STOCK PREDICTION MODEL

PInak Mahapatra (22B0447)
Suyash Gahankari(22B0426)

ME 228
Applied Data Science and Machine Learning

INTRODUCTION

1) Thorough Analysis: We extensively examine NASDAQ Composite data from 2001 to 2024, employing various machine learning models to ensure a comprehensive understanding of market trends.

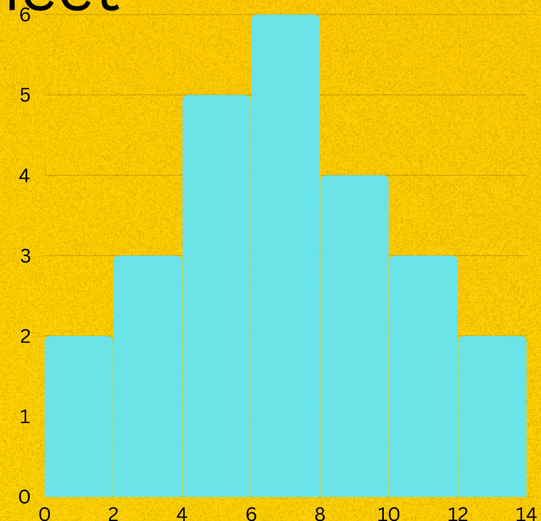
2) Advanced Techniques: Utilizing cutting-edge methods like LSTM, Random Forest, and XGBoost, we aim to extract intricate insights from the data, enhancing our predictive capabilities and analytical depth.

3) Significant Implications: Our approach not only forecasts future market movements but also uncovers key patterns and driving factors, offering valuable strategic insights for investors, analysts, and researchers navigating the complexities of financial markets.

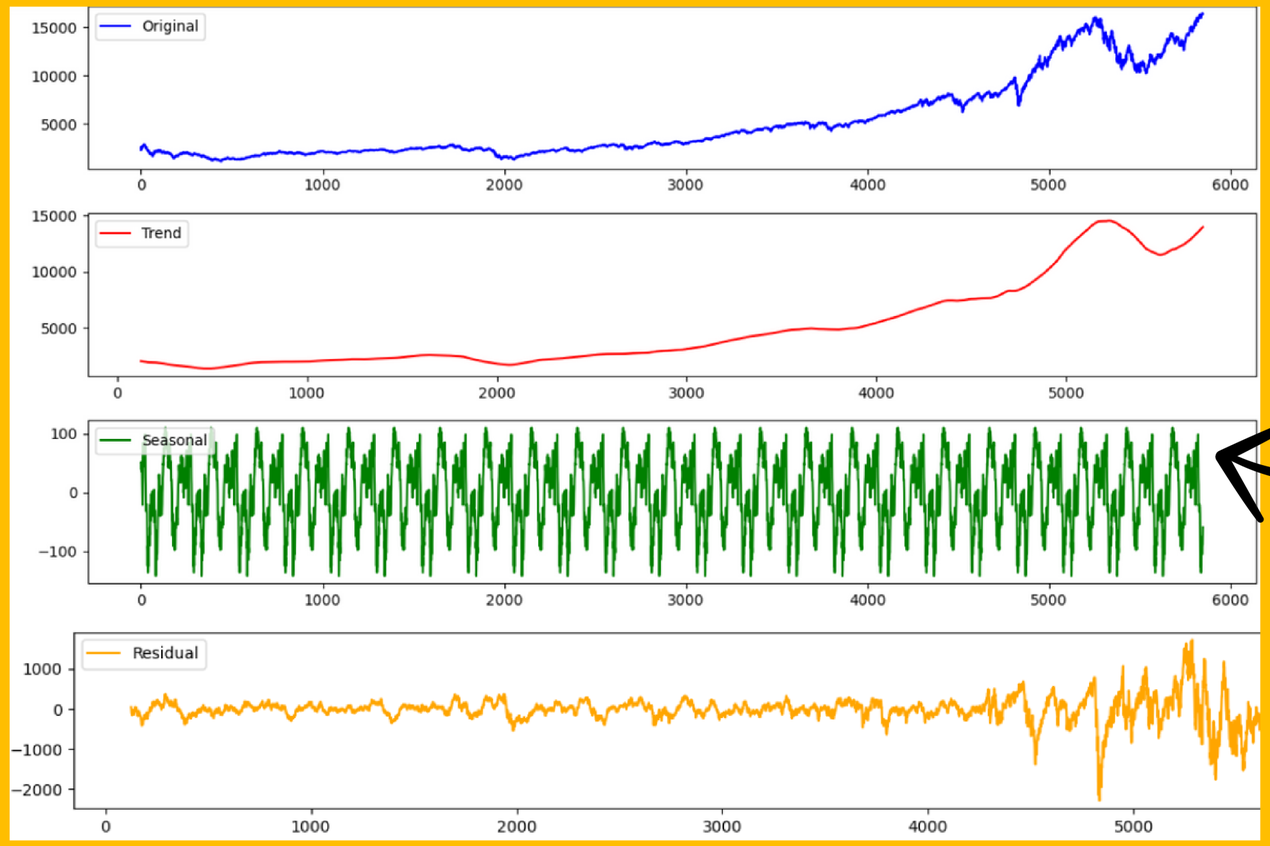
We have 7 columns namely Date, Open, High, Close, Low, Adj Close and Volume !

