models will be something I focus on in the future to grow my combined Economic and Statistical field knowledge.

# Appendix

Below is the code and the figures used for the model and deliverables. The code includes data preparation, model training, performance evaluation, and visualization of the actual vs predicted stock prices. The figures demonstrate the model's performance and the trends observed in the NASDAQ Composite from the training period to the test period.

**1. Plottiong Close Prices**

```
from plotnine import *
import pandas as pd

# Read the data
nasdaq = pd.read_csv('data/nasdaq.csv')

# Convert 'Date' column to datetime
nasdaq['Date'] = pd.to_datetime(nasdaq['Date'])

# Plot the 'Close' values
(ggplot(nasdaq, aes(x='Date', y='Close'))
    + geom_line(color='black')
    + scale_x_datetime(date_labels='%Y', breaks='1 year',)
    + theme_linedraw()
    + theme(axis_text_x=element_text(rotation=45, hjust=1),
            figure_size = (5,4))
    + labs(title='NASDAQ Composite from 2010-2024', x='Year', y='Close Price'))
```

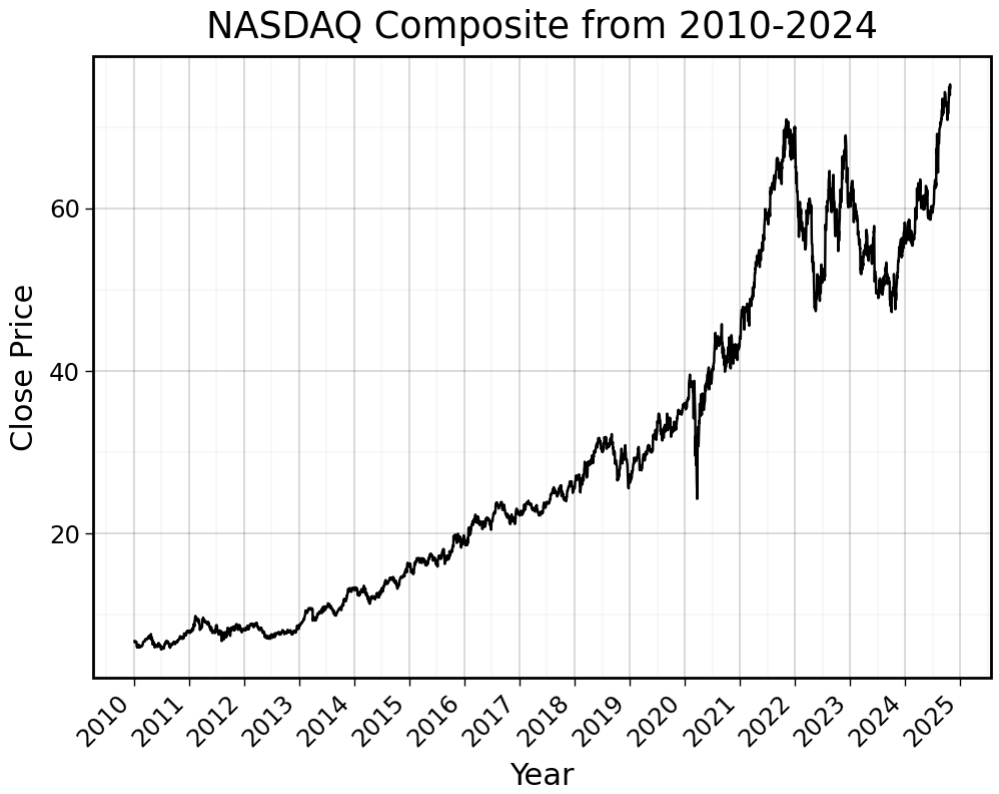


Figure 1: NASDAQ Composite from 2010-2024

**2. Cleaning Data**

```
# Cleaning data
import pandas as pd
nasdaq = pd.read_csv('data/nasdaq.csv')

# Converting to lowercase
nasdaq.columns = nasdaq.columns.str.lower()

# Creating 'month variable for seasonality'
nasdaq['date'] = pd.to_datetime(nasdaq['date'])
nasdaq['month'] = nasdaq['date'].dt.month

# Adding moving average to dataset (MA)
nasdaq['10_day_ma'] = nasdaq['close'].ewm(span=10, adjust=False).mean()
```

