University of Wisconsin-Madison

ISyE 521: Machine Learning in Action

2021

Prediction of Reliance Industries stock price

Harshit Kothari
Neeraj Deshingkar

December 14, 2021

1 Problem Statement

The aim of our project is to predict the stock price of Reliance Industries, which is a multi national company listed in the Indian stock market. The ability to predict prices can lead to great profits during trading. Hence, the target is the next day's closing price, which is a continuous variable. We plan to use ensemble methods and LSTM to predict the price.

2 Data

We have extracted the data from the Yahoo Finance python library. The data includes the price history and trading volumes of Reliance Industries from 1st Jan 2000 till the most recent day, but we have considered the data till 30th April 2021. We do not have any missing data in our dataset.

- **Date** - Date is given in YYYY - MM - DD format
- **Open and Close** - These columns tell the opening price and closing price of the stock on that day.
- **High and low** - These columns tell us the highest and lowest price the stock reached that day.
- **Volume** - This tells us the total volume of the stock traded that day.

2.1 Additional features created

- A moving average is an indicator that is frequently used for stock price prediction. It is helpful as it helps us to see the trend more clearly, as it smoothens out the noise. Hence, we have added a 7 day (weekly), and 30 day (monthly) MA.
- We extracted the day, month, week, year and day of the week from the Date column.
- Next, we created lag features for Open, High, Close, Low price values. A Lag Feature is basically a variable which contains data from a prior period. A 1 day Lag Feature will contain value of yesterday's stock price. We created a 1 day, 7 day and 30 day lag feature. A 7 day mean lag feature basically contains the mean of the last 7 day's stock price and similarly a 30 day mean.
- We also created NIFTY Lag features with the same lag period mentioned above. We used the rolling window function in pandas to create the lag periods. The essential goal of feature engineering in our problem was to convert time series data into a supervised learning problem so that we can implement the machine learning models.

2.2 Candlestick chart

Figure 1: Reliance Industries candlestick chart for 2021

- We also have a candlestick chart above. It is a concise way of representing all the four prices. (open, close, high, and low). It also has the volume chart at the bottom. The Red color indicates that the price of the stock decreased that day, and green indicates that the price increased.
- In the above figure, the blue line denotes the weekly MA, and the orange line denotes the monthly MA. Note that the monthly MA is not available for first few datapoints, as we need at least 30 for calculating the MA. Similarly for the weekly MA.

3 Why is time series different?

- Usually we are not interested in predicting something which has already happened. Hence, time series is inherently an extrapolation problem. As a result, the techniques which are used are very different from normal interpolation methods.
- For interpolation problems, the error which we get while prediction remains relatively constant as we change the test data. But for extrapolation problems, as we keep going in the future, predictions keep getting worse. This can be understood intuitively as errors keep accumalating as we try to predict ahead into the future. Hence for this project, we limit ourselves to just predict the closing price of the next day.

4 Methods

4.1 Stationarity

- A stationary time series is one which is a flat looking series. It has a relatively constant mean and variance over time, and does not have any seasonality (periodicity). We are concerned about stationarity as many statistical methods for prediction have been developed assuming stationarity. AR models are an example.
- Clearly from the plot below, we can see that the Reliance closing price data is not stationary, as the mean is increasing over time. (The orange trend in the plot)
- The Augmented Dicky Fuller (ADF) test is used to check if a time series is stationary. We get a p value of 0.362. Hence, we conclude that the time series is not stationary.
- One way to convert the time series to a stationary one, is taking the closing price differences between consecutive days. (The blue trend in the plot). If we do this, we get the p value to be 0.0. Hence, we reject the null hypothesis that the time series is non-stationary.