09-01-2001	2424.6899	2474.1599	2406.0801	2441.3	2441.3	1.975E+09
10-01-2001	2392.71	2525.28	2376.49	2524.1799	2524.1799	2.47E+09
11-01-2001	2495.5901	2661.9299	2495.01	2640.5701	2640.5701	2.843E+09
12-01-2001	2639.5601	2699.8701	2589.6299	2626.5	2626.5	2.519E+09
16-01-2001	2631.49	2638.22	2576.95	2618.55	2618.55	2.074E+09
17-01-2001	2710.53	2756.6299	2668.48	2682.78	2682.78	2.81E+09
18-01-2001	2696.74	2769.98	2661.26	2768.49	2768.49	2.559E+09
19-01-2001	2838.3601	2841.25	2752.0601	2770.3799	2770.3799	2.697E+09
22-01-2001	2759.1001	2789.6299	2722.96	2757.9099	2757.9099	2.037E+09
23-01-2001	2759.26	2845.3899	2736.28	2840.3899	2840.3899	2.278E+09
24-01-2001	2850.74	2892.3601	2828.3201	2859.1499	2859.1499	2.567E+09
25-01-2001	2836.3501	2849.5601	2753.3701	2754.28	2754.28	2.298E+09
26-01-2001	2705.3899	2785.6201	2686.6499	2781.3	2781.3	2.269E+09
29-01-2001	2757.29	2840.02	2742.5	2838.3401	2838.3401	1.97E+09
30-01-2001	2845.01	2861.71	2817.1299	2838.3501	2838.3501	2.074E+09
31-01-2001	2848.1101	2872.47	2772.3501	2772.73	2772.73	2.277E+09
01-02-2001	2771.5701	2796.8899	2742.4399	2782.79	2782.79	1.776E+09