**3. Splitting Data into Training and Testing**

Because this is a time-series model, the split will not be a random split as we want to see how the model performs on unknown future data. The test data will be all dates after July 1, 2024.

```
predict_start = pd.to_datetime('2024-07-01')
days_predict_start = (predict_start - nasdaq['date'].min()).days

nasdaq['date'] = pd.to_datetime(nasdaq['date'])
nasdaq['days_since_start'] = (nasdaq['date'] - nasdaq['date'].min()).dt.days
```

```
X = ['days_since_start', 'open', 'high',
    'low', 'volume', 'interestrate',
    'exchangerate', 'vix','tedspread',
    'effr', 'gold', 'oil',
    'month', '10_day_ma']
y = 'close'

#Train-test split
train_data = nasdaq[nasdaq['days_since_start'] < days_predict_start]
test_data = nasdaq[nasdaq['days_since_start'] >= days_predict_start]

#Train data
X_train = train_data[X]
y_train = train_data[y]

#Test data
X_test = test_data[X]
y_test = test_data[y]
```

#### 4. Running the Model

```
import xgboost as xgb
from sklearn.metrics import mean_squared_error

eval_set = [(X_train, y_train), (X_test, y_test)]

# Initialize the model
xgb_nasdaq = xgb.XGBRegressor(
                            n_estimators=1000,
                            learning_rate=0.01,
                            max_depth=3,
                            random_state=42,
                            eval_metric="rmse")

xgb_nasdaq.fit(X_train,
                y_train,
                eval_set=eval_set,
                verbose=False)

y_pred = xgb_nasdaq.predict(X_test)

mse = mean_squared_error(y_test,y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 6.2701337284484
```

#### 5. Training and Testings RMSE

```
import numpy as np
import matplotlib.pyplot as plt

results = xgb_nasdaq.evals_result()

train_rmse = np.array(results['validation_0']['rmse'])
```

```
test_rmse = np.array(results['validation_1']['rmse'])

# Normalize RMSE
train_rmse_normalized = train_rmse / train_rmse[0]
test_rmse_normalized = test_rmse / test_rmse[0]

# Plot
plt.figure(figsize=(6, 4))
plt.plot(train_rmse_normalized, label="Training", color='blue')
plt.plot(test_rmse_normalized, label="Testing", color='red')
plt.xlabel("Number of Boosting Rounds")
plt.ylabel("Root Mean Squared Error (RMSE)")
plt.title("Training and Testing RMSE Over Boosting Rounds")
plt.legend()
plt.grid()
plt.show()
```