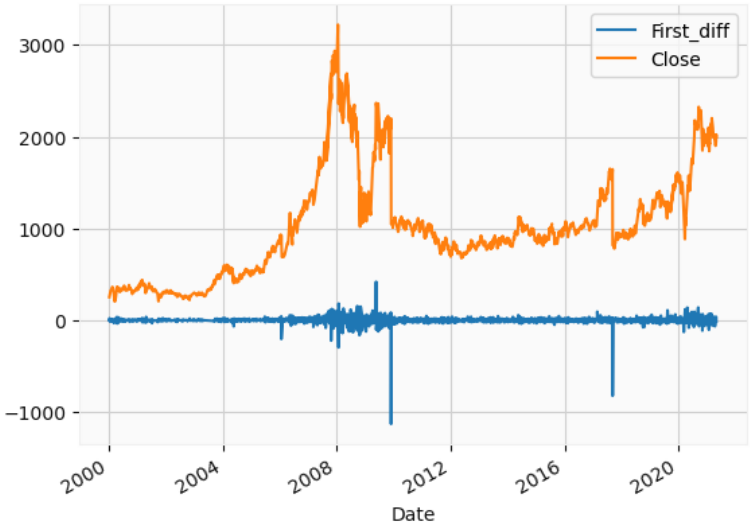
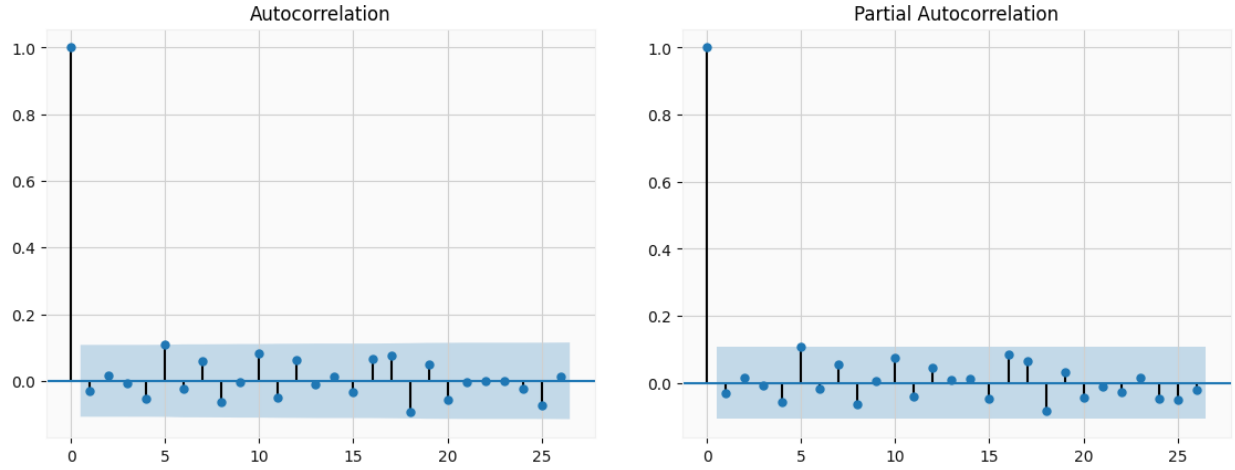


Figure 2: Stationary

4.2 Autocorrelation (ACF) and Partial Autocorrelation (PACF)

- **Autocorrelation** - It is the correlation between the closing price values of two different days.
- **Partial Autocorrelation -** This is also similar to ACF, but it differs as the correlation for each lag is the unique correlation between those two observations.
- Looking at the plot below, we note that both PACF and ACF don't help us as the points inside the blue region in the plot are statistically insignificant. This just reinforces that predicting the stock prices is a difficult problem. If we would have seen a clear correlation with some other lag, then we could have predicted the future stock price quite easily. But this is not the case.



5 Cross Validation

Usually in ML models, for grid search, we create random splits in the data for training and testing. But this can't be done for time series data as we can't train the model on future data (which might get included in the fold). Hence, here we do cross validation on a rolling basis. We start with a small subset of data, and predict ahead. These forecasted points are then included in the training dataset and we predict the subsequent data points. As we can see in the figure below, the blue region is used for training and the red data points are used for testing. Also, in the later iterations of cross validation, we have more data to train on as more time has passed.
To implement this we have used the TimeSeriesSplit function in sklearn.

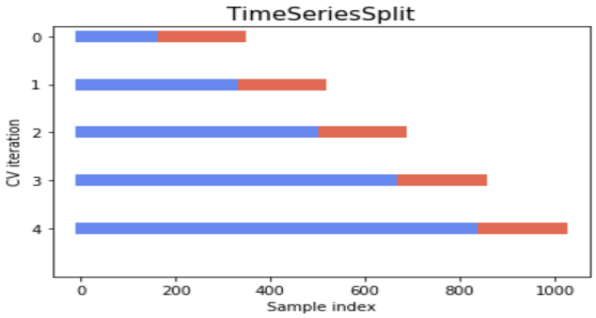


Figure 3: Cross Validation in Time series

6 Models Implemented

Stock price prediction is a tough time series problem and we tried to implement the most common and powerful methods which are used for time series analysis. Usually XGBoost and LSTM are the most preferred models if we consider supervised learning methods. More complicated models involve

reinforcement learning, which we have not explored in this project. We are using ARIMA as a baseline model, and have compared the ensemble methods and LSTM with it. We have included all the plots of the prediction we have made in the Results section.

