Time Series stock prediction: A comparative study of LSTM, XG-Boost, Random Forest

Author: Anirban Das.

Date: 11th February, 2024.

Introduction

In the dynamic realm of financial markets, the ability to predict stock prices remains a coveted yet challenging endeavour. Traders, investors, and researchers alike have sought to harness the power of advanced technologies and machine learning algorithms to gain insights into the intricate patterns of stock movements. Among the myriad of approaches, the combination of Long Short-Term Memory (LSTM), Extreme Gradient Boosting (XGBoost), and Random Forest has emerged as a compelling strategy for stock prediction.

This project is divided into three parts each focussed on different algorithms. For this analysis, Boeing stock is chosen as the focal point of the stock prediction project. As since the past two decades, the Boeing company has been through a lot of developments and changes due to its reputation as the Aerospace giant. From introduction to the 737-Max aircrafts with highest ever peaks in stock market to the crashes, the Boeing company has been into a steady turmoil recently.

By applying the intricate capabilities of LSTM, XGBoost, and Random Forest, the aim is to unravel the historical patterns and trends specific to Boeing's stock performance. This comprehensive exploration will delve into the dynamics of Boeing's stock movements, leveraging the strengths of each algorithm to provide nuanced insights. Through this focused approach, the endeavour is to enhance the understanding of the factors influencing Boeing's stock prices and contribute to the broader discourse on effective stock prediction methodologies in the financial domain.

Data extraction and loading

The data is originally obtained from https://github.com/yumoxu/stocknet-dataset

It can be obtained from yfinance directly with any specified time frame. Then the data is processed to create extra features like Moving Average, Close Lagged Values and Volatility. These extra features are very necessary as they capture important

aspects of a stock's historical price behaviour, providing valuable information for understanding trends, patterns, and potential future movements.

Plots of the main data

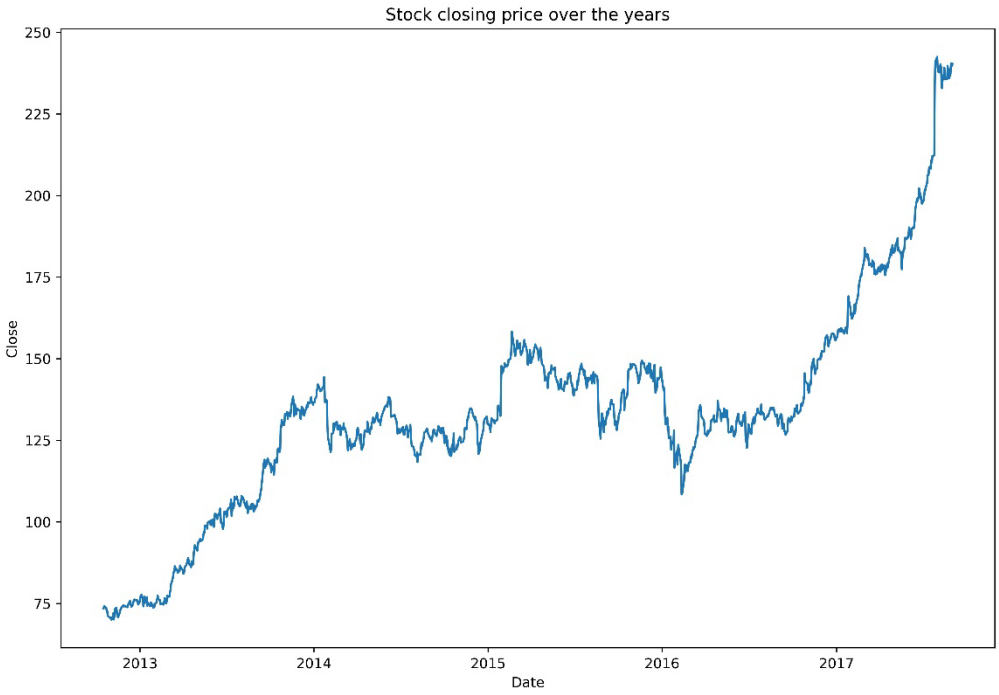

fig 1: Original plot of the data.

