Forecasting Stock Price Using Sentiment Analysis and LSTM Networks

Blake Hillier, Grace Li, Joe Puhalla

May 11, 2020

1 Overview

Forecasting stock prices is a widely known problem many people have attempted to solve through various models. A common approach is to use only numerical data to predict tomorrow's price. However, investors often use the news about the company's decisions, the public's attitude towards the company, and the state of the economy when developing trading strategies. In this paper, we propose a model using macro-economic variables to predict the future price of a stock, one of which is statements from the Federal Reserve about decisions on economic policies. Our model is comprised of XLNet to perform sentiment analysis on one macro-economic variable and an LSTM Neural Network to combine all the variables while capturing the effect time has on the future stock price. In this paper we first explain the ideas behind XLNet and LSTM Neural Networks. We then explain our implementation of XLNet and LSTM and calculate their mean squared error (MSE) using appropriate subsets of our data. Finally, we explain how we use them together to forecast the stock price and calculate the entire models MSE before making some concluding remarks.

2 XLNet

XLNet is an autoregressive pretraining approach for NLP models. Pretraining a model is used to teach a model a broad field before being applied to a specific problem. This allows it to pick up potential nuances within the data and then apply this knowledge to better understand the specific problem. It also allows copies of the same model to be fine tuned to different problems, decreasing the total learning time. Autoregressive pretraining approaches create a conditional probability distribution based on the likelyhood function

$$p(x) = \prod_{t=1}^{T} p(x_t | x_{<t})$$

which only sees the relationship between previous text (it can also be modified to only see the relationship between text after the word). This is problematic since most words derive their meaning from context at both the beginning and the end of a sentence. XLNet solves this problem by calculating a distribution based on all the other text in the word by maximizing

$$\max_{\theta} E_{z \sim Z_T} \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} | x_{z<t}) \right] = E_{z \sim Z_T} \left[\sum_{t=1}^{T} \log \frac{e^{g_\theta(x_{z<t,z_t}) l(x_t)})}{\sum_{x'} e^{g_\theta(x_{z<t,z_t}) l(x')}} \right]$$

where Z_T is the set of all permutations of text of length T, $z \in Z_T, x_{z<t}$ is the sequence of text from 1 to $t-1$, and g_θ transforms x to a sequence of hidden words with the first $t-1$ set of words as additional information. They note the permutations don't affect the actual order of the text sequence, just the factorization of the likelihood function. Because this is based on the likelihood function, it removes the idea each hidden text is independent of the others while still maintaining the benefits through encoding due to g_θ. In order for g_θ to accomplish this, they split it into two different transforms: the query g_θ which looks at the first $t-1$ words in the permuted order to predict the t^{th} word, and the content h_θ which simply encodes the first t words in the permuted order. The one downside to this model is the complexity of the optimization leading to a slower convergence. To reduce this problem, they adjust the equation to

$$\max_{\theta} E_{z \sim Z_T} \left[\log_{p_\theta}(x_{z>c} | x_{z \leq t}) \right] = E_{z \sim Z_T} \left[\sum_{t=c+1}^{|z|} \log p_\theta(x_{z_t} | x_{z<t}) \right]$$

which changes the model to only predict the last $T-c$ words in the permutation order, where $c \approx \frac{T(1+K)}{K}$ for some parameter K.

3 LSTM

In this paper, the LSTM neural network is uesed to predict the stock price, the input data is the historical stock price and sentiment analysis results. Here, the sentiment based LSTM neural network (named sentiment-LSTM) is aimed to minimize the following loss function:

$$\ell = min \sum_{t=p+1}^{p+T} \left\| X_t - \hat{X}_t \right\|_2^2$$

Where T denotes the number of prediction time slots, i.e, $t = 1, ..., p$ are the observations (training input data), $t = p+1, ..., p+T$ are the predicts (training output data); and $\hat{X}_t$ is given as following:

$$\hat{X}_t = \alpha X_t^A + \lambda S_t^A + c = \alpha X_t^A + \lambda f_2 \underbrace{((S_{t-i})_{i=1}^{p})}_{\text{Sentiment}} + c$$

