FORECASTING THE DOW JONES INDUSTRIAL AVERAGE WITH TIME SERIES MODELS

INTRODUCTION

The Dow Jones Industrial Average (DJIA) is a stock market index that tracks the performance of 30 large, publicly traded companies listed on stock exchanges in the United States. It is considered a key indicator of the overall health of the U.S. economy and is closely watched by investors around the world.

Accurately forecasting the DJIA can be a valuable tool for financial planning and investment decisions. However, due to the complex and dynamic nature of financial markets, predicting future trends can be challenging. This project explores the use of various time series forecasting models to predict the movement of the DJIA.

While the Reserve Bank of India (RBI) website does not directly provide real-time stock market data, it can be a valuable resource for understanding the economic context surrounding the DJIA. The RBI's annual reports often discuss the performance of major international market indices, including the DJIA. This information can be used to identify potential factors influencing the DJIA's movement, which can then be incorporated into the forecasting models.

This report focuses on evaluating the effectiveness of several popular time series models for forecasting the DJIA. We will compare the performance of Autoregressive Integrated Moving Average (ARIMA), linear regression, Support Vector Regressor (SVR), XG Boost regressor, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) models against a baseline model. We will then perform hyperparameter tuning to optimize the performance of the most promising models. The ultimate goal is to identify the most effective model for forecasting the DJIA based on the chosen data timeframe (March 5th, 2003 to April 19th, 2024).

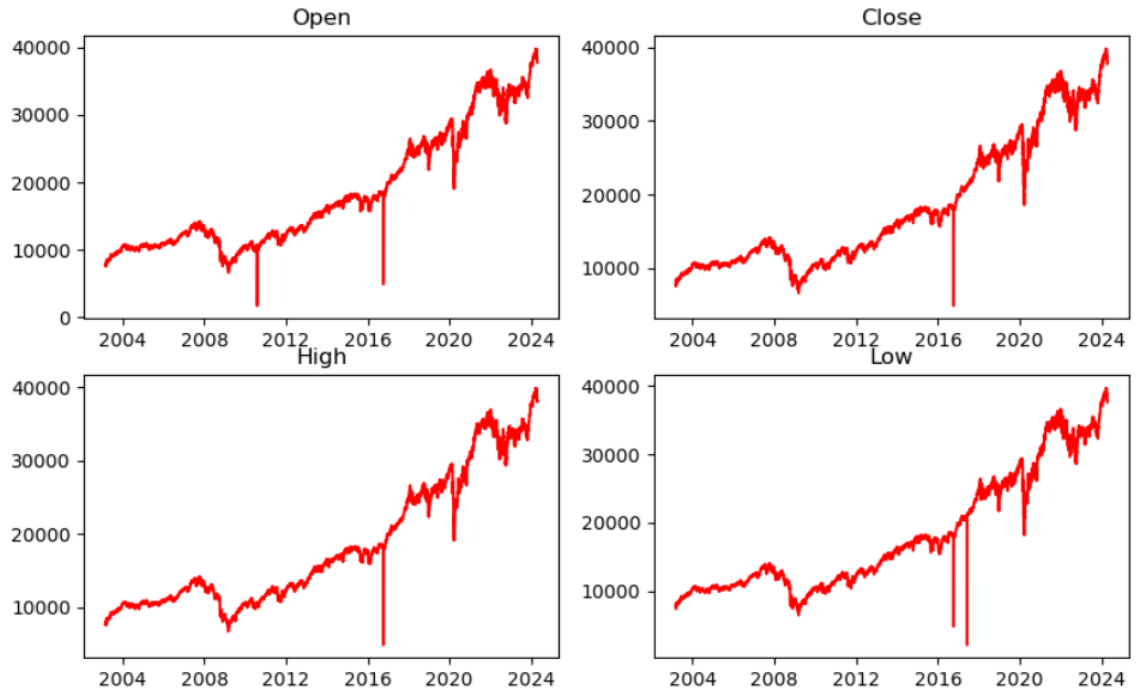
INITIAL EXPLORATION AND DATA CHARACTERISTICS

Our analysis begins with an exploration of the dataset, which spans the period from March 5th, 2003 to April 19th, 2024. The dataset includes four key features: closing index value, opening index value, highest value of the index on a given day, and lowest value of the index on a given day.

Date	Open	Close	High	Low
2003-03-03	7890.24	7837.86	7981.46	7822.73
2003-03-04	7838.14	7704.87	7845.71	7704.31
2003-03-05	7702.35	7775.60	7775.60	7661.32
2003-03-06	7774.76	7673.99	7777.42	7659.09
2003-03-07	7671.75	7740.03	7743.82	7562.65
...	...	...	...	...
2024-04-15	38075.38	37735.11	38386.81	37657.79
2024-04-16	37992.22	37798.97	37992.22	37713.70
2024-04-17	37949.67	37753.31	38036.70	37611.56
2024-04-18	37847.21	37775.38	38083.76	37681.52
2024-04-19	37801.98	37986.40	38102.57	37781.61

5211 rows × 4 columns

Visualizing these features through time series plots reveals a consistent upward trend across all four categories. This initial observation suggests that the data might exhibit non-stationarity, a characteristic that needs to be addressed before applying a time series forecasting model (such as ARIMA) effectively. For our case, we will be focussing solely on the closing index value. The following sections will detail the specific models explored and their performance evaluation.