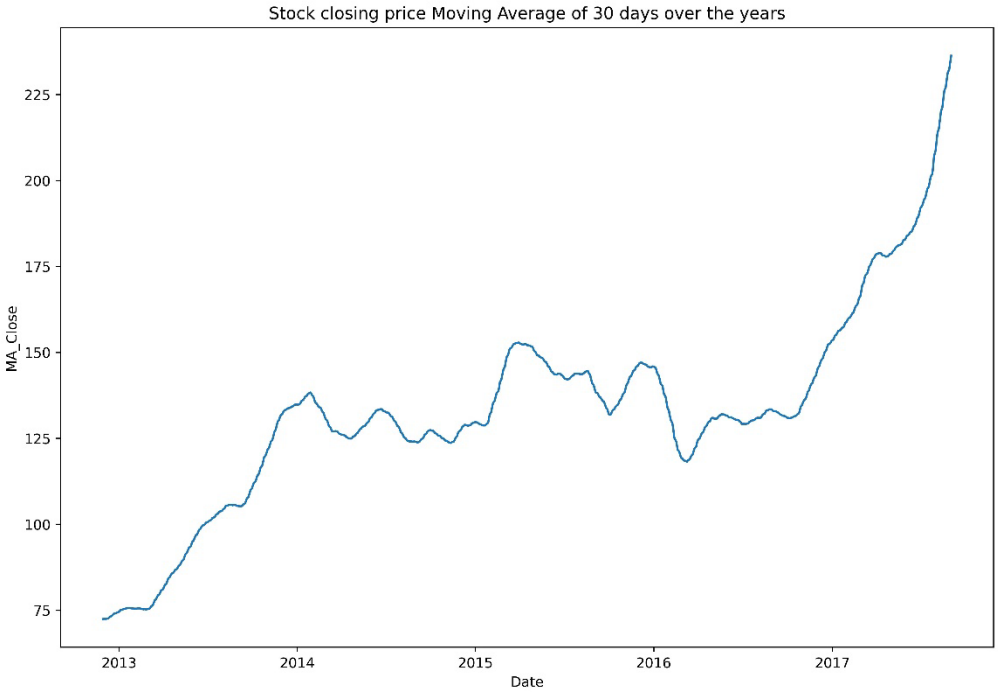

fig 2: 30 days moving average

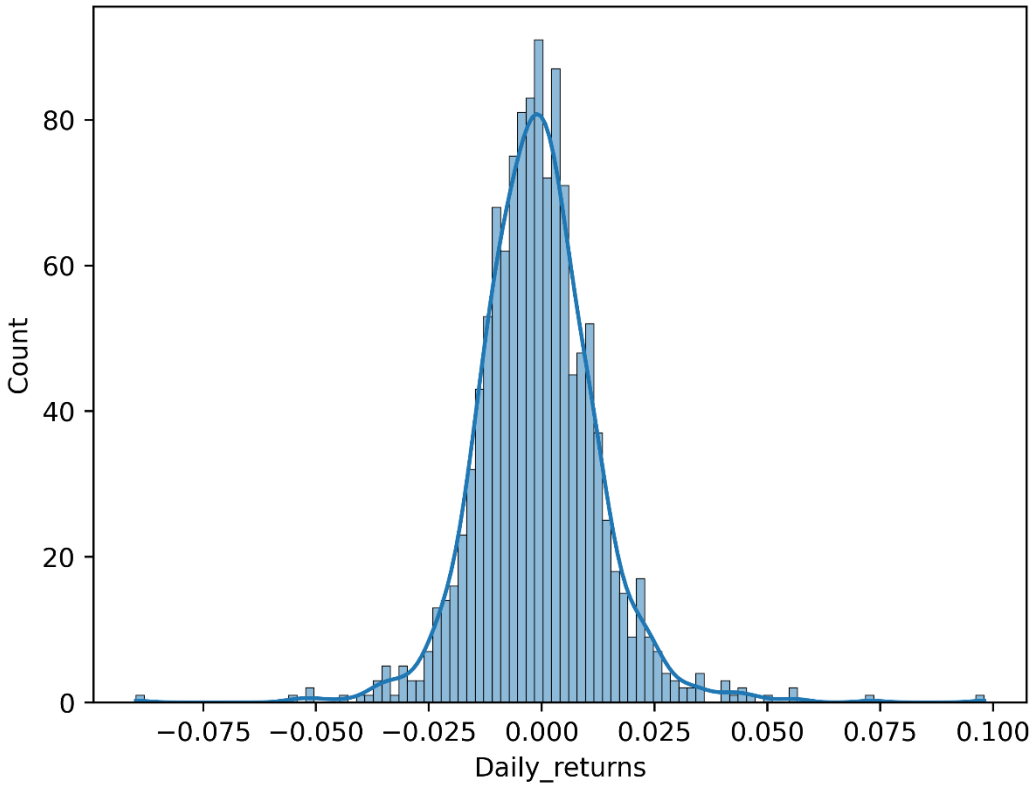

fig 3: Distribution of the data

Data scaling and splitting

The next process in data preparation is the scaling of the data as it is essential, particularly when utilizing models like LSTM. The features such as 'Open,' 'High,' 'Low,' 'Close,' 'Adj Close,' 'MA_7days,' 'Volatility,' and 'Close_lagged_7' often have varying magnitudes. Scaling ensures uniformity, aiding optimization algorithms and preventing any single feature from dominating the model. For LSTM, efficient convergence and improved learning are achieved through standardizing or normalizing input features. Consistent scaling practices across training and testing datasets contribute to the model's ability to generalize effectively and enhance the accuracy of stock price predictions.

For simplicity, the scale is maintained from 0 to 1.

After Data scaling, it is split into train and test data. A very important thing to note here is that the split data is a function of *npast* and *nf* which are the number of days that are used as training set to predict next few day's price. More details will be discussed in data preparation section here.

Algorithms

The regression project is based on three algorithms which are vividly discussed in the following sections.

Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN), which is designed to address the vanishing gradient problem, making it effective for sequential data analysis such as stock price prediction. LSTMs utilize memory cells with self-regulating gates to selectively store and retrieve information over extended time periods. This architecture allows LSTMs to capture long-term dependencies and patterns in time series data, making them well-suited for predicting stock prices based on historical trends. LSTMs excel at learning complex relationships within the sequential nature of stock data, enabling them to capture and leverage essential temporal features for more accurate predictions.

LSTM Time Series data preparation