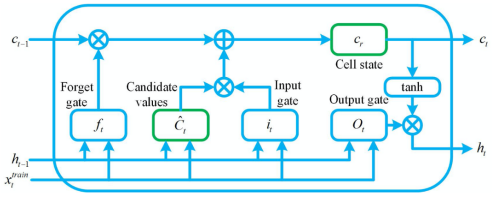

Figure 1: LSTM Procedure

Denote Formula as the training input data. Figure 1 shows the LSTM's structure network, which comprises one or more hidden layers, an output layer and an input layer. LSTM networks' main advantage is that the hidden layer comprises memory cells. Each memory cell recurrently has a core self-connected linear unit called " Constant Error Carousel, which provides short-term memory storage and has three gates:

Input gate, which controls the information from a new input to the memory cell, is given by

$$i_t = \sigma(W_i \times [h_{t-1}, \chi_t^{train}] + b_i)$$
$$\hat{c}_t = tanh(W_c \times [h_{t-1}, \chi_t^{train}] + b_c)$$

h_{t-1} is the hidden state at the time step $t-1$; i_t is the output of the input gate layer at the time step t; $\hat{c}_t$ is the candidate value to be added to the output at the time step t; b_i and b_c are biases of the input gate layer and the candidate value computation, respectively; W_i and W_c are weights of the input gate and the candidate value computation respectively; and $\sigma(x) = 1/(1+e^{-x})$ is the pointwise nonlinear activation function.

Forget gate, which controls the limit up to which a value is saved in the memory, is given by

$$f(t) = \sigma(W_f \times [h_{t-1}, \chi_t^{train}] + b_f)$$

where f_t is the forget state at the time step t, W_f is the weight of the forget gate; and b_f is the bias of the forget gate.

Output gate, which controls the information output from the memory cell, is given by

$$c_t = f_t \times c_{t-1} + i_t \times \hat{c}_t$$
$$o_t = \sigma(W_o \times [h_{t-1}, \chi_t^{train}] + b_o)$$
$$h_t = o_t \times tanh(c_t)$$

where new cell state c_t are calculated based on the results of the previous two steps; o_t is the output at the time step t; W_o is the weight of the output gate; and b_o is the bias of the out put gate.

4 Experiment

Because our model has multiple core algorithms, we need to calibrate each main component individually as well as when they are all together to ensure we are obtaining an optimal accuracy. This section will discuss the data we use, how each algorithm performs on their own, and how they perform when put together.

4.1 Data

Our data consisted of macroeconomic data and the FOMC Federal Statement text data. The macroeconomic data consisted of the quarterly GDP, monthly CPI, monthly unemployment rate, monthly inflation rate, the ten-year treasury rate, the 12-month LIBOR, and the FOMC statements from the Federal Reserve. All the data ranges from the beginning of the year 1994 to the end of 2019. We used Marathon Oil Corporation (MRO), Fifth Third Bancorp (FITB) and Devon Energy Corp (DVN) as stock data for algorithm training. Since our stock is daily data, all previous data were duplicated into daily data and aligned with the stock information.

4.2 Sentiment Analysis

We used pytorch's implementation of XLNet-base for our model. Since we were analyzing statements from the fed, we felt the sentiment should come from a macro variable. If the variable increased to the next time-step, the sentiment was positive, and negative otherwise. Since we weren't sure which feature would work best with the stock, we decided to test GDP, Libor, Unemployment rates, and Inflation rates (MICH). We also tested a variety of batch sizes, maximum string lengths, and epochs to try and maximize the accuracy. Figure 2 shows both the accuracy and the variance of the tests. Some results to note:

Feature	String Length	Batch Size	Epochs	Accuracy
GDP	256	8	10	62%
MICH	64	48	10	51%
UNEMP	128	24	10	56%
Libor	128	24	10	67%

These parameter sets will be used to configure the final model when deciding which sentiment metric best helps forecast stock prices.