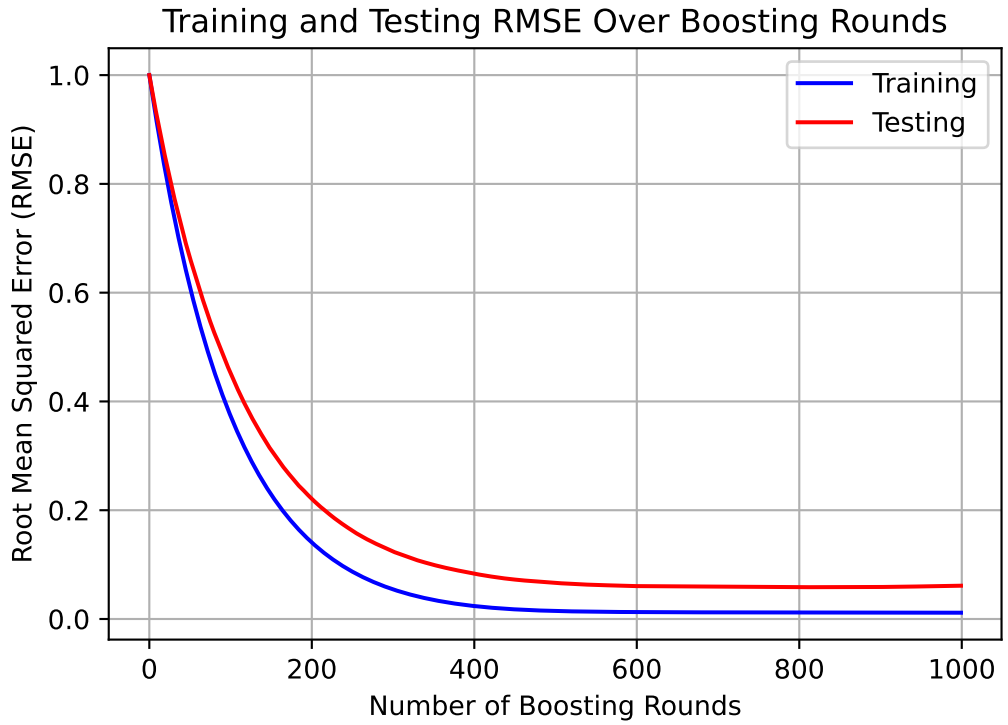


Figure 2: Training and Testing RMSE Over Boosting Rounds

**6. Depicting Model Performance on a Graph**

```
train_pred = xgb_nasdaq.predict(X_train)

train_results_df = pd.DataFrame({
    'date': train_data['date'],
    'actual': y_train,
    'predicted': train_pred})

test_results_df = pd.DataFrame({
    'date': test_data['date'],
```

```
    'actual': y_test,
    'predicted': y_pred})

combined_results_df = pd.concat([train_results_df, test_results_df], axis=0)

df_melted = pd.melt(combined_results_df, id_vars=['date'],
                                    value_vars =['actual', 'predicted'],
                                    var_name='type', value_name='actual_price')

df_melted.loc[(df_melted['date'] < pd.to_datetime(
    '2024-07-01')) & (df_melted['type'] == 'predicted'), 'actual_price'] = None

# Start date for graph
start_date = pd.to_datetime('2024-01-01')

# Filter to include only data from the start_date onward
df_melted_filtered = df_melted[df_melted['date'] >= start_date]

# Plot the data
(ggplot(df_melted_filtered, aes(x='date', y='actual_price', color='type'))
    + geom_line()
    + geom_vline(xintercept=pd.to_datetime('2024-07-01'),
                 linetype='dashed',
                 color='gray',
                 size=1)
    + labs(title='Actual vs Predicted Close Price (2024)',
           x='Date',
           y='Close Price',
           color='Legend')
    + scale_color_manual(values={'actual': 'black',
                                 'predicted': 'red'})
    + scale_x_datetime(date_breaks='2 months', date_labels='%Y-%m')
    + theme_linedraw()
    + theme(figure_size = (6,4)))
```

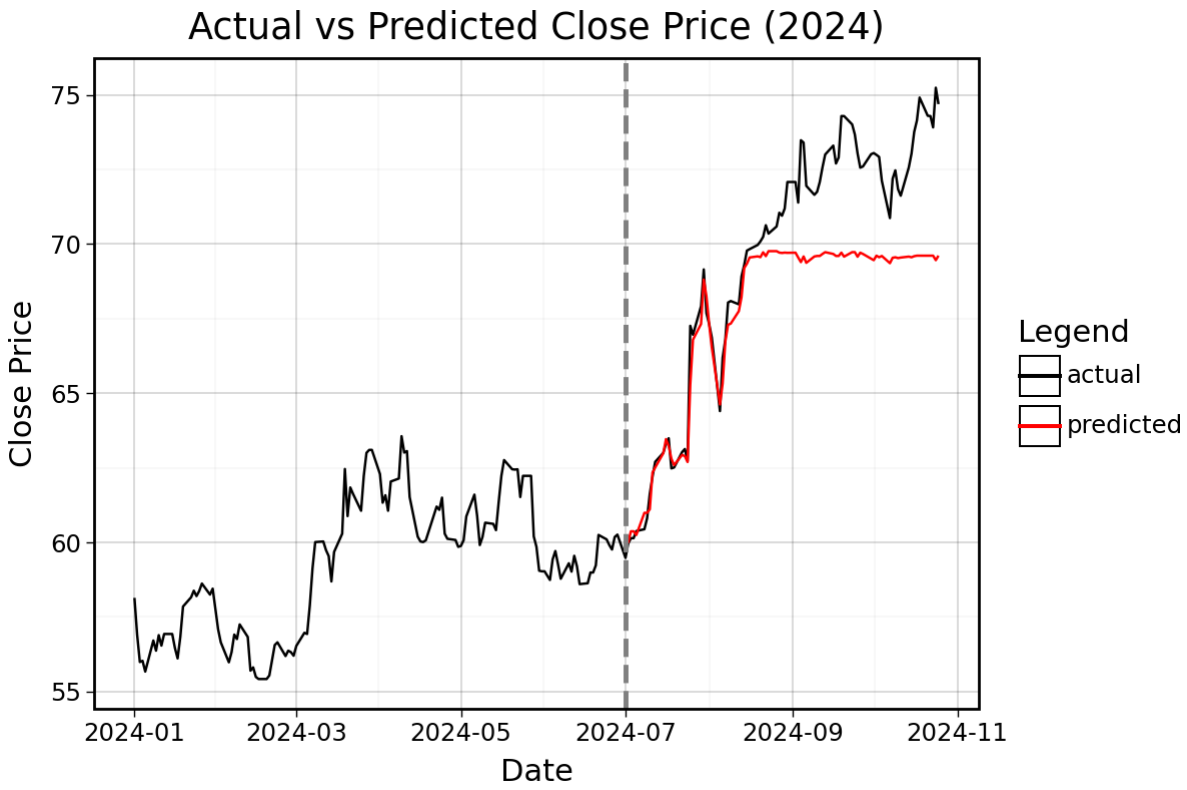


Figure 3: Actual vs Predicted Close Price (2024)

