Mini-Project 4:

Stock Price Prediction using LSTM and CNN

Report by – Himanshu Chauhan

Submission Date – 03/27/2019

Table of Contents

(1) Problem Statement

Task 1: Use the daily [Open, High, Low, Volume] to predict [Close] **on that day** using **a fully-connected/dense neural network**. Use the first 70% of the records for training and the remaining 30% of the records for test. Report the RMSE of the model. Show the "regression lift chart" of your test data.

Task 2: Predict [Close] of a day based on the last 7 days' data [Open, High, Low, Volume, Close] using a **LSTM model**. In other words, we want to predict the price in the green cell using all the numbers in the red cell. Use the first 70% of the available records for training and the remaining 30% of the available records for test. Report the RMSE of the model. Show the "regression lift chart" of your test data.

Task 3: Do the same as Task 2 but use a **CNN model**. Report the RMSE of the model. Show the "regression lift chart" of your test data.

(2) Methodology

Used the following models to predict stock price prediction. Compared the RMSE and R2 score for all the following models.

1. Fully Connected Neural Network
2. LSTM
3. CNN

- Used the entire dataset for implementation
- Loaded the dataset into the data frame
- Dropped null rows
- Removed date and adj_close columns first since we don't need them.
- Studies and understood each column values and accordingly performed normalization. Since all the fields are numeric, therefore, scaled all the input featured to give it as input.
- Split the data, 70% for training and 30% for testing
- Used EarlyStopping and ModelCheckpoint when training neural networks, LSTM, and CNN using Tensorflow
- Added **dropout layer(s)** to see how it changes RMSE.
- Tuning the following hyperparameters to see how they affect performance
    - **Activation:** relu, sigmoid, tanh
    - **Number of layers and neuron count in each layer**
    - **Optimizer:** adam and sgd
    - **Kernel number and kernel size** (for CNN only)
    - **Number of LSTM layers and neuron count in each layer** (for LSTM only)
- Implemented Tensorflow models (Fully Connected Layer) with activation function ReLU, Sigmoid and Tanh and compared their performances
- Implemented CNN model to handle numeric data. Transformed the shape of the data to make it look like an 2D image to be used for Conv2D.
- Also plotted the Regression Lift Chart for each model to analyze the performance of the models for the given data.
- Implemented model by splitting the test data in the order of date(sorted) and another by using method test_train_split (random). Both the models gave almost similar performance. Random input gave little better RMSE score (**RMSE = 0.44610342383384705**) and compared to ordered split (**RMSE = 0.815961480140686**)

3. Experimental Results and Analysis

3.1 Fully Connected Layer

- Trained and experimented with the Tensorflow models with activation function ReLU, Sigmoid and Tanh. Further experimented with them by implementing adam and sgd optimizers to compare their performances.
- Also, implemented different combinations of neuron count and hidden layers as mentioned below in the table.
- Added a dropout layer to the models
- Calculated RMSE and R2 score for each model
- Plotted Lift chart for each model

3.2 LSTM

- Used the data set for training the model by using data for 7 days and then predicting the stock price for 8^{Th} day.
- Normalized the columns Open, High, Low, Volume and Close and used it as input data for the seven-day dataframe. Used close field as a output label.
- Generated a sequence (sliding window) of input data to give as input(3-D) to the LSTM model.
- Split the data into 70% training data and 30% testing data
- Used the optimizer function adam
- Used early stopping and model checkpointing
- Added a dropout layer to see how it affected the model
- Calculated the RMSE and R2 score