4.3 LSTM Stock Prediction

The LSTM network we use includes four layers each with a dropout rate of twenty percent. The layers are of size $256, 128, 64, 32$ and then the final layer. We sampled through several activation functions, starting with linear activation, and determined the ReLu activation to provide the best accuracy of the model. In order to determine the accuracy of the model, we plot both the predicted and actual stock price values. We then determine the mean squared error between the predicted and actual prices. We sample across several amounts of epochs and batch sizes in order to determine the best architecture for the model. For simplicity purposes, we used the already given 'adam' optimizer. With a batch size of 8, the 20 and 50 epoch sized networks caused memory issues and thus were unable to be completed. A batch size of 64, and 50 epochs yielded the lowest MSE among all models (3). Our feature set for the model were daily measures of price, GDP, CPI, inflation, unemployment, LIBOR, and the 10-year treasury yield. GDP data was given quarterly and thus had to be converted to daily data. In addition, CPI, inflation, and unemployment were given as monthly data and were converted to daily data. Our training data was from 1994 up to 2016 and our testing data was 2016 to 2019. From out results for each stock we calculated the average mean squared error. The average mean squared error was 2559.10. Looking at each stock prediction, we observe that the LSTM model predicts some stocks very well, but also very poorly predicts a large amount of stocks. A selected group is shown in the appendix: FITB, DVN, and MRO. There is a very high variance in the prediction power of our model. We also look at the distribution of the MSE scores for each stock in order to achieve a better understanding of the prediction power of our model. Looking at the distribution below for MSE values less than 500, we see that most of the MSE values are in the range of 0 to 150. More tests need to be done to determine why some stocks are predicted incredibly well while others do so poorly so that we can reduce the variance of our model in the future. Increasing the the number of epochs might be able to reduce this variance at the expense of increasing computational time. Furthermore, the number of layers needs to be varied as well as the optimizer used. Tests need to be done to determine which features contribute the most to the predicting power of our model. Sentiment scores of the FOMC transcripts also need to be added as features of the model.

4.4 XLNet and LSTM

After finding the most accurate sentiment prediction parameters and the best LSTM configuration, we tested both models working together. We first trained the XLNet on our calculated sentiment from a chosen macro feature. We then transformed all of the non-daily features into daily features, and began training the LSTM using our sentiment, macro features, and stock price for each stock. Lastly, we tested the model on the last four years of data to calculate it's MSE. We ran this experiment for each macro feature discussed in section 4.2 to see which feature helped create the most

accurate predictions. Our results are below:

Feature	Avg MSE	Variance
GDP	23704.33	64384386348.59
MICH	35575.65	198994904967.61
UNEMP	39754.39	198372093847.72
Libor	30592.24	125957788602.12

We calculated the average MSE from each stock to judge the effectiveness of the macro feature, as well as the variance. Comparing this list to the one in 4.2, we see generally the more accurate XLNet is with a given macro feature, the better the combined model performs. Based on these tests, using GDP rates seems to be the best metric. Unfortunately, using XLNet did not improve the accuracy of the LSTM model. We can see some of the results from FITB, DVN, and MRO below.

5 Conclusion

In this paper, we attempt to build a stock forecasting model using sentiment analysis with XLNet and then forecast using an LSTM network. Firstly, we used text data from Federal Reserve Statement because we considered the economic strategy Fed made can reflect the market's yearly performance. Secondly, we combined the XLNet sentiment analysis results with the LSTM to identify and extract opinions from the federal statements, combining the stock adjust the close price and macro-economic data to make stock prediction more accurate. However, the accuracy performance of XLNet-LSTM ends up with an MAE which is twice as much as the LSTM MAE. That means that XLNet can not make the prediction result more accurate and robust. Then we also used MSE as a loss function to further evaluate each feature. Among all the features we selected, LIBOR has the smallest MSE which means the LIBOR values dispersed closely to the central moment. Lastly, while judging the MSE of the predicted value is a great way of understanding the model's accuracy, simulating a trading strategy using the model would allow us to more concretely understand how usable our model really is.