6.1 Auto Regressive Integrated Moving Average (ARIMA)

ARIMA is a model used for time series analysis and forecasting. The model is used on time series data which will be transformed into a stationary time series; the predictions are a linear regression upon features including time differences (lags) and moving averages. We have used the implementation from the Statsmodel library to fit the model. In ARIMA, the data is differenced; that is, the price features are transformed to the difference between prices. Let L be the lag operator, then the ARIMA equations are

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right) (1 - L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

and p, d, q are hyper-parameters over which we optimized. Hence, we predict the price at day t, by fitting the model with the price history till day $t - 1$.

6.2 Random Forest

Random Forests is a ensemble learning method that is used for regression and classsfication problems, that operates by constructing a multitude of decision trees at training time. This Learning method is an excellent solution for reducing variance whilst keeping bias constant. Our approach was to see if random forest can give better accuracy than our baseline model. Random forest can also handle large datasets with higher dimensionality and can identify important variables that are used for prediction. The computational cost of training a random forest is not very high and it is quite robust to outliers. Our goal was to utilize these advantages of the model in our time series problem to give us an accurate forecast at low cost. We used the RandomForestRegressor in sklearn to implement this model. We ran GridSearch CV with 5 folds of cross validation using timeseriessplit and searched the best parameters for number of trees , maximum features and maximum depth of the decision tree. Our Scoring metric was root mean squared error. The optimal parameters we got were 1500 trees, max depth of 7 and maximum features as 'Auto'.

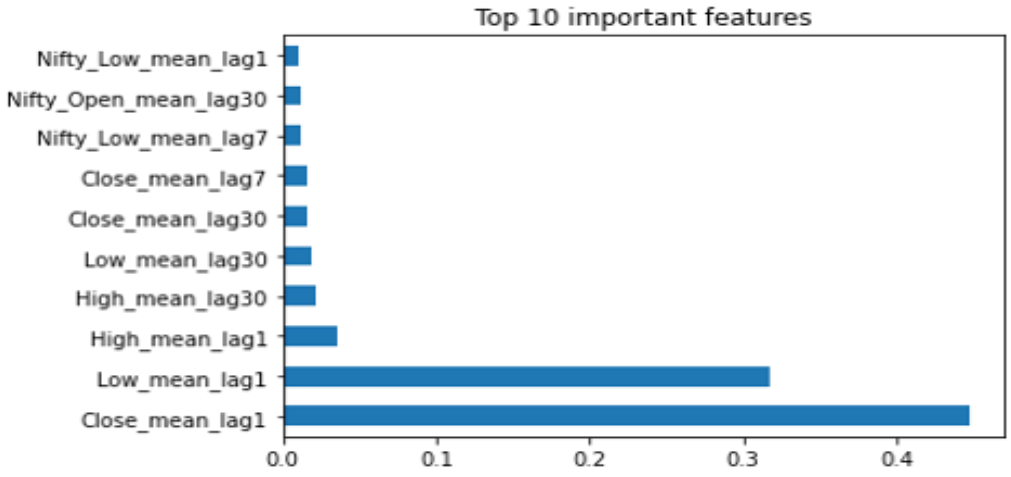


Figure 4: Random Forest Feature Importance

After implementing the Random forest model, we found out that the close price value on the previous day was the most important feature and had an importance of around 0.45. This makes sense intuitively. We see that NIFTY features also hold some importance. It is a bit suprising that date time features like week, month, day of week did not hold much importance in the model.

6.3 XGBoost

XGBoost, short for Xtreme Gradient Boosting is a regularizing gradient boosting framework. Some advantages of XGBoost over other boosting algorithms are the clever penalization of trees, automatic feature selection and extra randomization parameter. We have used the XGBoost library to implement this model. We ran GridSearch CV with 10 folds of cross validation using TimeSeriesSplit and searched the best parameters for number of trees, subsample size, maximum depth, learning rate and the gamma value. Some of the optimal parameters we got after cross validation were 70 trees, learning rate of 0.1 and maximum depth of 3.