Snapshot of our Datasheet

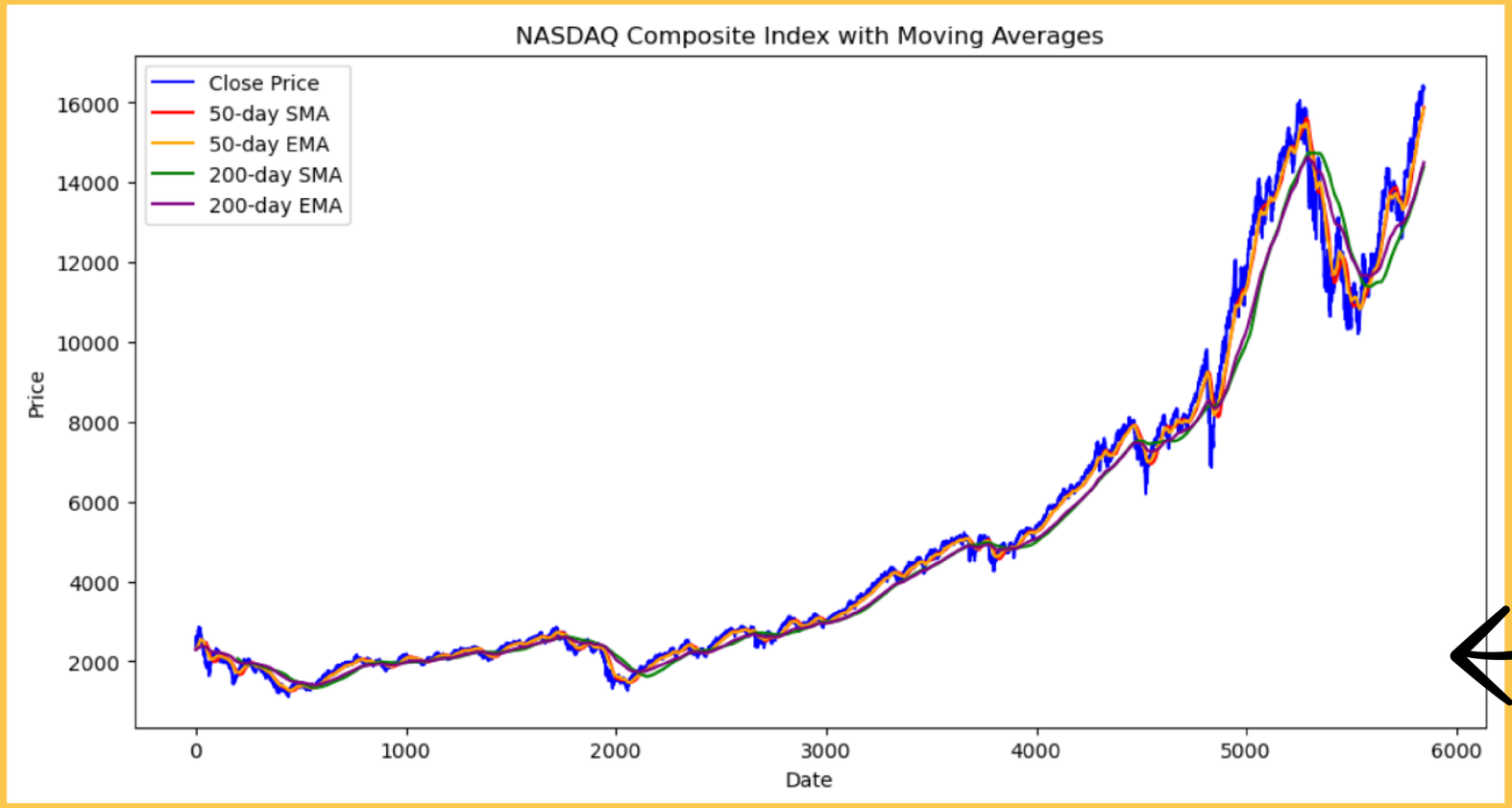


We performed time series decomposition on the closing prices of the NASDAQ Composite index. It decomposes the series into trend, seasonal, and residual components using an additive model with an assumed annual seasonality period of 252 trading days. Finally we plot the original series and the decomposition components for visualization.