While this project's results are disapointing, there are still things we can learn from this project. Looking back on this project, we think a few changes to our overall process would have resulted with a much better model. One is how we first approached this complex model. We should have spent more time configuring the LSTM before we began working on the XLNet in order to better optimize the predictive power of the LSTM model. Another problem is in how we combined the XLNet and the LSTM models. These should have been wired together to allow the error in the forecasts propagate back to the weights in XLNet. This would have also required us to create a unique loss function to evaluate the power of the sentiment from the XLNet as well. If this project was to be redone, these steps would either create a much more accurate model, or more firmly show this models inability to properly forecast a stocks price. For a full look at the code, check out our github.

References

[1] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregresive Pretraining for Language Understanding*. Carnegie Mellon University, 2Google AI Brain Team, Jan 2^{nd} 2020.

[2] Xinyi Li, Yinchuan Li, Hongyang Yang, Liuqing Yang, Xiao-Yang Liu. *DP-LSTM: Differential Privacy-inspired LSTM for Stock Prediction Using Financial News*. Columbia University, Beijing Institute of Technology.

[3] Github LSTM Stock Prediction, https://github.com/laxmimerit/Google-Stock-Price-Prediction-Using-RNN---LSTM/blob/master/Google%20Stock%20Price%20Prediction%20Using%20RNN-%20LSTM.ipynb