For the LSTM data preparation, a special approach is applied known as "The sliding window data sequence method". This procedure systematically extracts fixed-length sub-sequences from the time series data. By sliding a window through the temporal sequence, each window corresponds to an input sequence for the LSTM model. The objective is to predict the subsequent value based on the information within the window. This approach allows the LSTM to capture relevant patterns and dependencies in the time series, facilitating the learning of temporal dynamics. Whether the windows overlap or not depends on the specific design choice, and this process is repeated until the end of the time series is reached. For this analysis the previous 20 days are used for 'explaining' next 5 day's price.

The resulting structured data is then reshaped into a 3D array with each dimension consisting number of samples, *npast* and number of features, suitable for LSTM input.

LSTM model development and training

This model development was a repetitive process until output accuracy is optimum. The model that had been implemented and trained, needed to be tuned from single to multiple layers, different activation functions (*relu*, *tanh*, *sigmoid*), numerous combinations of units, regularizations and dropouts. A detailed result and analysis has been provided in this table.

Finally the model is trained in the training data and validated. To make sure that there is lowest chance of overfitting, the method of Early Stopping is used and almost all the models were trained within 100 epochs and stopping the further training in case there is no improvement in validation loss.

The best model that was found out was a three-layered LSTM model with activation function *relu* as the first layer, *tanh* in the second layer and output layer is consisting of the *dense* layer, with 100 and 32 units, 0.03 dropouts and with L2 regularizer value 0.005. With this model, the data is trained and validated with the learning curve below depicting the training and validation loss.

