# CS337: AI/ML
# Project: Stock Market Prediction

Teja Bale 200050020
Guduru Manoj 200050044
Janaki Ram 200050112
Vavilapalli Sainath 200050125

November 24, 2022

## Contents

**1 Overview** **2**

**2 Data Extraction & Cleaning** **2**

**3 Models Used & Architectures** **2**

**4 Training & Results** **3**

# 1 Overview

Prediction and analysis of stock market is one of the most researched areas these days. We studied few linear(ARIMA, ARMA etc) and non-linear (MLP, CNN, RNN, Transformer etc) algorithms to forecast and predict the market. Of these, non-linear models outperformed the linear models because the non-linear models have the capability of capturing the non-linearity and various underlying patterns in the data. The linear models cannot be used to generalize on other stocks data as well.
Even in non-linear models, conventional models cannot be used because the movement of the price depends on the past data and the conventional models do not capture and account for the data dependence. One other reason is that the data is sequential data, i.e, not of a fixed shape. One can convert the data into blocks of fixed data size and can feed it to the network built (We implemented that as well as part of our analysis). Because of these difficulties, RNNs and its variants are commonly used for stock market forecasting

# 2 Data Extraction & Cleaning

To extract the price data, we used `yfinance`, a python package which fetches the data from *yahoo finance*. It returns the data in the form of a `pandas dataframe` which consists of 6 columns namely *open, close, high, low, adjusted close, volume*. It extracts the data given the *scrip, startdate, enddate* (end date is optional). It consists of NYSE stocks the most. To fetch NSE/BSE stocks data, we just need to add `.NS` to the end of the scrip (though most values of Indian stocks turned out to be NA values)
To decide upon which **features** to use, we ended up using the **close** price because we're forecasting the price of the market and the correlation among all the metrics other than volume came out to be 1.
After extracting the data using yfinance package, we scaled it using the

$$\text{MinMax} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

because the price variation of the stock might be high over a long period of time which would result in inefficient and improper training. The scaler is different for different stocks i.e, each stock is scaled using its own scaler. Since the prediction would be made on previous data upto a particular size of the window, it is like a hyperparameter that should be tuned for better results. Having a window size of 200 turned out to be reasonably well. So we rearranged the scaled values into shape (200,1) by sliding the window of size 200 over the entire dataset extracted

# 3 Models Used & Architectures

We used four models `CNN, LSTM, GRU, Transformer` for implementation and comparison purposes. CNN was used to know how a model which does not have memory property would

perform on predicting stock prices. LSTM (Long Short Term Memory) and GRU (Gated Recurrent Unit) are some of the architectures which are capable of remembering past data to generalize and predict. Both LSTM and GRU are used inorder to know which generalizes better over data because LSTM having a complex architecture, could result in overfitting as well.
Transformer is an attention-based model which uses positional encoding whose values represent the importance of the different input features in each time step, i.e which features that the model should pay more attention to. This along with multihead scaled dot product layers with residual connections and feed-forward linear bottleneck creates the transformer encoder. The architectures are as follows

- **CNN**: It comprises of couple of `Conv1D` layers each followed by a `MaxPooling1D` layer. Then there's a `Flatten` layer followed by a couple of `Dense` layers where the first layer has an activation of `ReLU`
- **LSTM** and **GRU**: Both the LSTM and GRU are made to have the same architecture so that the comparison will be more accurate. Each consists of 4 layers of its components and a Dense layer where the number of components in each layer is dependent on the window size used
- **Transformer** : In the earlier models, we used only one feature, i.e, the closing price to predict the stock price. However, while training the Transformer Model, we used a feature vector of size 16 consisting of various Momentum, Volume, Voltility and Trend Indicators. The model consists of an Encoder layer followed by a Linear Layer and a ReLU Layer. We used MultiHeadAttention in the Encoder layer.