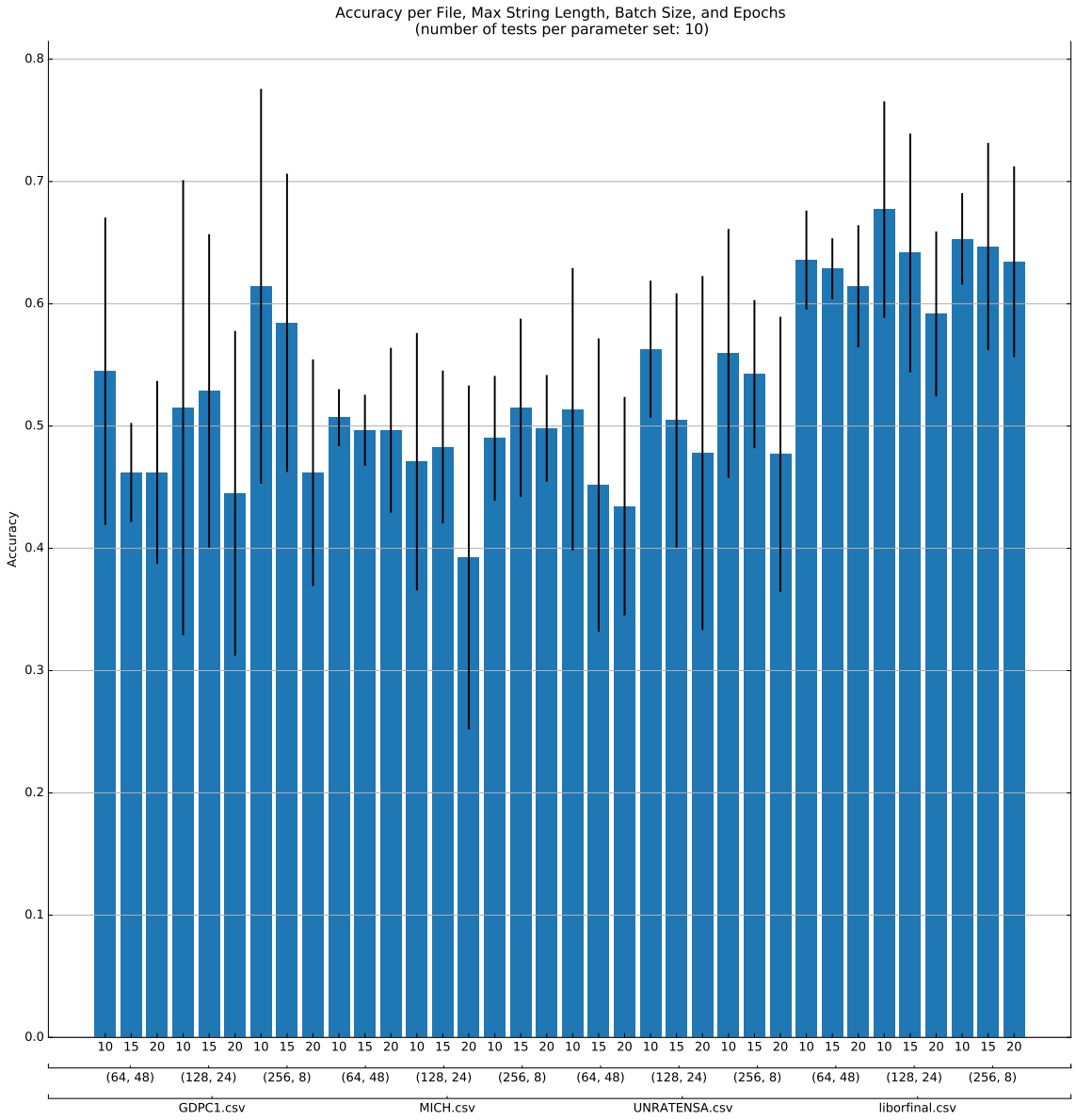

Figure 2: Simulation results for XLNet parameter tests

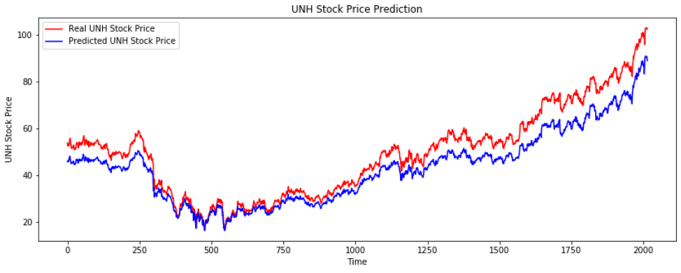

Figure 3: Predicted vs. Actual Stock Price

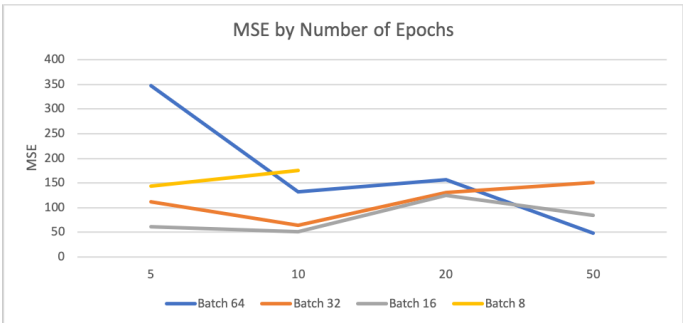