Snapshot of our Time Series of Decomposition

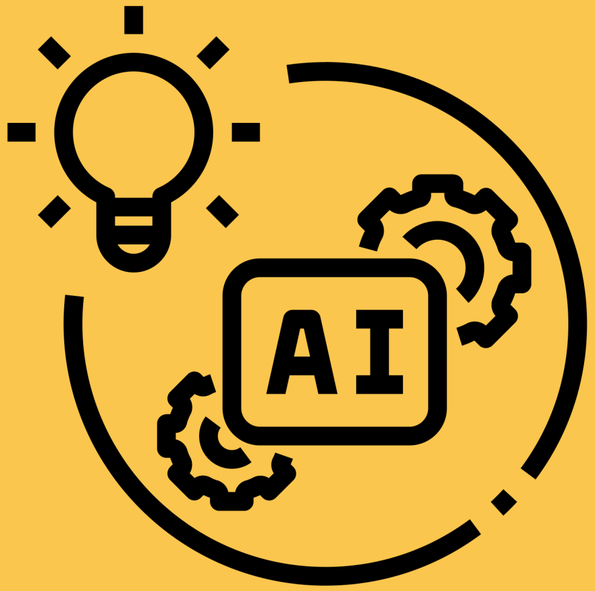
We compute both Simple Moving Averages (SMA) and Exponential Moving Averages (EMA) for 50-day and 200-day periods using the closing prices of the NASDAQ Composite index. This enables visualization and analysis of short-term and long-term trends in the index's price movements.



Snapshot of our plots

ML MODELS WE IMPLEMENTED:

1) LSTM
2) XG Boost
3)Random Forest

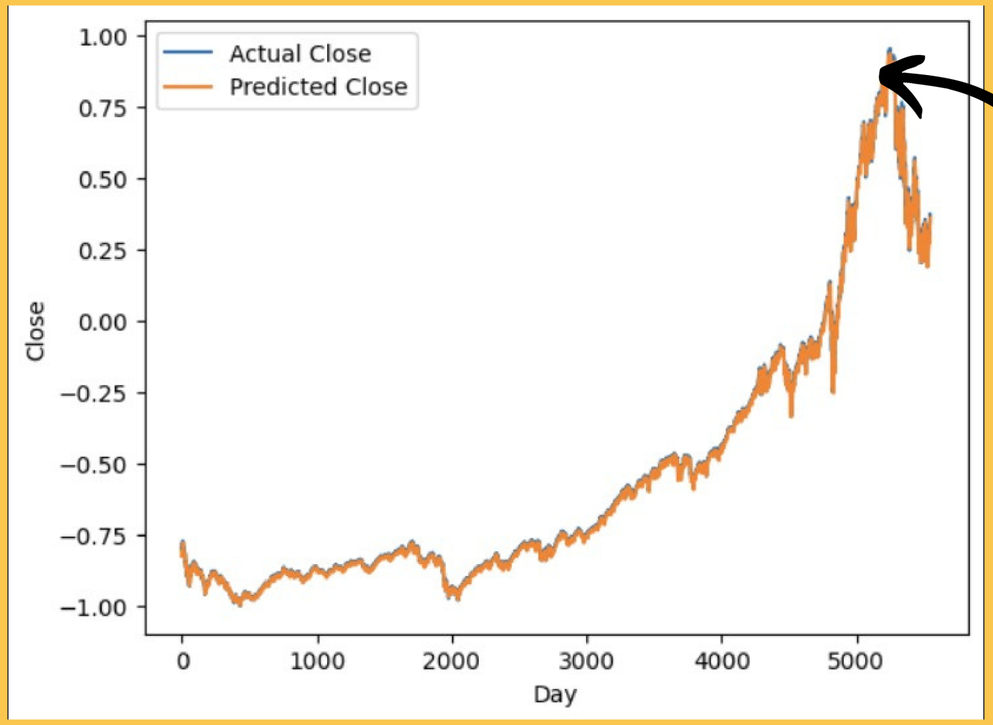


LSTM

1) LSTM (Long Short-Term Memory) networks are a specialized type of recurrent neural network (RNN) architecture.