6.4 LightGBM

LightGBM, short for Light Gradient Boosting Machine, is a gradient boosting framework. It's known for its extremely fast computational capacity and minimum memory requirement. LightGBM is similar to XGBoost but not completely alike, it differs in the construction of trees. LightGBM does not grow a tree level-wise, instead it grows trees leaf-wise. It chooses the leaf it believes will yield the largest decrease in loss. We have used the LightGBM package for model fitting as this package was not available in sklearn. In practice, this model takes 0.023 seconds to fit, whereas the XGboost model takes 0.137 seconds to train. Hence, LightGBM is a better model compared to XGBoost for our application as we got similar accuracy from the former in lesser time.

6.5 LSTM

Long Short Term Memory (LSTM) is a kind of Recurrent Neural Network (RNN) with the capability of remembering the values from earlier stages for the purpose of future use. This feature makes it naturally useful for time series data. Three types of gates (forget, memory and output gate) are involved in each LSTM with the goal of controlling the state of each cell.

6.5.1 Data Preparation

LSTMs are known to be sensitive to the scaling of the data. So, we used a MinMaxScaler to scale all the values between 0 and 1. Also, we create arrays of 30 data points each, and train the model to predict the next days closing price.

6.5.2 Regularization

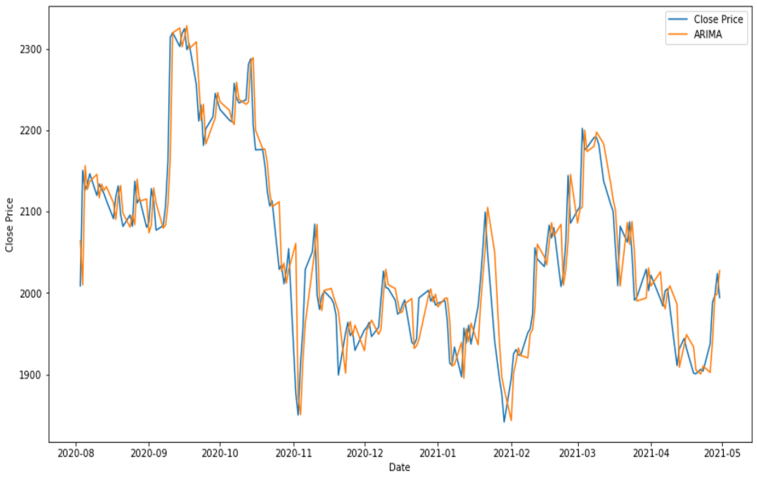
We know that neural networks have a very high tendency to overfit the data. Hence, we studied two techniques which help prevent overfitting.

- **Dropout** - This technique tends to drop a few units randomly in the neural network, which results in making the units lesser dependent on the inputs. It has been shown to be loosely similar to L2 regularization.
- **Early stopping** - In this, we tend to stop the training when the validation loss starts to increase, which is an indication of overfitting.

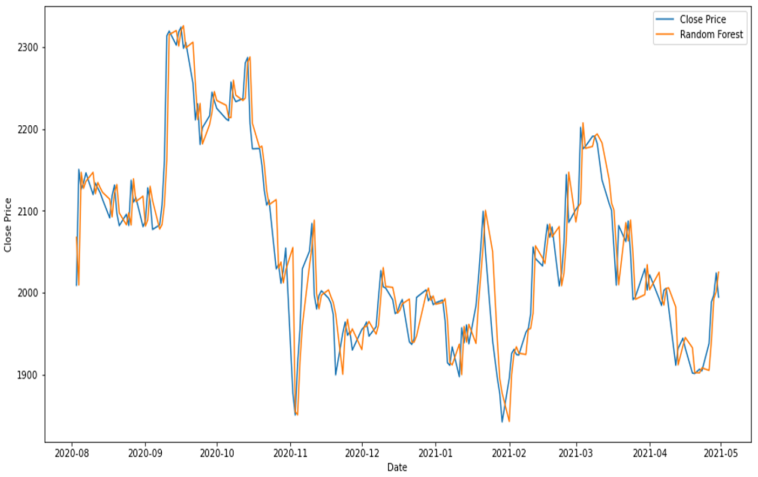
6.6 Training details of LSTM

We used keras and tensorflow to create the LSTM and dropout layers. We have used 5 LSTM layers and 1 dense layer. We have used 20% dropout for the 1st LSTM layer, and 50% dropout layer for the rest. We have used the ADAM optimizer, and mean squared error as our loss function. As we can see from the LSTM plot below, the predicted trend seems to be quite close to the actual trend. But the predicted trend (red) always seems to lag the actual curve (green). This implies that the model is not very smart and tries to *copy* what was the closing price of the previous day. Looking at the plot, we realized that training a Neural Network can be extremely difficult!