Figure 4: MSE by Number of Epochs

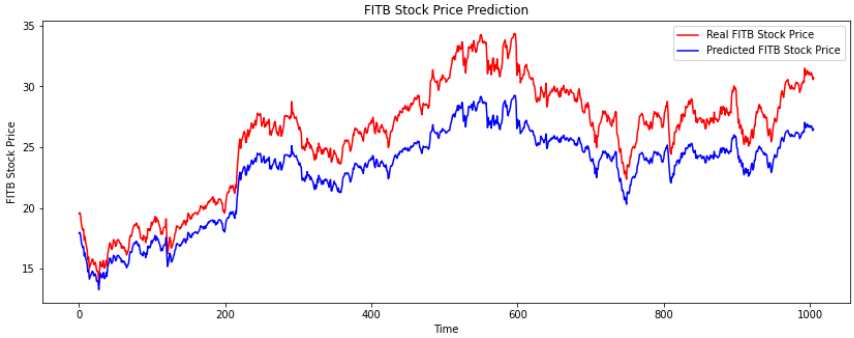

Figure 5: LSTM Forecast compared with the actual price

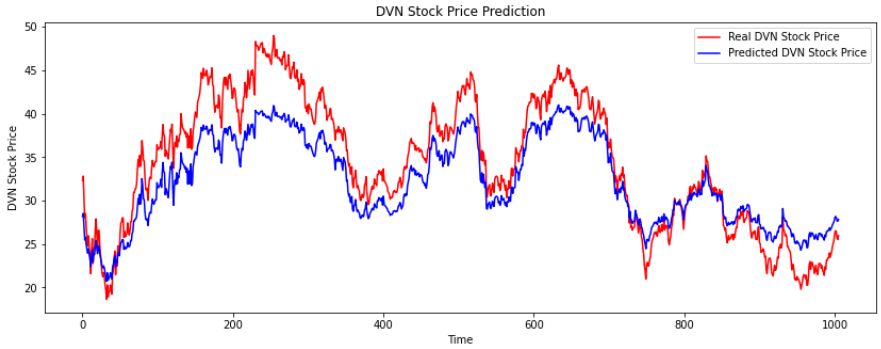

Figure 6: LSTM Forecast compared with the actual price

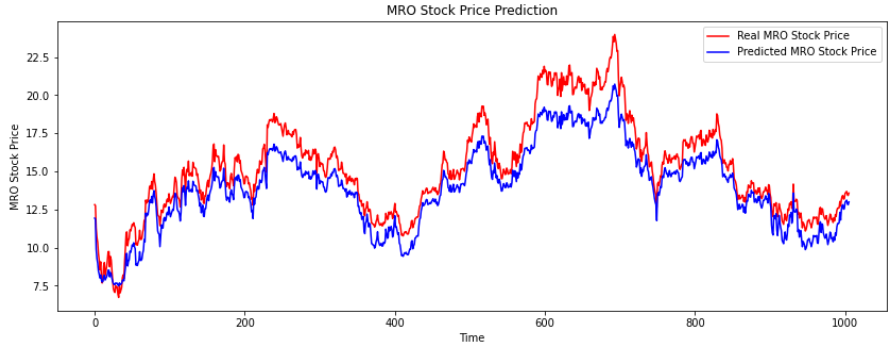