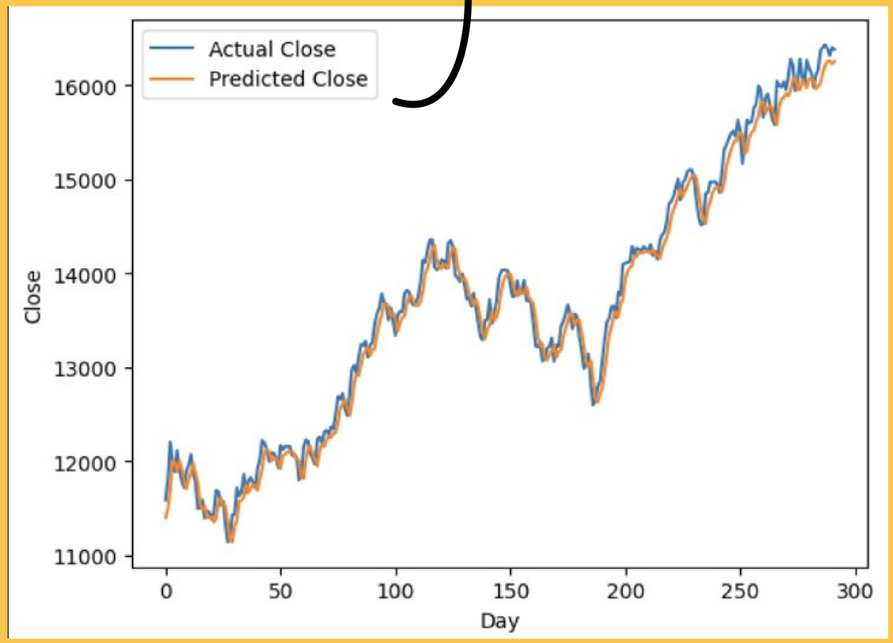
2) They address the vanishing gradient problem commonly encountered in traditional RNNs, enabling them to capture long-term dependencies in sequential data more effectively.

3) LSTM networks are particularly useful in tasks such as time series forecasting, natural language processing, and speech recognition due to their ability to retain and forget information selectively over time.