# 4 Training & Results

We trained the model for 100 epochs using a batch size of 32 for CNN, LSTM and GRU models. We did not employ batching for the Transformer model but trained for 100 epochs. We used *MSE* as our loss metric. The losses for all the models are 3.7451e-04, 1.9480e-04, 2.2890e-04, 3.8875e-05 for CNN, LSTM, GRU and Transformer respectively. The MSE reported above(as well as in the subsequent steps) for Transformer has been obtained by dividing the loss by 80, as the data is approximately composed of 80 batches. This shows that LSTM is able to learn the stock's variation better among the first three models. This could've been due to overfitting as well. The performance of Transformer is even better as it uses a self attention mechanism allowing the model to focus on the relevant parts of the time-series to improve prediction qualities. We chose *BTC-UTC* to train the models because Crypto being the most volatile, has many patterns to identify. The graphs are as follows

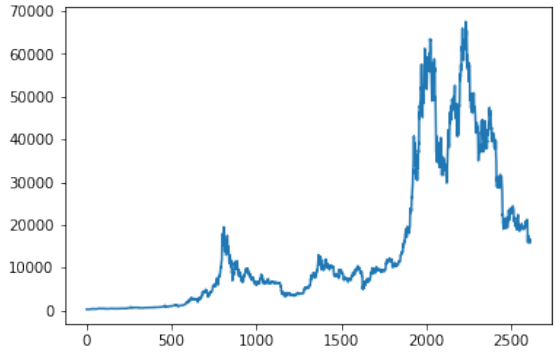


Figure 1: BTC-USD

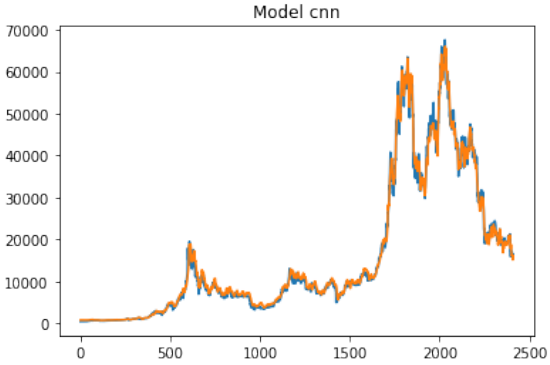


Figure 2: BTC-USD

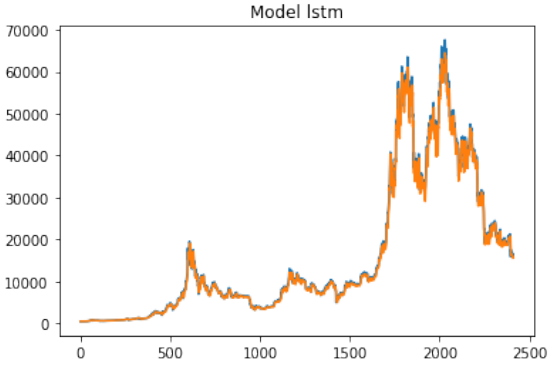


Figure 3: BTC-USD

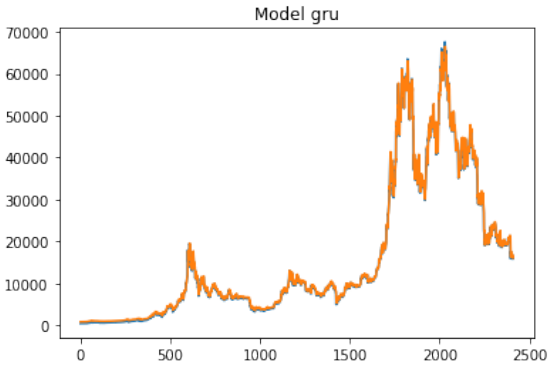


Figure 4: BTC-USD

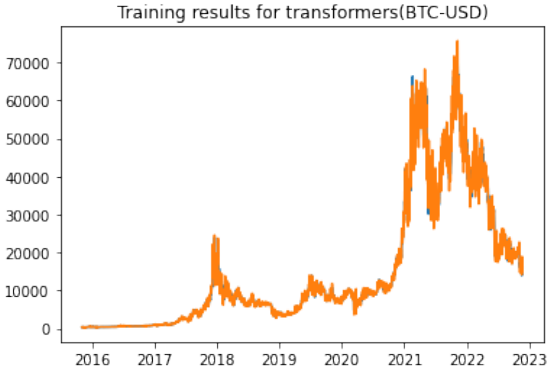


Figure 5: BTC-USD

We tested the models on other stocks: Apple, Google and Tesla over the last 10 years data. The loss values are as follows (In the order of CNN, LSTM and GRU, Transformers)