Figure 7: LSTM Forecast compared with the actual price

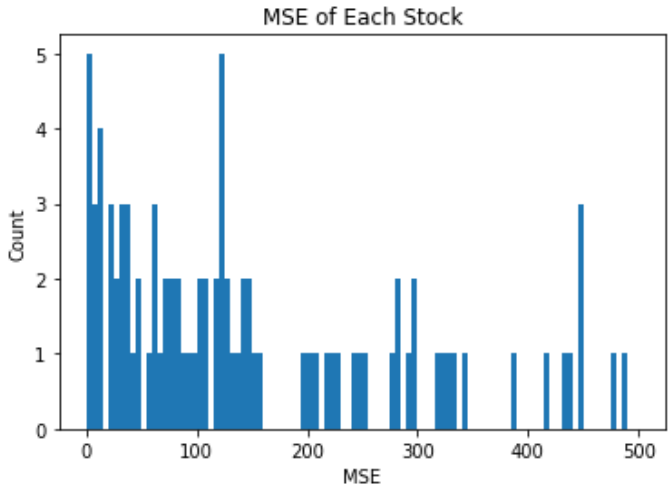

Figure 8: MSE Distribution

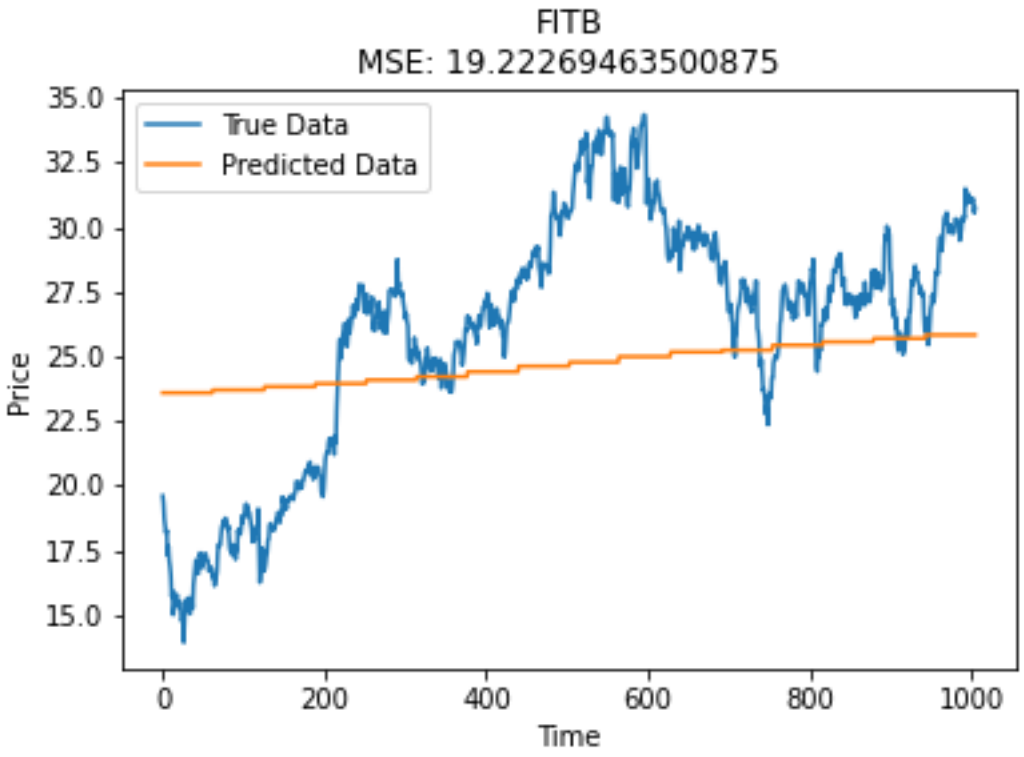

Figure 9: Forecast with XLNet and LSTM compared with the actual price

