REPORT

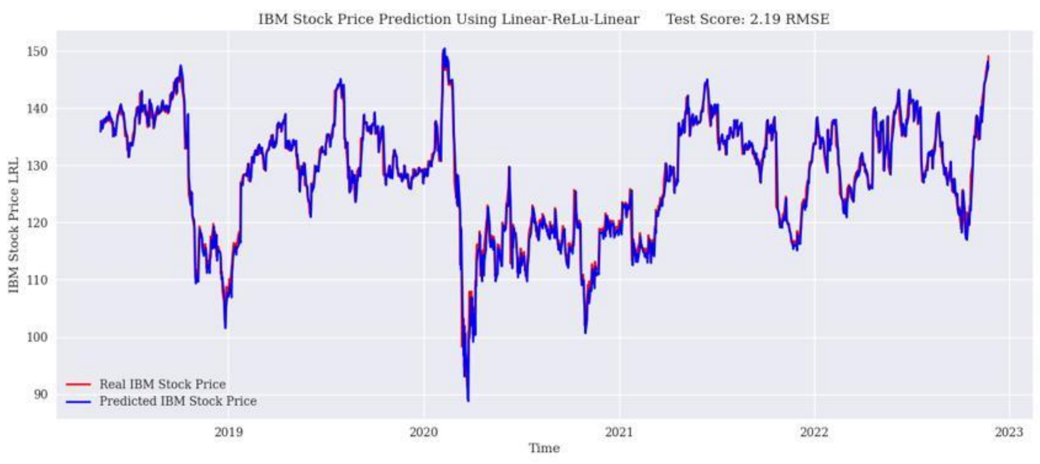
- Team Members :
    - K.Namitha : 20005006
    - Sai Pavan : 200050104
    - R. Rakesh Kumar : 200050120
    - KS. Vishwesh : 200050077

STOCK MARKET PREDICTION

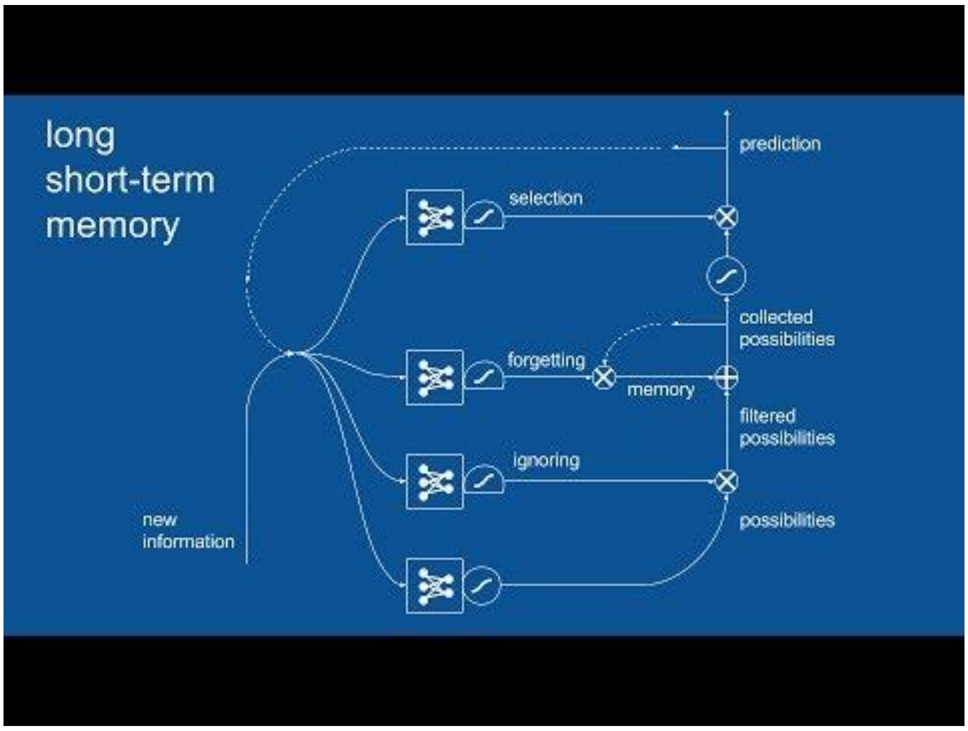
- In business the important thing is stocks.
- Stock traders are people who buy stocks and the most important quality needed for them is to estimate the value of stocks in the future and they need to do stocks transaction to make profit.
- Stock traders use Newspapers, media and previous years stock prices and take some assumptions based on it and confirm whether the prices rise or not.
- Now our target is we need to create a machine learning program in which it takes previous years stock prices and predict the stock prices for our desired time according to the situation till now.
- To do this we need to come up with a model which is trained based on the previous values of stock prices and use this trained model to find the stock price at desired time.
- To do this we need to come up with a model first. There are various models which can do this for us. For now let us consider the following models
    - LRL (Linear – Relu – Linear)
    - LSTM (Long Short Term Networks)
    - LRLSTM (Linear – Relu – LSTM)
    - GRU (Gated Recurrent Units)
- We trained each model with our input data.
- In case of stock market prediction we are having dates as inputs on one side. So unlike other straight forward input we need to take care a bit her.
- The data set we are using belongs to IBM company.