BASELINE MODEL: PERSISTENCE FORECASTING

Establishing a baseline performance metric is crucial for evaluating the effectiveness of more complex forecasting models. In this project, we opt for a persistence model, also known as a naive forecast, as our baseline. This simple model assumes that the next day's value will be the same as the current day's value.

We evaluated the performance of the persistence model using the Root Mean Squared Error (RMSE) metric. RMSE measures the average magnitude of the difference between the predicted and actual values.

When the data was scaled, the baseline model achieved an RMSE value of 0.043. When the model was applied to unscaled data, the RMSE came out to be 379.363. The unscaled RMSE will be useful for comparing the performance of baseline model against ARIMA, since we will not be scaling our data for ARIMA. The following sections will explore the performance of more sophisticated forecasting models and compare their effectiveness against the established baseline.

ARIMA

We will begin with simple models and gradually move towards more advanced ones. Therefore, we start by using the ARIMA model which stands for Autoregressive Integrated Moving Average, and it's a popular method for forecasting time series data. It's essentially a statistical model that analyses past data points to predict future trends. ARIMA can be broken down into the following three components:

- **Autoregressive:** This refers to the model's ability to use past values of the time series to predict future values. In simpler terms, it considers how past data points are influencing current and future values.
- **Integrated:** This part acknowledges that some time series data might have trends or seasonality. The integration process involves differencing the data (subtracting a previous value from a current value) to remove trends and make the data stationary (having constant statistical properties).
- **Moving Average:** This involves averaging past forecast errors (the difference between predicted and actual values) to account for randomness and smooth out fluctuations in the data

STATIONARITY TESTING AND CONSIDERATIONS:

To be able to use ARIMA it is necessary that the data is non stationary. To ensure that this condition is fulfilled we conduct an Augmented Dickey-Fuller (ADF) test to assess the stationarity of the time series data. The test results showed that the p-value (0.9754) was greater than the typical significance level (0.05) which suggests non-stationarity of the data.

HYPERPARAMETER TUNING FOR THE ARIMA MODEL:

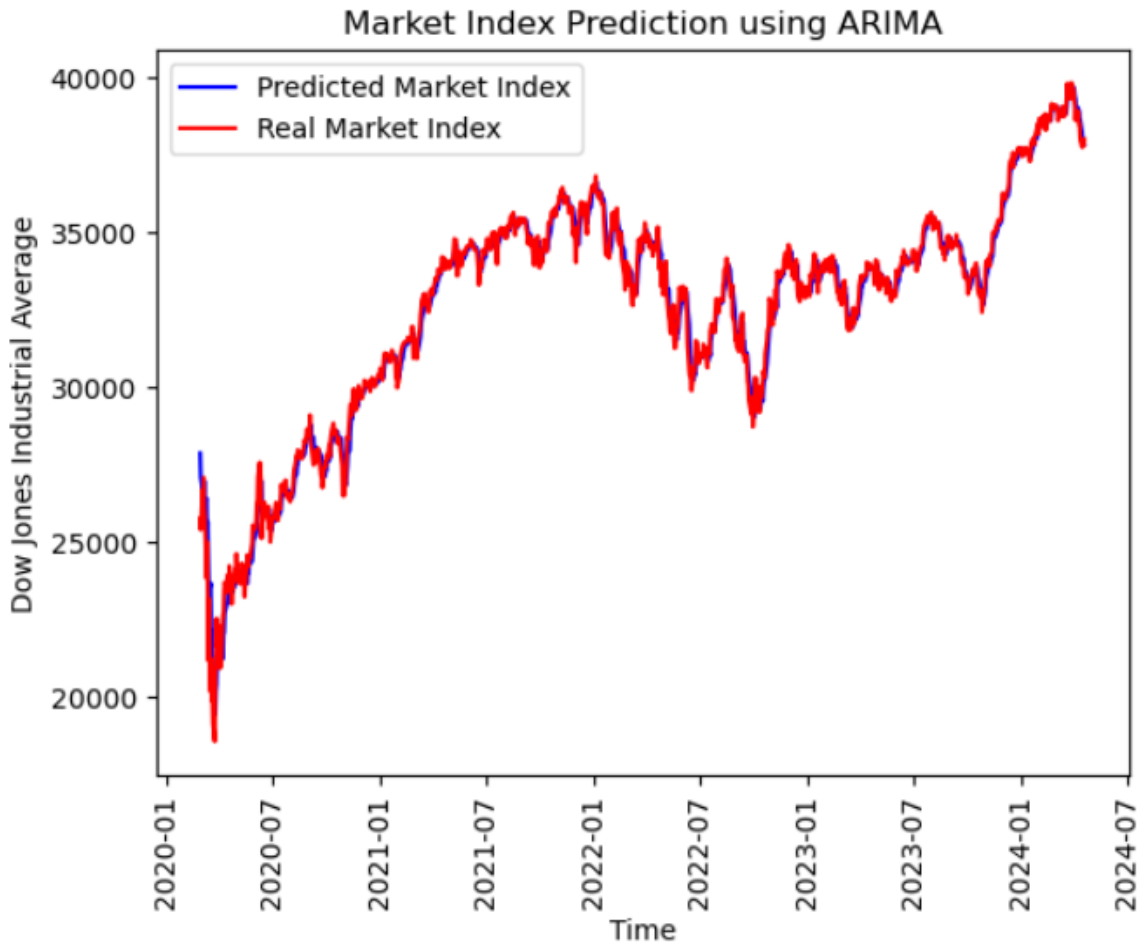
We will use a random search cross-validation (RandomSearchCV) approach with auto_arima from the pmdarima library to identify potentially suitable values for the ARIMA model's hyperparameters (p, d, q). This is a well-suited approach for exploring a large space of hyperparameter combinations and identifying promising configurations. The results suggested that ARIMA(3, 1, 2) achieved the lowest AIC score among the explored models.