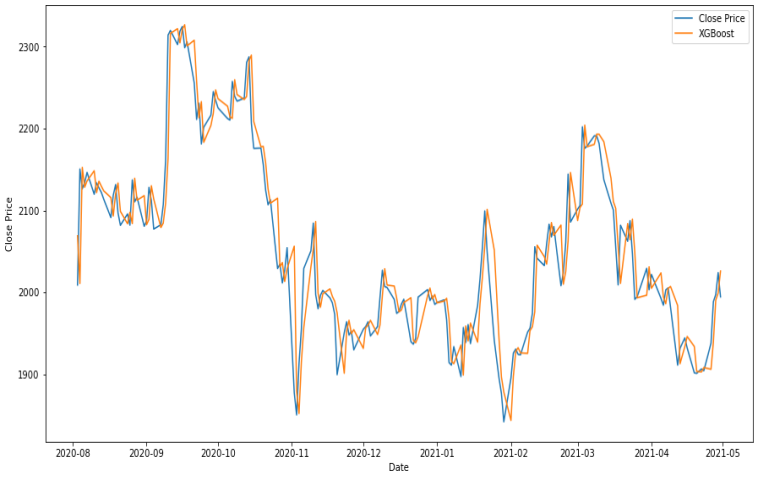
7 Results



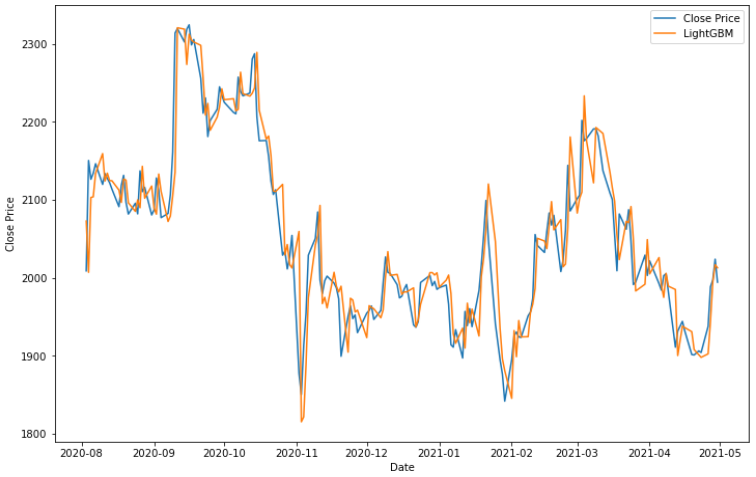
ARIMA



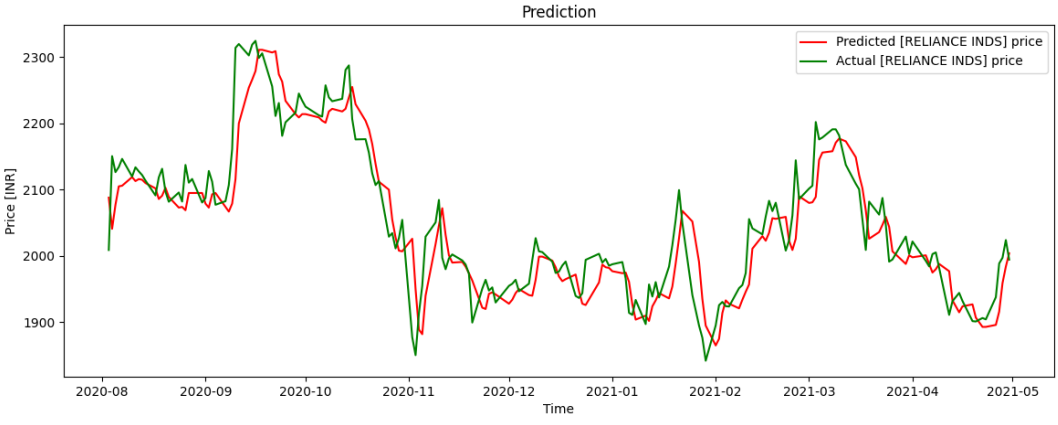
Random Forest



XGBoost



LightGBM



LSTM

7.1 Metrics used to evaluate the models

We have primarily used two metrics -

- **RMSE** - We have used the most natural metric for regression problems

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}$$

- **SignChange** - We have defined a new metric called sign change. This is basically used to evaluate if the model is predicting the correct direction of price movement for the next day. So, for example, if the model predicts that the price should go up, but it goes down instead, then this metric increases by 1. Ultimately, the metric tells us how many mistakes the model has made every 100 predictions.