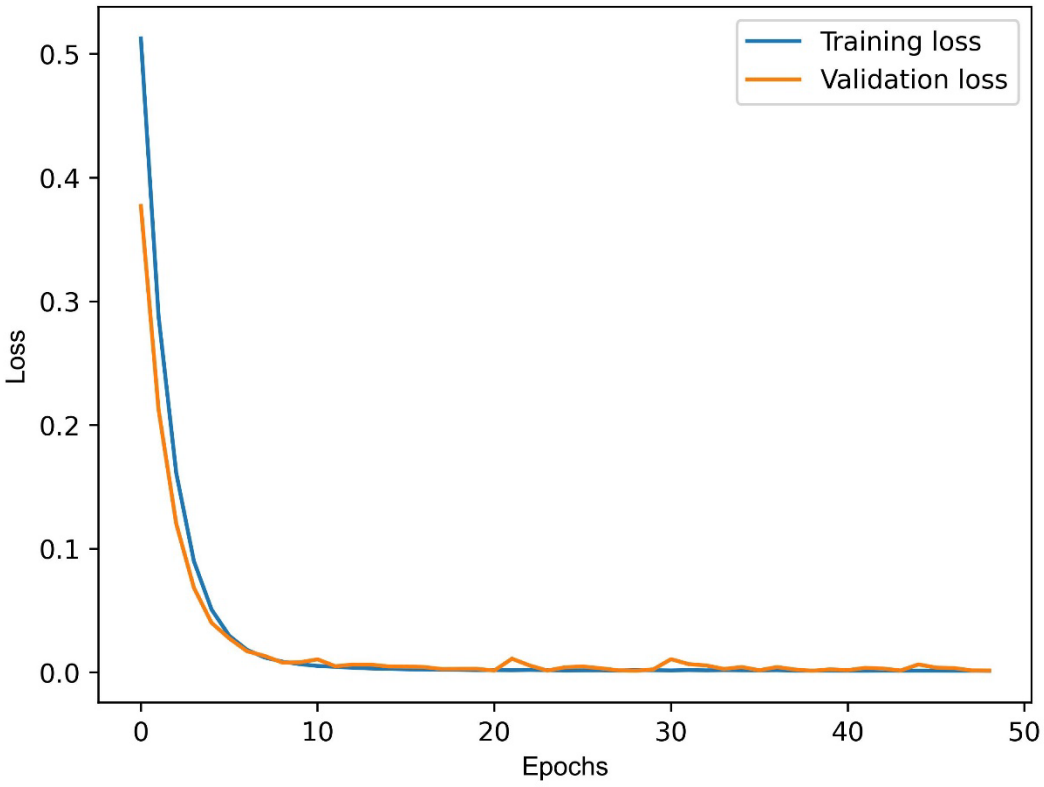

fig 4: Training curve

fig 4 depicts the training curve which is a perfect fit avoiding the overfitting or underfitting the data. As according to Jason BrownLee, "A good fit is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values." Thus the training and validation data is perfectly fitted.

Results and discussion

After training the data to the model, the test data is used to measure the accuracy of the prediction of the model.

Mean Absolute Error: 1.798

Mean Squared Error: 1.170

Root Mean Squared Error: 1.341

MAE value of 1.798 suggests, on average, the model's predictions have an absolute deviation of approximately 1.798 dollars from the actual values. With the MSE of 1.170, the model's predictions, when squared and averaged, result in an overall squared deviation of approximately 1.170 dollars.