- **Apple**: 6.8796, 0.5471, 0.2523, 0.086
- **Google**: 5.1853, 0.5898, 0.2968, 0.0558
- **Tesla**: 5.6628, 0.5161, 0.3935, 0.8141

The loss values suggest that **CNN** was unable to generalize the patterns over other stocks compared to **LSTM & GRU** which generalized well for other stocks as well. Out of LSTM and GRU, GRU seems to achieve less loss even though LSTM had less loss during the training phase. This could have been due to overfitting during training phase. Based on the loss values, Transformer still performs better than the other models. The graphs are as follows

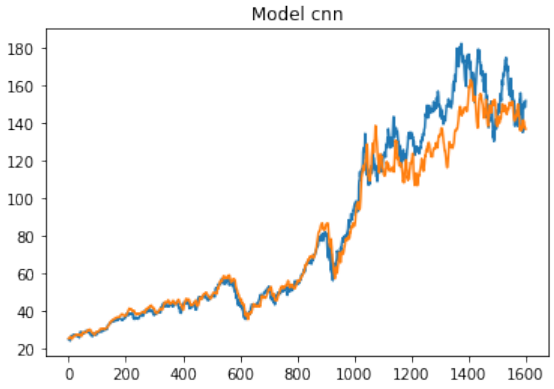


Figure 6: Apple


Figure 7: Apple

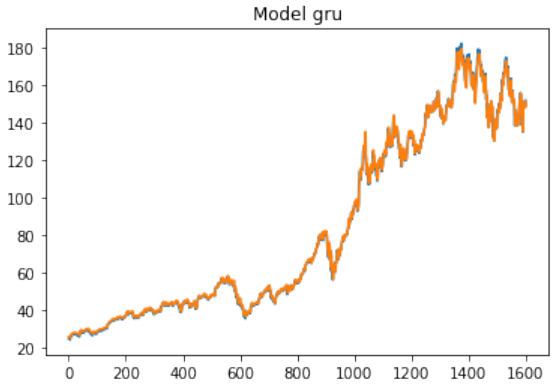