7.2 Results with and without NIFTY

The NIFTY 50 is an Indian stock market index. It is similar to the S&P 500 in the American stock market. The value of the index is calculated by taking a weighted average of the 50 most traded stocks. Reliance Industries has the highest weightage in the NIFTY 50.

Models	**RMSE with NIFTY**	**Previous RMSE**
XGBoost	38.83	39
Random forest	**38.57**	**38.71**
LightGBM	39.04	40.8

To give the model some more sense of what is happening in the economy, we included NIFTY values of the previous day as a feature. As we can see in this table, that the accuracy of the models improve when given this additional information, but not by a significant amount.

7.3 Final results

Model	**RMSE**	**Sign Change**
ARIMA	39.04	0.559
XGBoost	38.83	0.483
Random Forest	**38.57**	0.489
LightGBM	39.04	**0.456**
LSTM	46.47	0.539

The above table includes our final results. As we can see that the Random Forest performs the best, with respect to the RMSE scores. But if we look at the sign change metric, we realize that none of the machine learning methods is doing well. Typically the models are predicting the stock movement in the wrong direction roughly 40% of the times. Hence, these models can't be used for trading.

8 Conclusion

We conclude that the random forest model has performed the best among all the models we tested. Surprisingly, LSTM performs the worst. But, these models can not be trusted in their current state as we can see from the SignChange metric. We realize that predicting the stock market returns is a challenging problem as the values are continuously changing due to several factors.

9 Future Directions

- Sentiment is also very important in predicting stock prices. There might be extraneous factors like a pandemic or the management of the company changes. We can incorporate this by creating a model which keeps track of the news (NLP model).
- For a similar reason as above, machine learning methods might be better applied to cryptocurrencies like Bitcoin or Ethereum, as they don't have any underlying asset. Hence, the ML models with some sense of the news, might be able to capture most of the trend.

10 Acknowledgement

We would like to thank Prof. Boutilier and the TA Ari Smith for their guidance and constructive feedback with our reports and proposals. We also found some blogs and github repositories helpful with coding and understanding the concepts of time series. We have compiled a list of those here.

11 Appendix

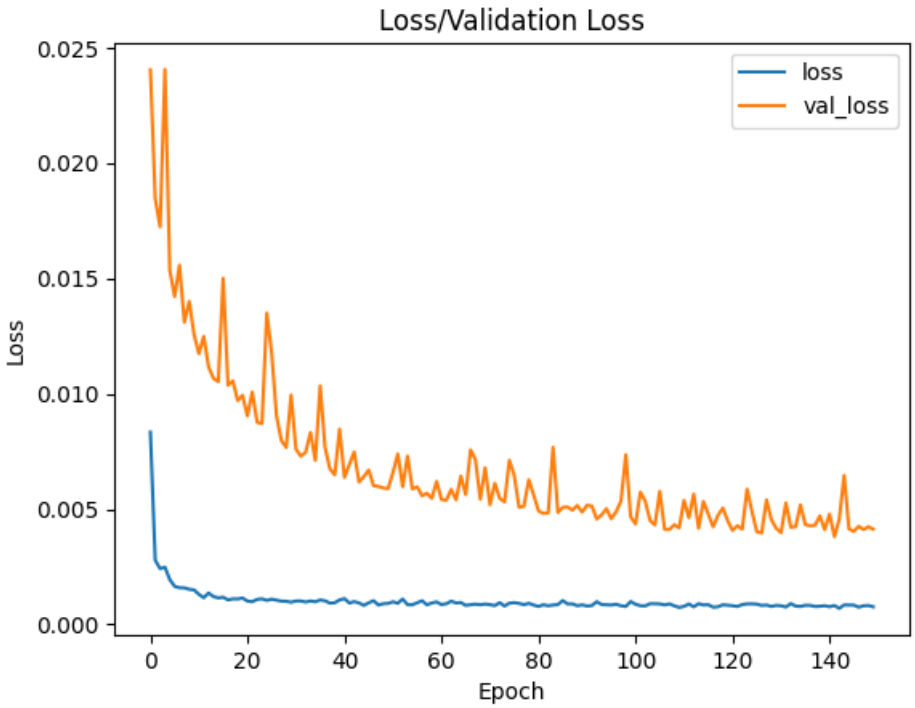


Figure 6: Validation Loss and Training loss for the LSTM

- We have stopped the training the LSTM roughly when the validation loss almost became constant.