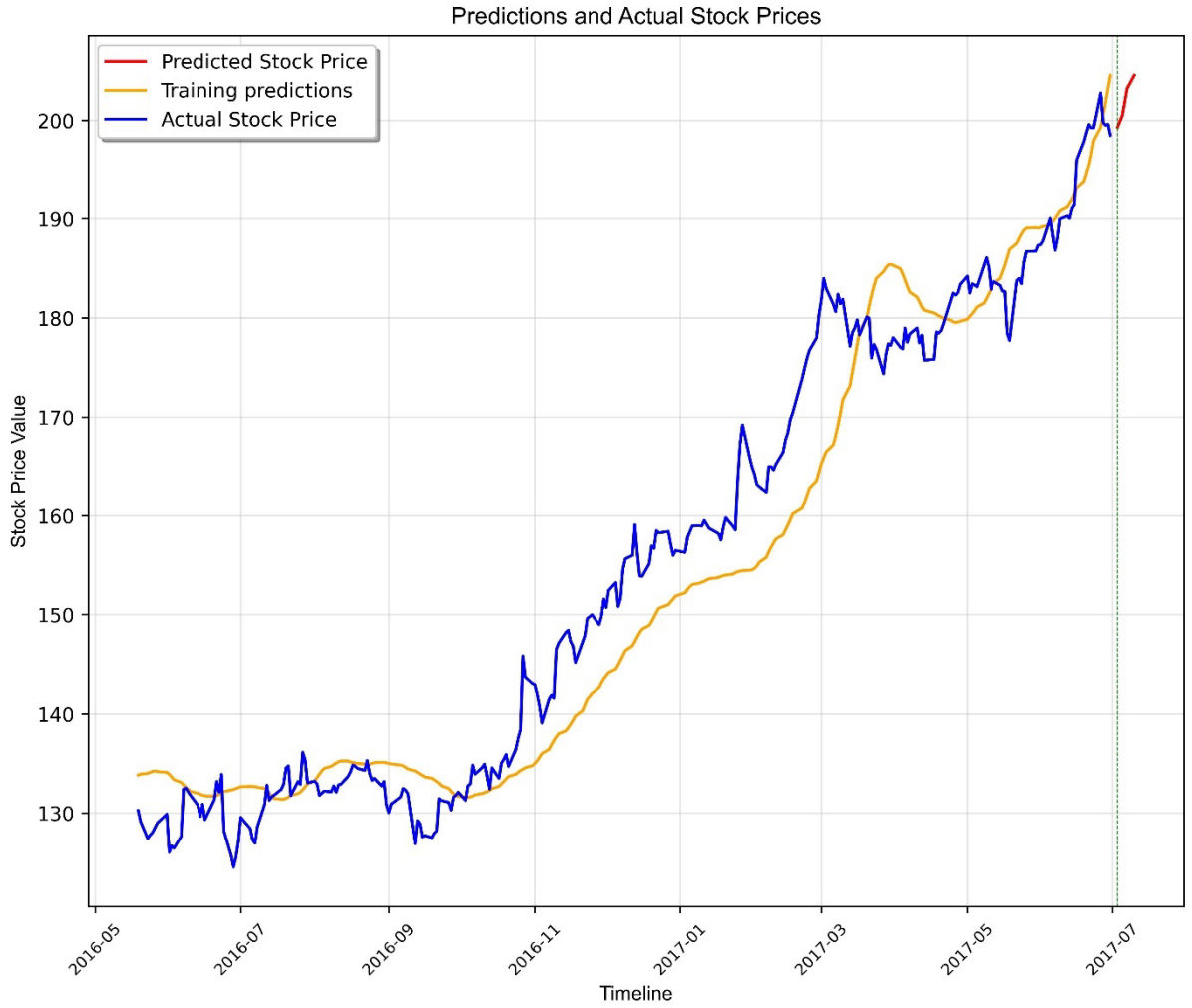

fig 5: LSTM prediction graph

MAE	MSE	Units	Ac.Func.	Layers	Dropout	Epochs
235,6	15.3	100	rd	2	0,03	45
1366.51	36.9	100,32	rrd	3	0.03,0.03	27
1.79	1.17	100,32	rtd	3	0.03,0.03	58
5797.26	76.04	100,32,10	rrrd	4	0.03,0.03,0.25	29
73.35	8.54	100,32,10	rrtd	4	0.03,0.03,0.25	37

Table 1: Different model training and results

Here, the activation function represents the initials of each function, thus forming their combinations.