LRL Model

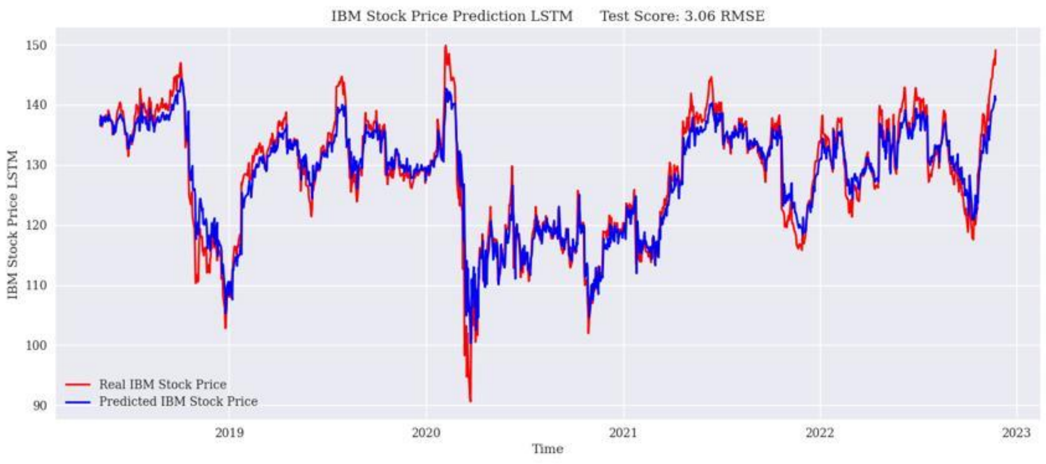
- The code file name is lrl.py
- The libraries we are using are numpy, random, pandas, pyplot, sklearn etc.
- We are using pandas for loading the data. The reason why we are using this specifically is there is a famous in-built “pandas.data_range” which take input as the range of dates we are considering as the dates are to be take carefully.
- In load data function we are loading dividing the data into training set and test set. Also since this is time series related problem we are going to consider 60 years as the sliding window for our code. It means we are going to predict a value by taking a look at the past 60 years. So we are going to create sets with 60 years values in it and by sliding it year by year.
- In LRL there are 3 layers, linear layer , Relu layer and another linear layer.
- We are using In-buit functions in torch to construct these layers which take input as number of inputs for the layer and number of outputs for that layer and it defines corresponding parameters.
- The optimizer we are using is “torch.optim.Adam()” and the loss function we are using “torch.nn.MSELoss()” with the help of these two we can directly use in-built backward function for training.