**7. Feature Importance**

```
importance = xgb_nasdaq.get_booster().get_score(importance_type='weight')

importance_df = pd.DataFrame(list(importance.items()), columns=['feature', 'importance'])

# Sort the features by importance
importance_df = importance_df.sort_values(by='importance', ascending=False)

importance_df['normalized_importance'] = importance_df[
    'importance'] / importance_df['importance'].sum()

# Plotting the feature importance using Plotnine
(ggplot(importance_df, aes(x='reorder(feature, normalized_importance)',
                           y='normalized_importance'))
    + geom_bar(stat='identity', fill='steelblue')
    + coord_flip()
    + theme_minimal()
    + labs(x='Feature', y='Importance', title='XGBoost Feature Importance')
    + theme(figure_size = (6,4)))
```

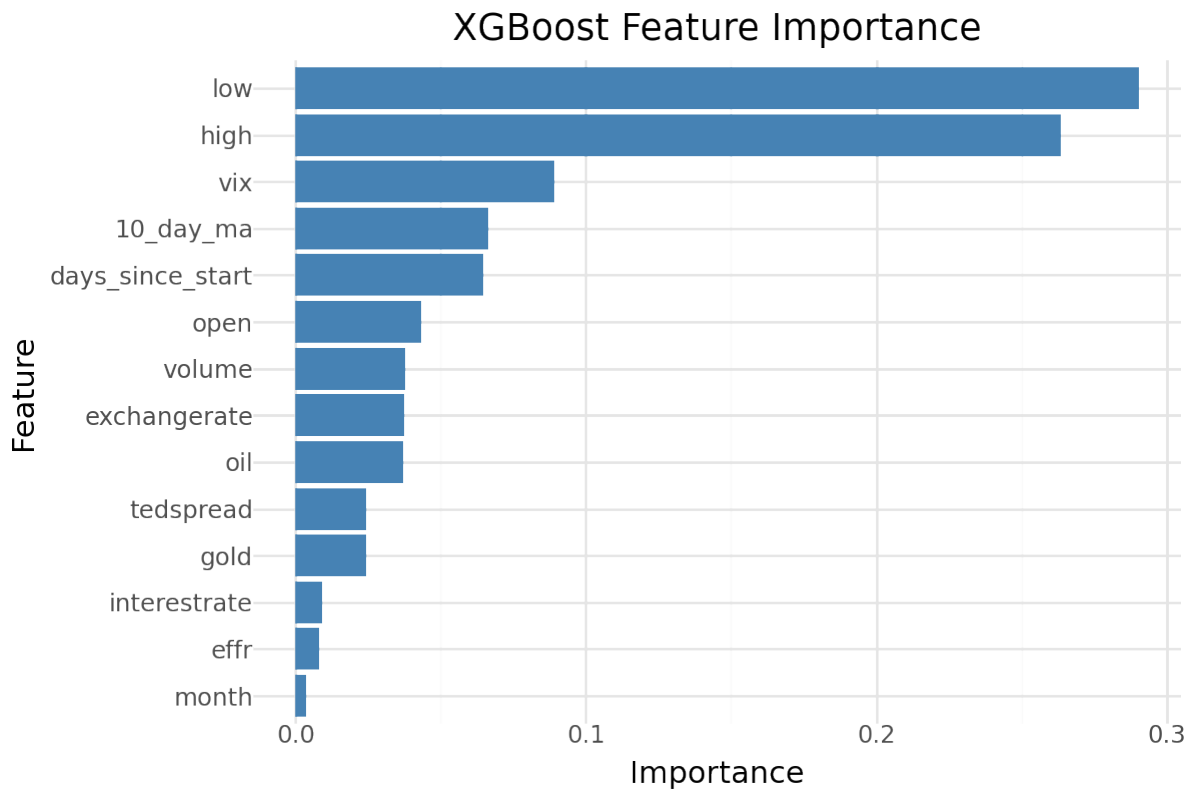


Figure 4: XGBoost Feature Importance