Plot of Validation and Training Data sets

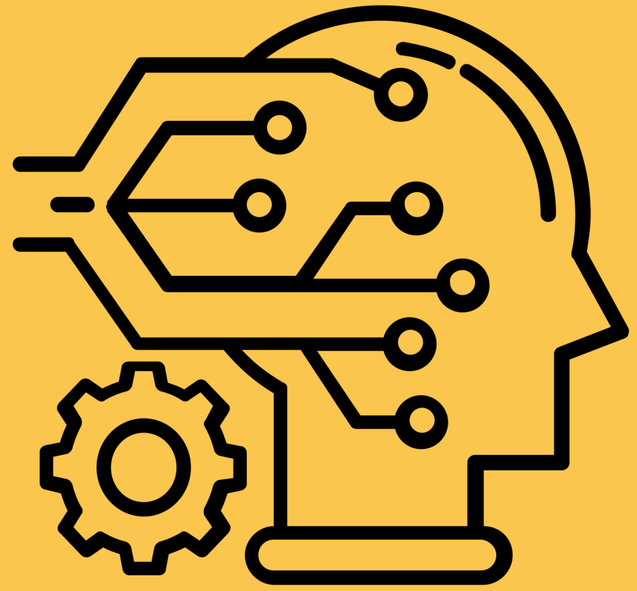
Plot of Testing Data



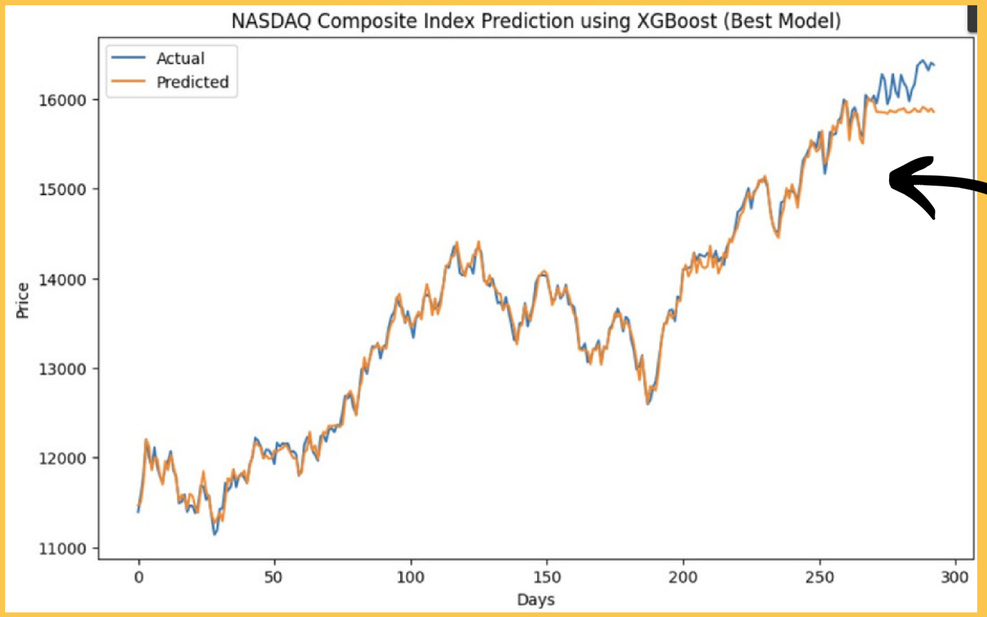

```
[63] from sklearn.metrics import mean_squared_error

     mse = mean_squared_error(new_y_test, test_predictions)
     rmse = np.sqrt(mse)

     print("Mean Squared Error:", mse)
     print("Root Mean Squared Error:", rmse)

     Mean Squared Error: 28528.19204926011
     Root Mean Squared Error: 168.90290716639578

[65] from sklearn.metrics import r2_score

     r2 = r2_score(new_y_test, test_predictions)
     print("R-squared (R2) Score:", r2)

     R-squared (R2) Score: 0.9852004015440785
```

Code snippet validating our results

XG Boost Regressor



1) XGBoost Regressor is a popular implementation of the gradient boosting algorithm known for its efficiency and effectiveness in regression tasks.
2) It builds an ensemble of weak decision trees sequentially, where each subsequent tree corrects the errors of the previous ones.
3) XGBoost Regressor utilizes techniques such as gradient boosting, regularization, and tree pruning to optimize predictive performance and handle complex datasets effectively.



Plot for Validation Data

Code snippet vallidating our findings

```
# Define hyperparameter grid for GridSearchCV
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.05, 0.1, 0.2]
}
# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error')

# Perform GridSearchCV with validation data
grid_search.fit(X_val, y_val)

# Get the best parameters
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
```

```
Best Parameters: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 300}
```

```
[138] # Predict on the test data with the best model
best_xgb = grid_search.best_estimator_
y_pred_best = best_xgb.predict(X_test)

# Calculate the Mean Squared Error with the best model
mse_best = mean_squared_error(y_test, y_pred_best)
print("Mean Squared Error (Best Model):", mse_best)
```

```
Mean Squared Error (Best Model): 15312.489394688331
```