WALK-FORWARD VALIDATION AND PERFORMANCE:

To obtain a reliable estimate of model performance, we will employ walk-forward validation. In this technique, the model is trained on a portion of the data, predictions are made for the next few periods, and then the model is refitted on the expanded dataset (including the newly predicted values). This process iterates throughout the data, providing a more robust assessment of the model's generalizability. Using this technique the reported Root Mean Squared Error (RMSE) of 441.748 was higher than the baseline model's RMSE (379).

DISCUSSION AND NEXT STEPS:

While ARIMA offers the advantage of explicitly considering future values during forecasting but its performance in this specific case fell short of the baseline RMSE (379.363).


Market Index Prediction using ARIMA
Predicted Market Index
Real Market Index
Dow Jones Industrial Average
40000
35000
30000
25000
20000
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
2024-07
Time

LINEAR REGRESSION

Linear regression is a popular machine learning technique for modelling relationships between variables. In time series forecasting, it can be leveraged to capture the influence of past observations on future values. This section explores the application of linear regression with lagged features for predicting the Dow Jones Industrial Average (DJIA).

Lagged Feature Engineering:

To incorporate past information into the model, we will employ lagged features. These features represent the closing price of the DJIA shifted by a certain number of days (lags) in the past. We will train a series of ten linear regression models, each incorporating an increasing number of lagged closing prices as features (from 1 to 10 lags). This approach aims to identify the optimal number of lags that best captures the historical trends influencing future closing prices.

Date	Close	Close_l1	Close_l2	Close_l3	Close_l4	Close_l5	Close_l6	Close_l7	Close_l8	Close_l9	Close_l10
2003-03-17	8141.92	7859.71	7821.75	7552.07	7524.06	7568.18	7740.03	7673.99	7775.60	7704.87	7837.86
2003-03-18	8194.23	8141.92	7859.71	7821.75	7552.07	7524.06	7568.18	7740.03	7673.99	7775.60	7704.87
2003-03-19	8265.45	8194.23	8141.92	7859.71	7821.75	7552.07	7524.06	7568.18	7740.03	7673.99	7775.60
2003-03-20	8286.60	8265.45	8194.23	8141.92	7859.71	7821.75	7552.07	7524.06	7568.18	7740.03	7673.99
2003-03-21	8521.97	8286.60	8265.45	8194.23	8141.92	7859.71	7821.75	7552.07	7524.06	7568.18	7740.03
...	...	...	...	...	...	...	...	...	...	...	...
2024-04-15	37735.11	37983.24	38459.08	38461.51	38883.67	38892.80	38904.04	38596.98	39127.14	39170.24	39566.85
2024-04-16	37798.97	37735.11	37983.24	38459.08	38461.51	38883.67	38892.80	38904.04	38596.98	39127.14	39170.24
2024-04-17	37753.31	37798.97	37735.11	37983.24	38459.08	38461.51	38883.67	38892.80	38904.04	38596.98	39127.14
2024-04-18	37775.38	37753.31	37798.97	37735.11	37983.24	38459.08	38461.51	38883.67	38892.80	38904.04	38596.98
2024-04-19	37986.40	37775.38	37753.31	37798.97	37735.11	37983.24	38459.08	38461.51	38883.67	38892.80	38904.04

5201 rows × 11 columns

WALK-FORWARD VALIDATION:

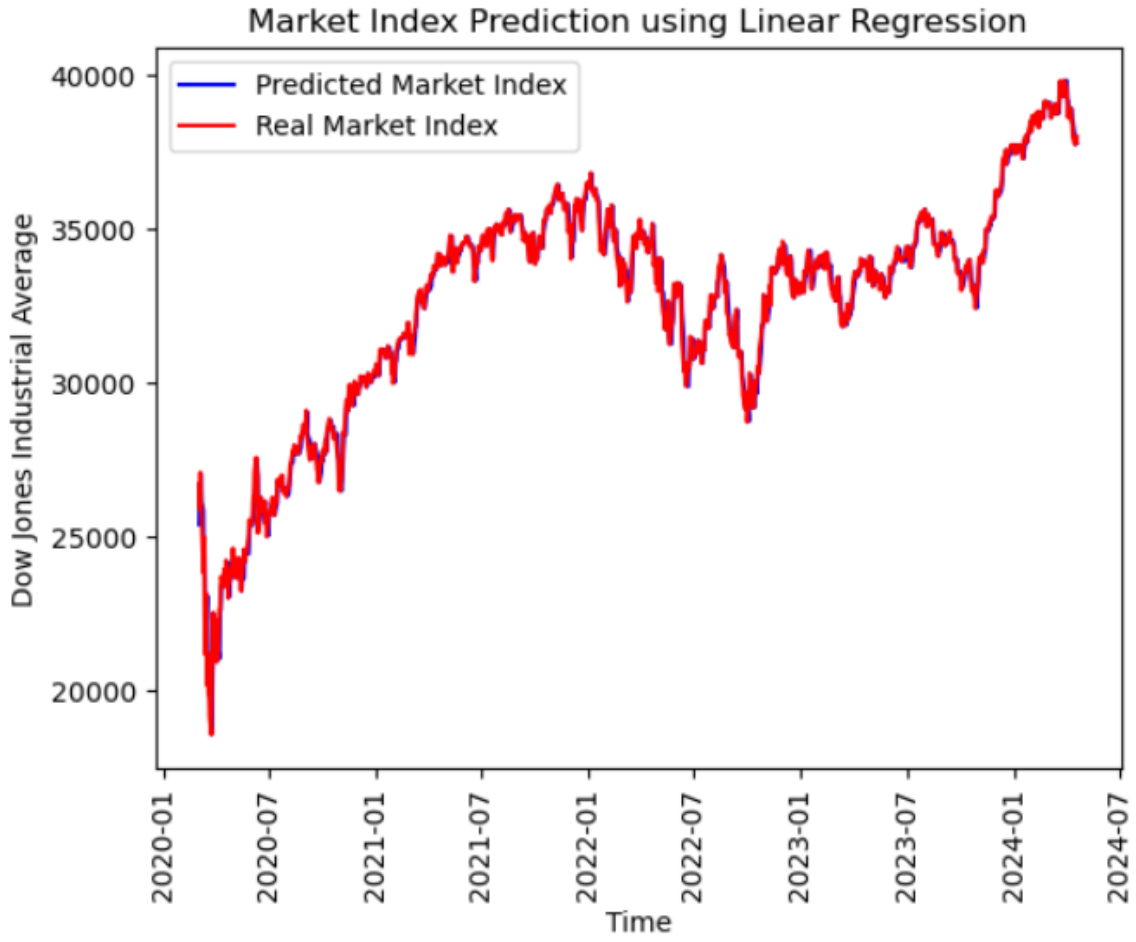
Just like ARIMA, we will use walk forward validation for linear regression as well.

RESULTS AND DISCUSSION:

The results of the linear regression models with varying lags are summarized in the following table:

Model with Lags	Mean Squared Error (MSE)
1	0.04316979
2	0.04297169
3	0.04670541
4	0.04665938
5	0.04788366
6	0.04795188
7	0.04815144
8	0.04817325
9	0.04804128
10	0.04768269

As evident from the table, the model incorporating two lags (lags = 2) achieved the lowest Mean Squared Error (MSE) of 0.04297169. This indicates that using the closing price from two days prior alongside the current day's closing price resulted in the most accurate predictions compared to other lag configurations. Interestingly, the performance improved slightly from the baseline model (persistence model with MSE of 0.043). However, including more than two lags resulted in a decline in model performance, suggesting that a simpler model with fewer features might be more effective in this case.


Market Index Prediction using Linear Regression
Predicted Market Index
Real Market Index
Dow Jones Industrial Average
40000
35000
30000
25000
20000
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
2024-07
Time

SUPPORT VECTOR REGRESSOR

Similar to the linear regression approach, we will explore the use of Support Vector Regressor (SVR) for forecasting the DJIA. SVR is another machine learning technique known for its ability to handle non-linear relationships in data. Here, we investigate the impact of incorporating lagged features on the SVR model's performance.