Extreme Gradient Boost (XG-Boost)

XG-Boost, short for Extreme Gradient Boosting, is a powerful and efficient machine learning algorithm known for its versatility and high performance in various tasks. It belongs to the gradient boosting family of algorithms and employs an ensemble learning approach by combining the predictions of multiple weak learners, typically decision trees. XGBoost enhances traditional gradient boosting by incorporating regularization techniques, handling missing values, and providing efficient tree pruning during the boosting process. Its strengths lie in its ability to handle complex relationships, feature importance analysis, and robustness against overfitting.

Data preparation and model training

Unlike LSTM, XG-Boost requires a 2D array for training. Scaling is not explicitly required but as the scaled data is already prepared and it is very stable, having values closer to unit scales instead of billions (or nanos) is more convenient; numerically the whole system is more stable. Thus for thus model training scaled data is used with 2D data.

GridSearchCV: a technique used for hyperparameter tuning in machine learning. It systematically searches through a predefined hyperparameter grid, evaluating the model's performance for each combination of hyperparameters using cross-validation. For XGBoost hyperparameter tuning, GridSearchCV is used keeping aside the fact that it is very time consuming but this is overshadowed by the accuracy of the proper combination.

```
search = {
'learning_rate': [0,0.1,0.15],
'gamma':[0,0.001,0.01,0.05,0.07,1],
'reg_lambda': [0,0.1,0.2],
'max_depth': [10,30,50],
'subsample': [0.2,0.5,0.7,0.9,0.95]}
```

Within these combinations, the best parameter that was obtained:

```
{'gamma': 0.001,
 'learning_rate': 0.1,
 'max_depth': 10,
 'reg_lambda': 0.1,
 'subsample': 0.2}
```

Results and Discussion

After training the data to the model, the test data is used to measure the accuracy of the prediction of the model.

Mean Absolute Error: 10.826

Mean Squared Error: 3.068

Root Mean Squared Error: 3.290

MAE value of 10.826 suggests, on average, the model's predictions have an absolute deviation of approximately 10.826 dollars from the actual values. With the MSE of 3.068, the model's predictions, when squared and averaged, result in an overall squared deviation of approximately 3.068 dollars.