Figure 8: Apple

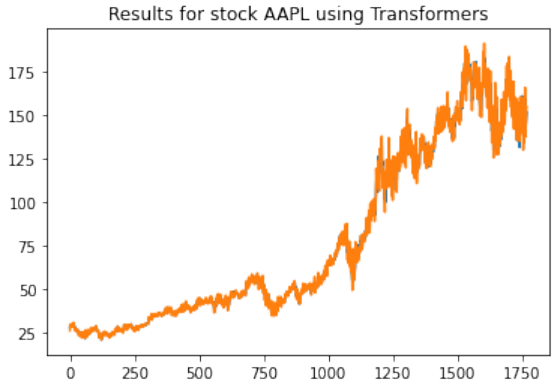


Figure 9: Apple

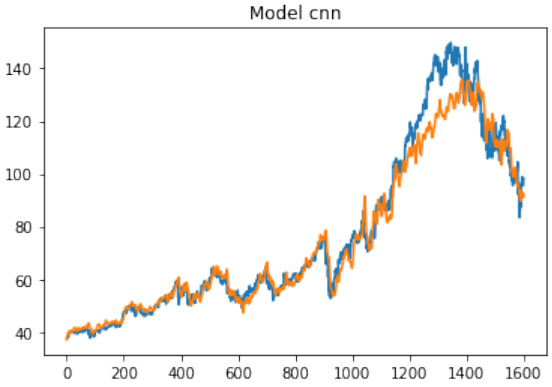


Figure 10: Google

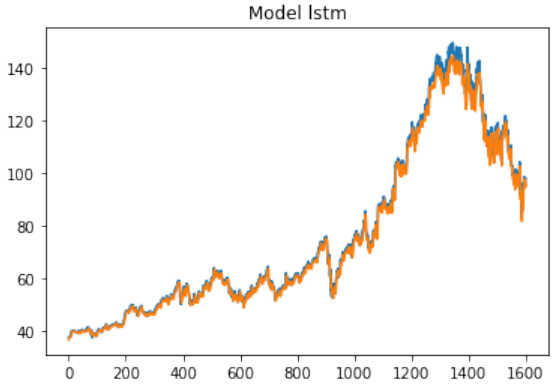


Figure 11: Google


Figure 12: Google

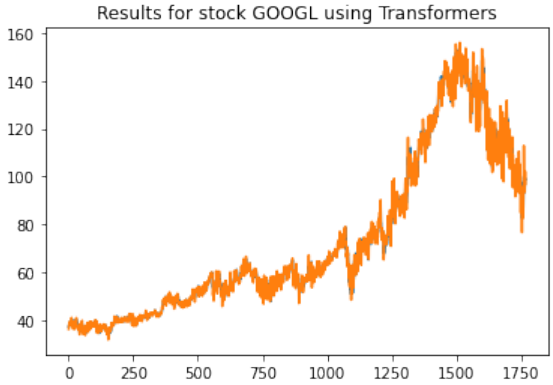


Figure 13: Google

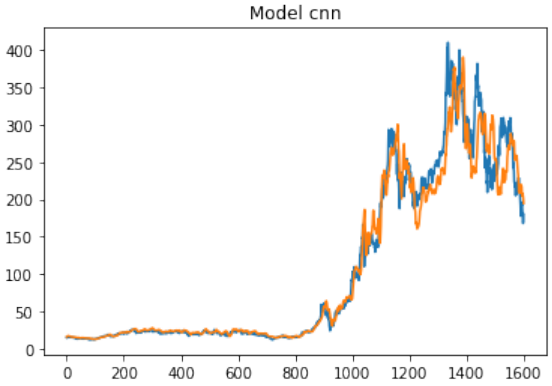


Figure 14: Tesla

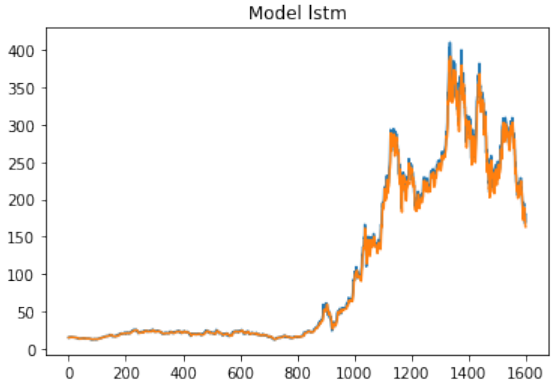


Figure 15: Tesla

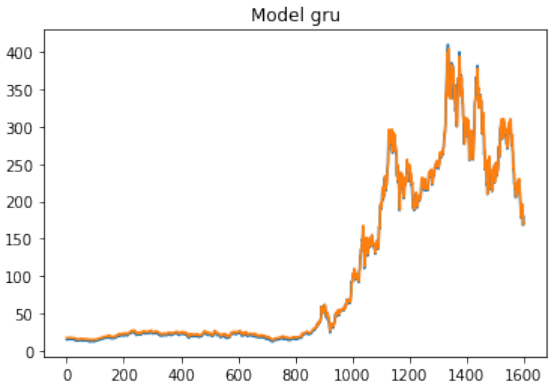


Figure 16: Tesla

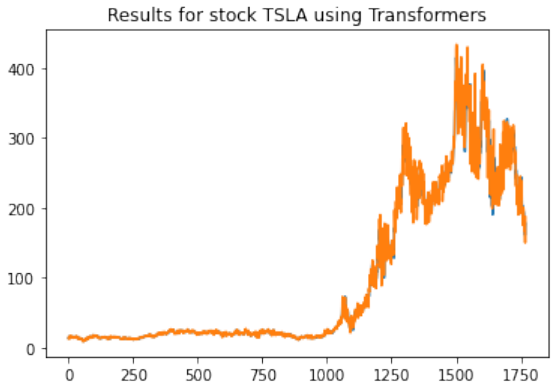


Figure 17: Tesla

## References

[1] https://www.sciencedirect.com/science/article/pii/S1877050918307828

[2] https://youtu.be/H6du_pfuznE

[3] https://github.com/olof98johansson/StockPrediction