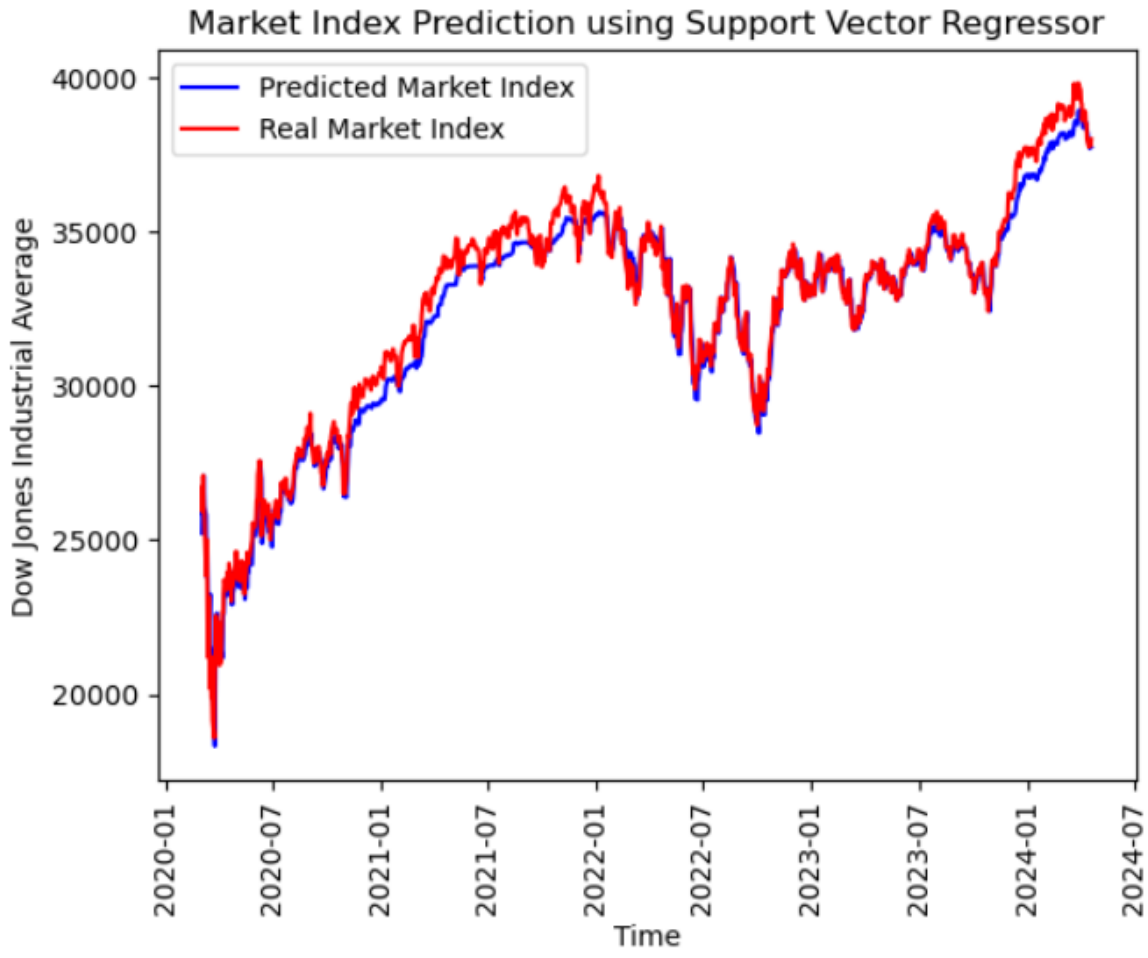
Lagged Feature Engineering and Walk-Forward Validation:
Following the same strategy as linear regression, we will construct SVR models with an increasing number of lagged closing prices (1 to 10 lags) as features. Walk-forward validation was again employed to obtain a robust evaluation of model generalizability.

RESULTS AND DISCUSSION:

The results for the SVR models with varying lags are presented in the table below:

Model with Lags	Mean Squared Error (MSE)
1	0.06914443
2	0.06974774
3	0.07035783
4	0.07078247
5	0.07153883
6	0.07203200
7	0.07194167
8	0.07249524
9	0.07275771
10	0.07326606

As can be seen, none of the SVR models with lagged features achieved a lower Mean Squared Error (MSE) compared to the best performing linear regression model (lags = 2, MSE = 0.04297169). In fact, all the SVR models exhibited significantly higher MSE values, ranging from 0.0691 (lags = 1) to 0.0733 (lags = 10). This suggests that under the current configuration, SVR might not be as suitable for this specific forecasting task compared to linear regression with lagged features.



XG BOOST REGRESSOR

Having explored linear regression and SVR, we now investigate the performance of XG Boost regressor, a popular ensemble machine learning technique known for its effectiveness in various forecasting tasks.

Lagged Feature Engineering and Walk-Forward Validation:
Similar to the previous models, XG Boost models will be trained with an increasing number of lagged closing prices (1 to 10 lags) as features. Walk-forward validation was again employed to ensure robust model evaluation.