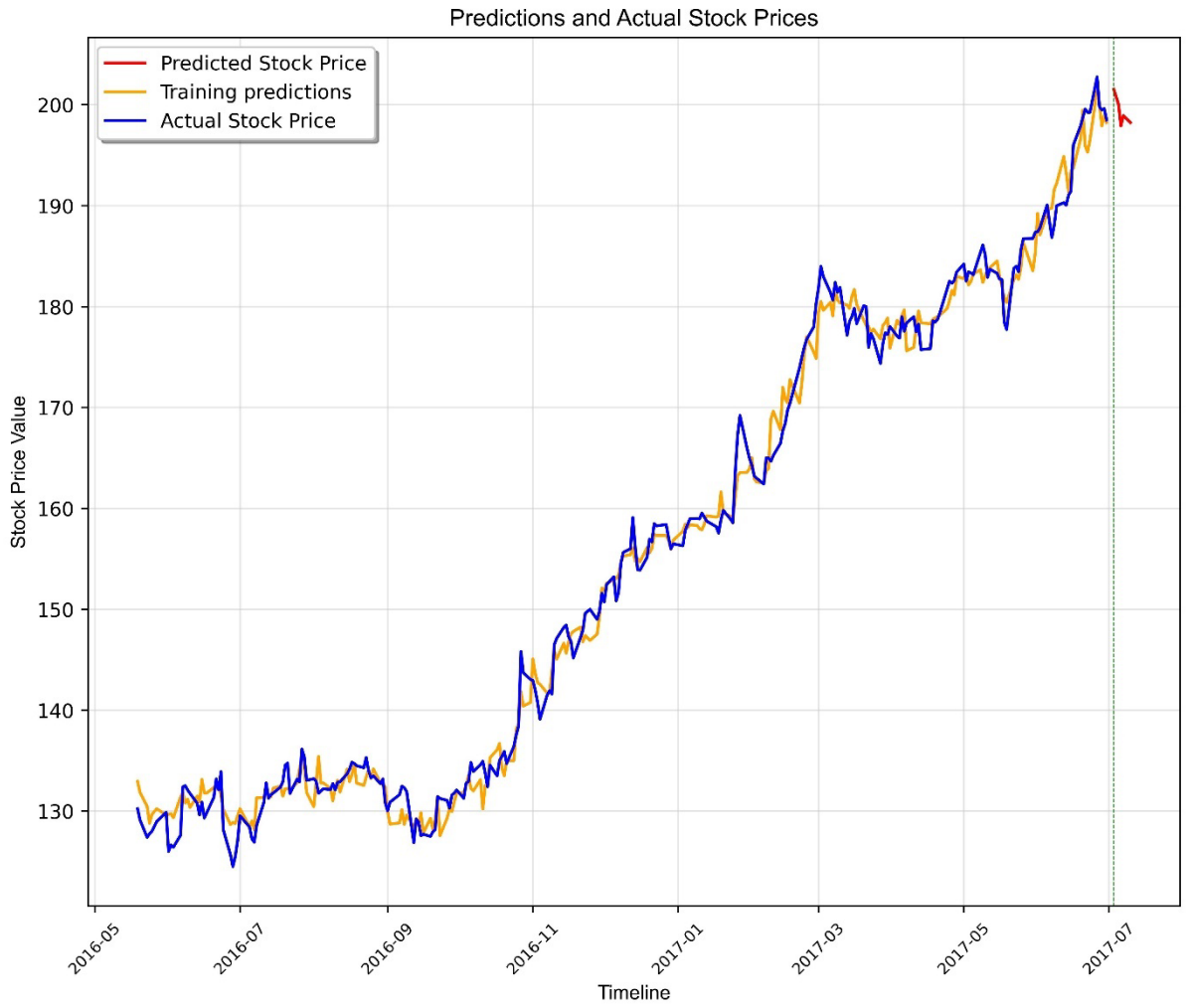

fig 6: XGB Predictions

Random Forest

Random Forest Regression is a versatile machine learning algorithm that constructs an ensemble of decision trees, each trained on a subset of the data and featuring random subsets of features. By aggregating the predictions from multiple trees, it provides a robust and accurate model for regression tasks. It excels in handling complex relationships, automatically handles feature selection, and mitigates overfitting. The algorithm's ability to capture non-linear patterns and its resilience to noisy data make it widely used for predicting continuous outcomes in diverse fields.

Data preparation and model training

Just like XG-Boost, Random forest requires a 2D array for training. Scaling is not explicitly required but as the scaled data is already prepared and it is very stable,

having values closer to unit scales instead of billions (or nanos) is more convenient; numerically the whole system is more stable. Thus for thus model training scaled data is used with 2D data.

GridSearchCV: for Random Forest optimal hyperparameter selection also, the GridSearchCV is used.

```
RFSearch = {
    'n_estimators':[10, 20, 50, 100],
    'max_depth': [None, 10, 50, 100],
    'min_samples_split': [5, 10, 20],
    'min_samples_leaf': [2,5,10],
    'bootstrap': [True, False]
}
```

The best obtained parameters are:

```
{'bootstrap': True,
 'max_depth': 5,
 'min_samples_leaf': 2,
 'min_samples_split': 10,
 'n_estimators': 50}
```

Results and discussion

After training the data to the model, the test data is used to measure the accuracy of the prediction of the model.

Mean Absolute Error: 10.826

Mean Squared Error: 3.068

Root Mean Squared Error: 3.290

MAE value of 10.826 suggests, on average, the model's predictions have an absolute deviation of approximately 10.826 dollars from the actual values. With the MSE of 3.068, the model's predictions, when squared and averaged, result in an overall squared deviation of approximately 3.068 dollars.