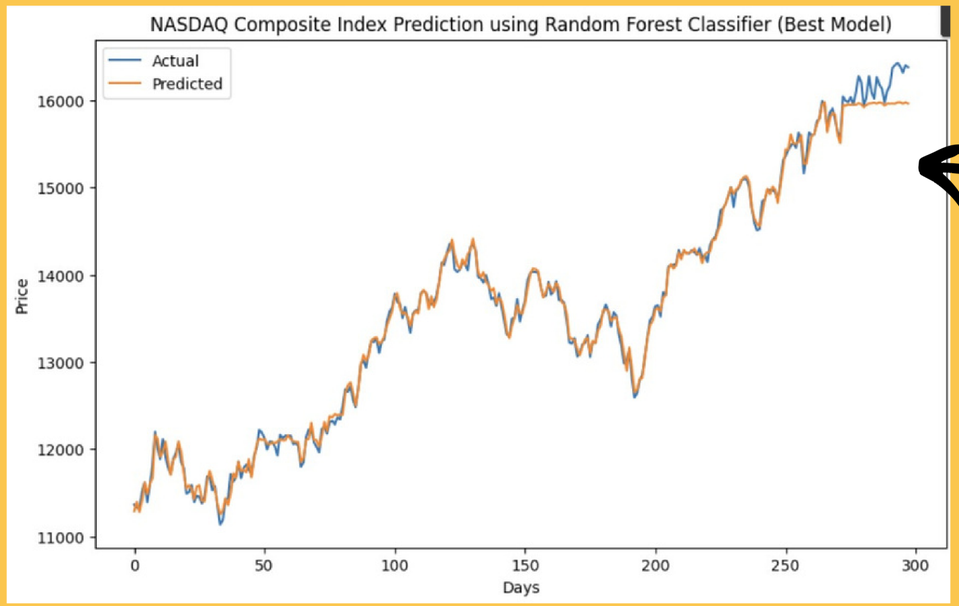
R2 Score Prediction

```
from sklearn.metrics import r2_score

r2 = r2_score(y_test.values, y_pred_best)
print("R-squared (R2) Score:", r2)
```

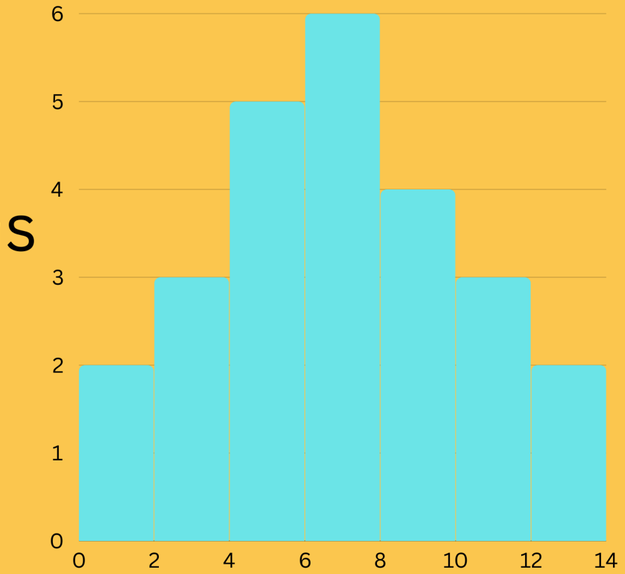
```
R-squared (R2) Score: 0.9921014740509322
```

Random Forest Regressor

1) Random Forest is an ensemble learning method based on decision trees, where multiple trees are trained on random subsets of the data.
2) Each tree in the forest independently predicts the target variable, and the final prediction is made by averaging or taking a vote over all trees.
3) RandomForest is known for its robustness to overfitting, ability to handle high-dimensional data, and capability to capture complex relationships in the data, making it widely used in regression and classification tasks.



Plot showing predictions in randomforest prediction



MSE and R2 Value

```
test_mse = mean_squared_error(y_test, y_test_pred)
print("Best n_estimators:", best_n_estimators)
print("MSE on testing set:",test_mse)
```

```
Best n_estimators: 100
MSE on testing set: 9776.49874458101
```

```
[148] from sklearn.metrics import r2_score

      y_test_pred = final_rf_regressor.predict(X_test)

      r2_test = r2_score(y_test, y_test_pred)
      print("R-squared score on testing set:",r2_test)
```

```
R-squared score on testing set: 0.9950834856780054
```

Results And Findings !

1) Our analysis identified the RandomForest Regressor as the top performer, boasting the lowest MSE and highest R-squared. Yet, in time series analysis, LSTM models excel due to their adeptness in capturing temporal dependencies and sequential patterns.

2) LSTM's prowess stems from its specialized memory cells, designed to retain information over time, its ability to comprehend the temporal structure inherent in sequential data, and its dynamic parameter learning, which adjusts to changing patterns.

3) While RandomForest Regressor excelled in our study, LSTM models offer a compelling alternative for forecasting tasks, particularly in dynamic markets where historical context is paramount.


THANK
YOU