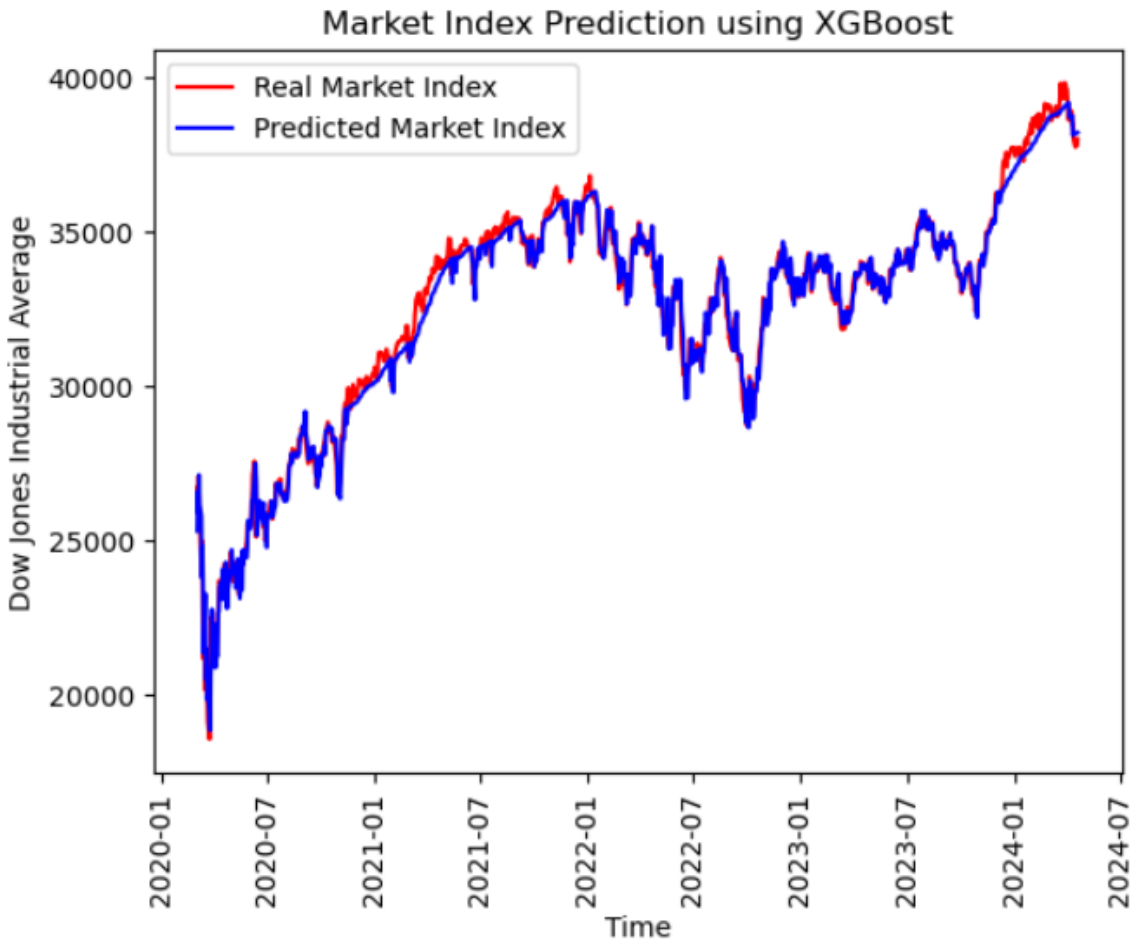
RESULTS AND DISCUSSION:

The results for the XG Boost models with varying lags are summarized in the following table:

Model with Lags	Mean Squared Error (MSE)
1	0.05205245
2	0.05348808
3	0.05464858
4	0.05435702
5	0.05433300
6	0.05391587
7	0.05398271
8	0.05451661
9	0.05479842
10	0.05443642

The XG Boost models exhibited a mixed performance compared to the baseline and previously explored models. While none surpassed the best performing linear regression model (lags = 2, MSE = 0.04297169), the XG Boost model with one lag (lags = 1, MSE = 0.05391587) achieved a lower MSE compared to all SVR models and some of the linear regression models with higher lag configurations.

While not the overall leader, XG Boost's performance justifies its consideration for further exploration. Unlike SVR, which underperformed without hyperparameter tuning, XG Boost offers more flexibility in terms of parameter optimization.



LONG SHORT-TERM MEMORY (LSTM) NETWORKS

Recurrent neural networks (RNNs) like LSTMs are powerful tools for time series forecasting, particularly when dealing with sequential data and potential long-term dependencies. We will explore the use of an LSTM model to capture potential temporal patterns in the DJIA closing prices.

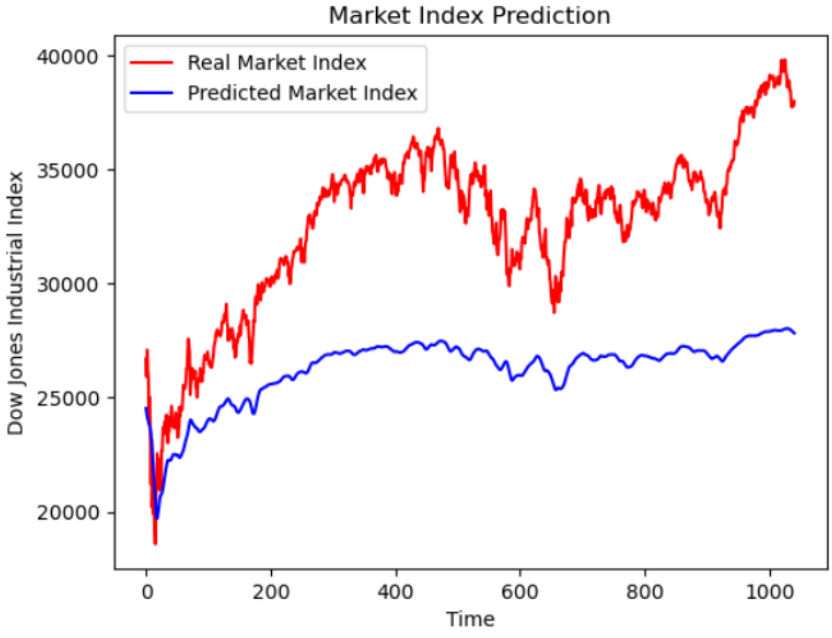
MODEL ARCHITECTURE AND TRAINING:

Unlike previous models that relied on lagged features and walk-forward validation, the LSTM model directly utilizes the entire sequence of closing prices as input. The model architecture consists of two LSTM layers, the first with 50 units and the second with 30 units, followed by a final dense layer with one unit for single-step forecasting. The model will be trained for 20 epochs.