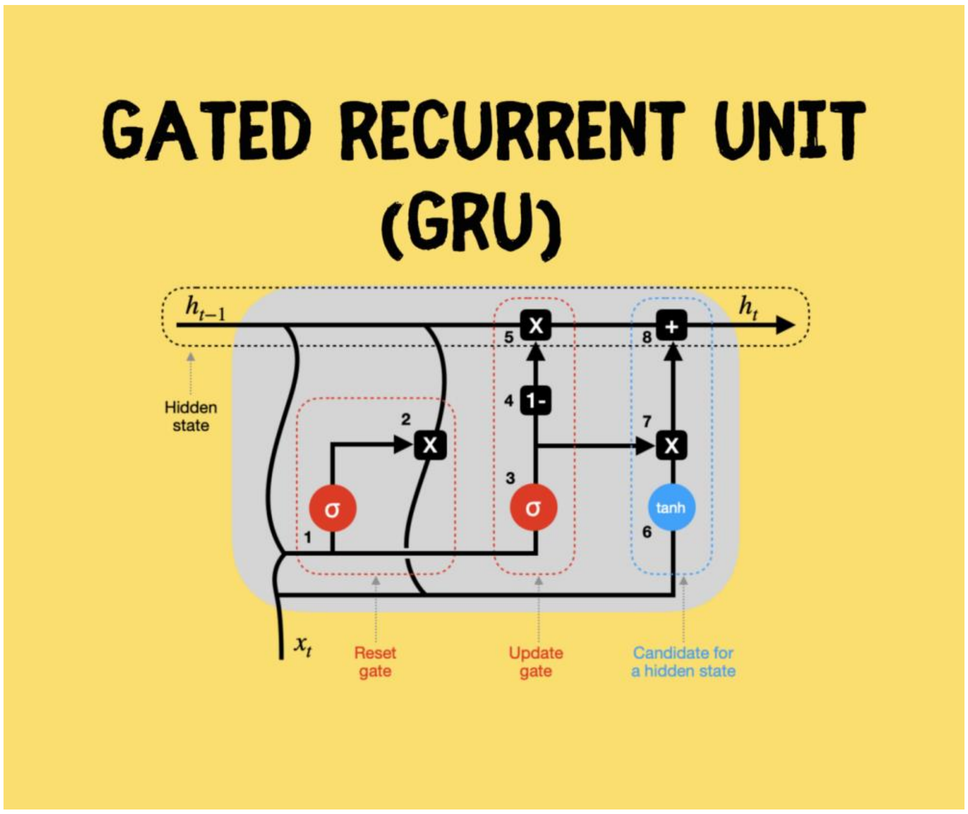
LSTM Model



- The code file is lstm.py
- The libraries we are using are numpy, random, pandas, pyplot, sklearn etc.
- We are doing everything as in LRL model but the only change in the model is the layers and the way the problem flow is going
- Torch library is like solely for machine learning there are various in-built functions for network layers.
- Likewise, even there exist a LSTM layer which take input as input dimension and hidden dimension which is the output dimension of lstm layer
- The reason they work so well is that LSTM for timeseries prediction because it can store past important information and forget the information that is not.
- LSTM has three gates:
- The input gate: The input gate adds information to the cell state,
- The forget gate: It removes the information that is no longer required by the model,
- The output gate: Output Gate at LSTM selects the information to be shown as output.



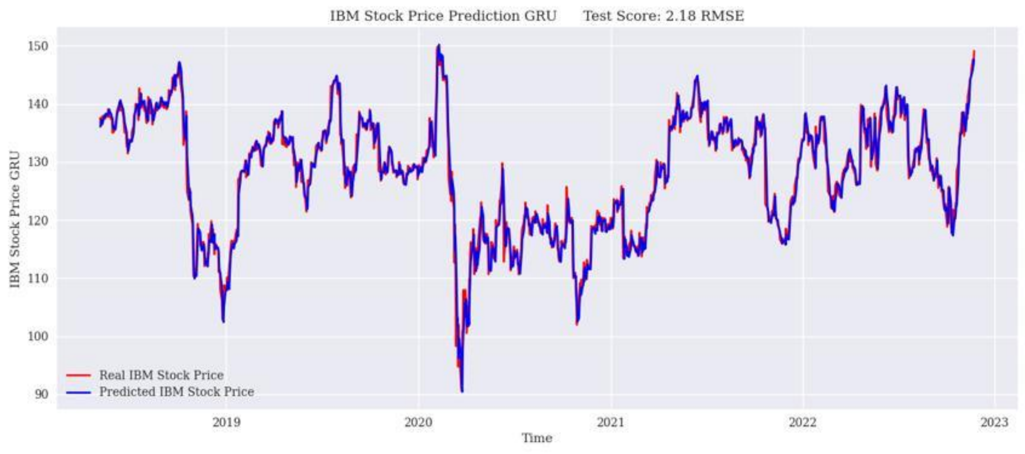
GRU Model



- The Code for this is in gru.py.
- Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks.

- The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate.
- To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, **update gate and reset gate**.

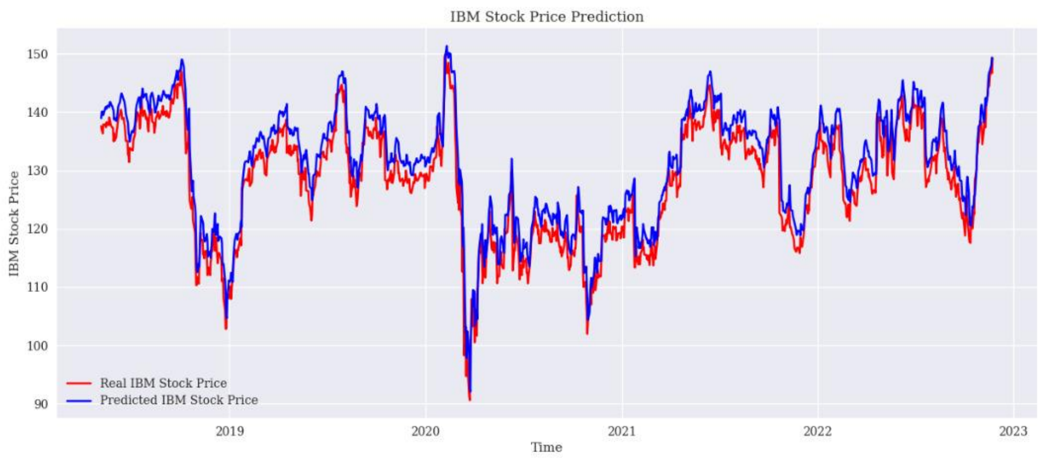
- When there are more layers in the network, the value of the product of derivative decreases until at some point the partial derivative of the loss function approaches a value close to zero, and the partial derivative vanishes. We call this the vanishing gradient problem.



LRLSTM Model

- The code for this is in **lrlstm.py**
- In this as we saw before, we are only going to change the layers in the model.
- We are first sending input to the linear layer and then sending the output of this layer to the ReLu layer and we

are sending the output of this layer to LSTM layer and again sending to linear layer.


IBM Stock Price Prediction
IBM Stock Price
Real IBM Stock Price
Predicted IBM Stock Price
Time