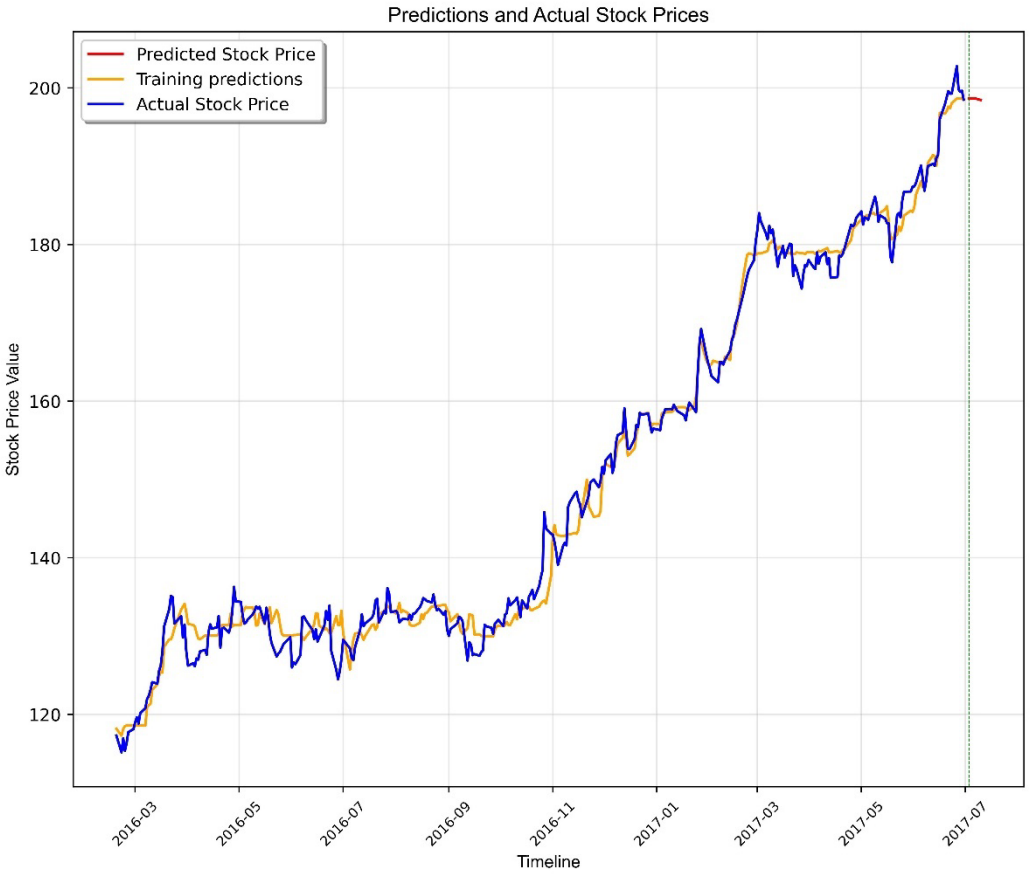

fig 7: Random Forest Predictions

Conclusion

This project is a part of a bigger endeavour which is to predict the stock market trends integrating the Twitter sentiment scores as a feature to the original data. The outcome of this project is a crucial step to identify the optimal parameters for simple time series data prediction using the machine learning. Though as it is seen from the MAE, MSE scores that the LSTM model performs far better than XG-Boost and Random forest, it is very important to note that there can be further enhancement and hyperparameter tuning to enrich the predictions.

Also another important misconception that is cleared from the above work is that different selections of vast numbers of hidden layer does not make the LSTM models more accurate. Thus there are numerous room for errors and opportunities that can be found in this endeavour. Some of them includes automating the hyperparameter combinations (activation functions, layers, dropouts, etc,.) for LSTM models and larger choices for GridSearch, and different ways to scale the features along with creation and integration of new features. Over all, this project can be used as a reference to a larger initiative and learning.

**Parental advisory: Check the memes in the GitHub 😉 and Goodluck!