RESULTS AND DISCUSSION:

Unfortunately, the LSTM model's performance was not encouraging. The Mean Squared Error (MSE) on the test set reached a concerningly high value of 1.24. This suggests that the LSTM struggled to learn effective representations of the time series data and make accurate predictions.

A prominent reason for the disappointing performance was the fact that LSTMs work poorly with data that is non stationary, i.e., data which has an increasing or decreasing trend. The DJIA clearly has an upward trend. In 2003, the index value started around 7000 and by 2024 it reached roughly around 35000. Had the index value stayed around 7000, LSTM was likely to outperform the previous models. One way to fix this would be to train the LSTM only around the rising portion of the index curve, but that would have had two drawbacks – One, we would not have been able to compare the performance of the LSTM with the other models. Two, LSTM would not be able to predict the future because the DJIA is likely to drift upwards in the future. As is evident from the graph below, LSTM was performing well in the initial stages of the series when the data exhibited less upward drift, but in the later time periods when the drift began to increase LSTM was continuously underpredicting the index value.



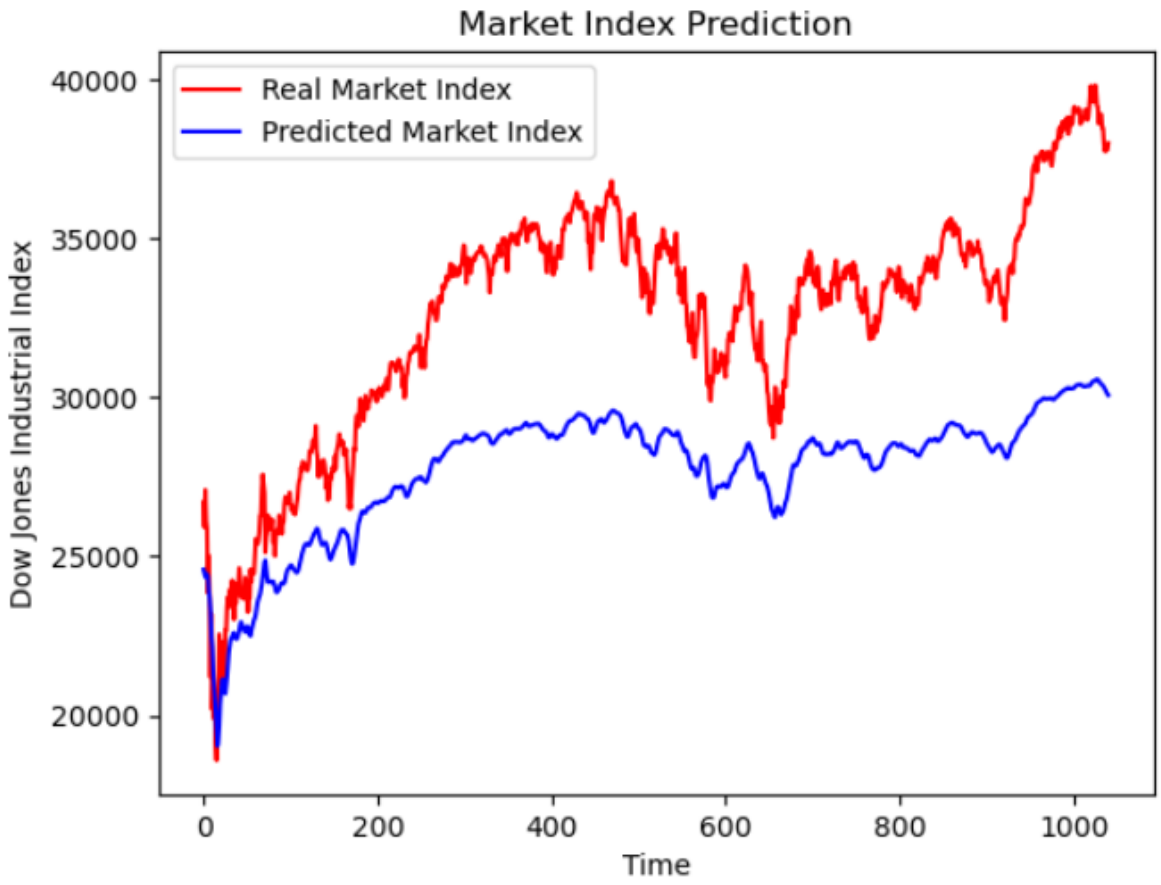
GATED RECURRENT UNIT (GRU) NETWORKS

Similar to LSTMs, GRUs (Gated Recurrent Units) are a type of recurrent neural network well-suited for time series forecasting. We investigate whether a GRU model could outperform the LSTM in handling the potential long-term dependencies within the DJIA closing prices.

Model Architecture and Training:
The GRU model architecture mirrored the LSTM configuration, with two GRU layers (first with 40 units, second with 30 units) followed by a final dense layer with one unit for single-step forecasting. The model was trained for the same 20 epochs as the LSTM.