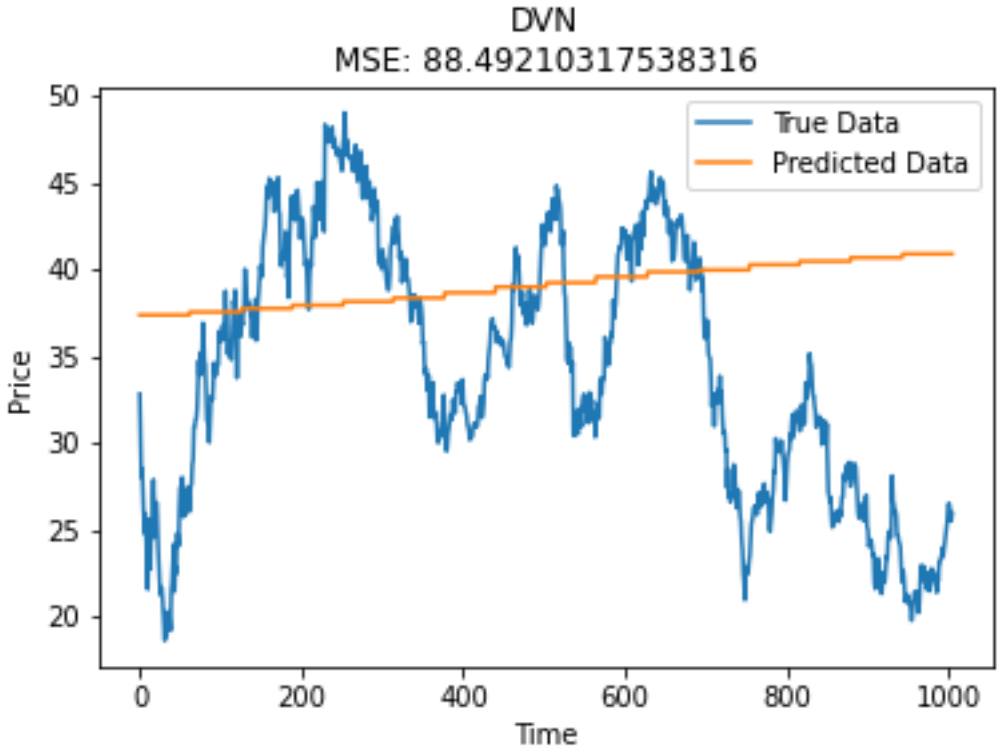

Figure 10: Forecast with XLNet and LSTM compared with the actual price

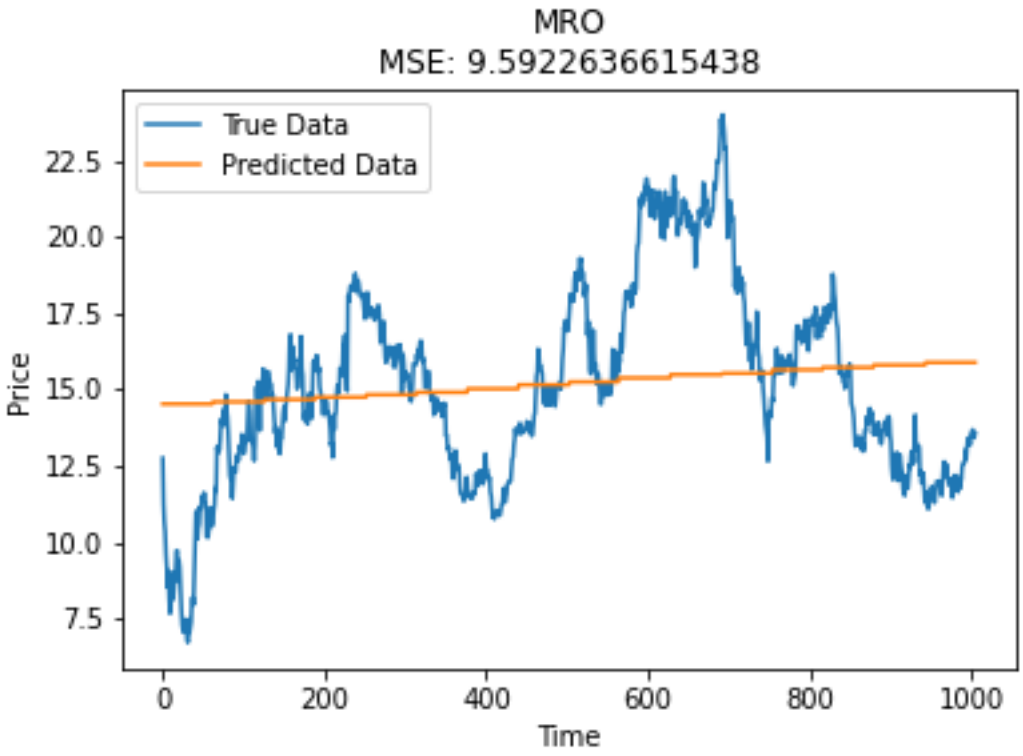

Figure 11: Forecast with XLNet and LSTM compared with the actual price