RESULTS AND DISCUSSION:

The GRU model exhibited marginally better performance compared to the LSTM, achieving a Mean Squared Error (MSE) of 0.95 on the test set. While this represents a slight improvement over the LSTM's MSE of 1.24, it still signifies a significant inability of the model to accurately capture the underlying trends and predict future values. Similar to the LSTM, the GRU likely struggled with the non-stationary nature of the data, particularly the strong upward trend.



HYPERPARAMETER TUNING

Therefore, we have shortlisted three models for the final hyperparameter tuning stage:

1. Linear Regression with 2 lags
2. XG Boost Regressor with 6 lags
3. ARIMA model

Let us begin with Linear regression

RIDGE REGRESSION WITH HYPERPARAMETER TUNING

In the previous exploration, we identified linear regression with lagged features as a promising model due to its low MSE. Here, we will perform a hyperparameter tuning process using Ridge regression to see if we can further improve its performance. We will focus on linear regression with two lagged features, the model that achieved the lowest MSE among non-tuned models.

Hyperparameter Grid Search:

Two key hyperparameters for Ridge regression will be explored:

- **alpha:** This parameter controls the strength of the regularization penalty. Higher alpha values correspond to stronger regularization.
- **solver:** This parameter specifies the algorithm used to solve the linear system of equations. I explored various solvers offered by scikit-learn to identify the most efficient one for this specific task.

A grid search will be employed, evaluating a combination of alpha values (0.0001, 0.001, 0.01, 0.1, 1, 10) and solvers ('auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga', 'lbfgs'). This resulted in a comprehensive exploration of 48 different model configurations.

Evaluation and Model Selection:

We will use walk-forward validation to assess the performance of each model configuration on a separate validation set. This approach helps prevent overfitting and provides a more reliable estimate of generalizability. The Mean Squared Error (MSE) on the validation set will be used as the performance metric to compare different configurations.

On running the search, the results indicated that the model with solver='sag' and alpha=1 achieved the lowest MSE on the validation set (0.07096155). This suggests that using the SAG solver and a moderate regularization strength (alpha=1) led to the best performance within the explored hyperparameter space.

Final Training and Performance:

The chosen model configuration (Ridge regression with solver='sag' and alpha=1) was then retrained on the combined training and validation set. This final model achieved an MSE of 0.0427 on the unseen test set, which is a significant improvement compared to the baseline model's MSE (i.e., 0.043).

Overall, the hyperparameter tuning process for Ridge regression proved successful. By carefully selecting the hyperparameters, we were able to improve the model's performance and surpass the baseline model.

XG BOOST HYPERPARAMETER TUNING

Following the success of ridge regression with hyperparameter tuning, we investigate the potential for improvement with XG Boost. Here's an analysis of the XG Boost hyperparameter tuning process and the results:

Selected Model and Tuning Strategy:

- Unlike the previous exploration where multiple lagged features were used, I focused on XG Boost with only one lagged feature as the input. This simplifies the model and potentially reduces overfitting.
- A manual grid search approach was employed to explore different combinations of two hyperparameters:
    - **n_estimators:** This parameter controls the number of decision trees used in the ensemble model. Higher values can lead to more complex models but also increase the risk of overfitting.
    - **learning_rate:** When a new tree is trained, it will be added to the existing trees. Before doing so, it will be multiplied by the learning_rate. Decreasing this hyperparameter reduces the likelihood of overfitting.

Hyperparameter Grid Search and Results:

- Sixteen different model configurations were evaluated using a combination of four values for n_estimators (150, 200, 250, 300) and four values for learning_rate (0.001, 0.01, 0.1, 1).
- The results suggest that higher learning rates (0.1 and 1) combined with various numbers of estimators (150, 200, 250, and 300) all achieved a lower MSE compared to the baseline model. The model with **n_estimators=200** and **learning_rate=1** achieved the lowest MSE of 0.0388.

Discussion and Next Steps:

- While XG Boost with hyperparameter tuning managed to surpass the baseline model's performance, it's important to acknowledge the stochastic nature of the algorithm. Randomness inherent in the training process can lead to slightly different results each time the model is trained.
- To account for this stochasticity, considering multiple models with similar performance (e.g., those highlighted) and averaging their predictions (ensemble forecasting) could be a good strategy. This can help reduce the variance in individual model predictions and potentially lead to more robust forecasts.

Overall, the XG Boost hyperparameter tuning yielded promising results. Several model configurations achieved lower MSE compared to the baseline.

ARIMA HYPERPARAMETER TUNING

Unlike earlier, where we used auto_arima from the pmdarima library to identify potentially suitable values for the ARIMA model's hyperparameters, we will use the ARIMA model class from statsmodels library because it would allow us to train the ARIMA on more values than the auto_arima features which restricts its random search only on a few values. The p and q values with the minimum AIC value came out to be 2 and 6. When the model was retrained using these values the MSE came out to be 436.37, which unfortunately does not beat the baseline model. In fact, there is nothing more that we can do more about the model.

CONCLUSION

Based on our analysis, the following models turned out to be the best among all:

1. Ridge regression with 2 period lag, solver = ‘sag’ and alpha = 1
2. XG Boost Regressor with 1 period lag, n_estimators = 200 and learning_rate = 1.