3.3 CNN

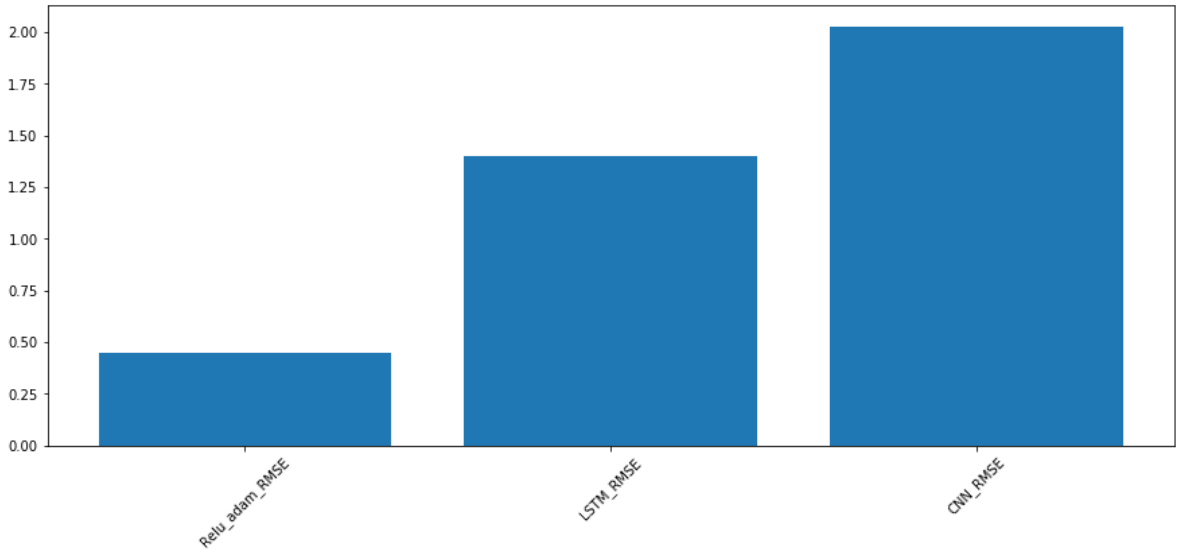
- Used the data set for training the model by using data for 7 days and then predicting the stock price for 8^{Th} day.
- Normalized the columns Open, High, Low, Volume and Close and used it as input data for the seven-day dataframe. Used close field as a output label.
- CNN model expects the data to be in 4-D, therefore reshaped the train and test data
- Gave input as an image with one row, seven columns and 5 channels (1,7,5)
- Split the data into 70% training data and 30% testing data
- Used the activation function ReLU and optimizer adam
- Used max pooling and dropout in the model.
- Used early stopping and model checkpointing
- Used early stopping and model checkpointing
- Added a dropout layer to see how it affected the model.
- Calculated the RMSE and R2 score

3.4 All Models (Task1, Task2 and Task3) Comparison Table

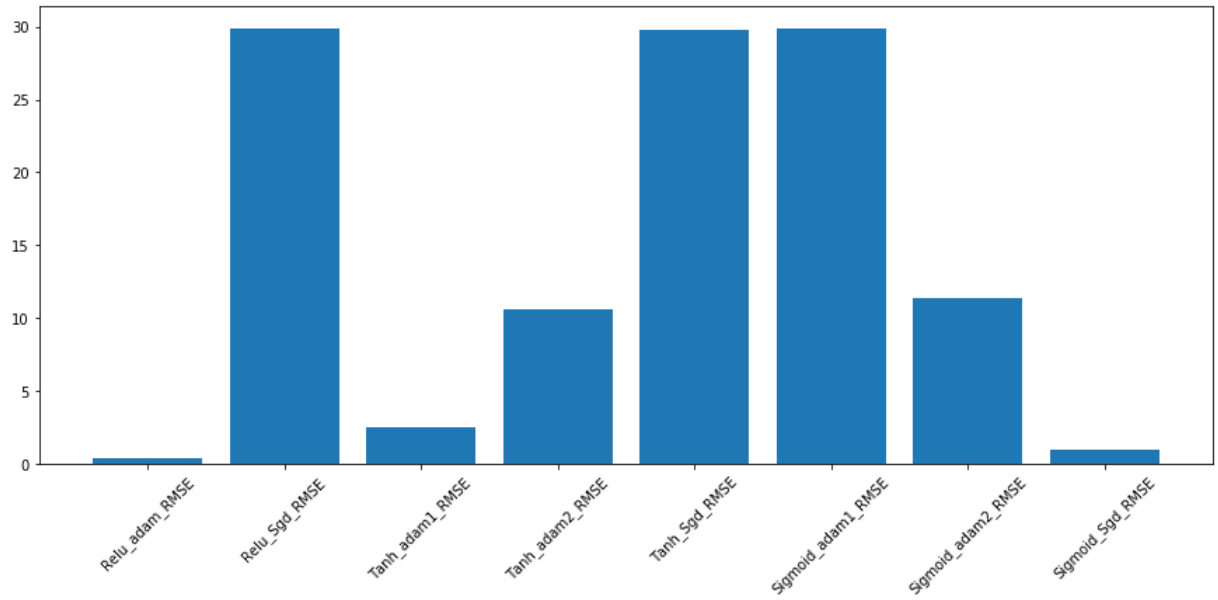
Models Tuning	RMSE	R2
Tensor flow regression neural network models		
ReLU + adam + 4 layers + early stopping + model checkpointing	0.37766316533088684	1.00
ReLU + adam + 5 layers + early stopping + model checkpointing + 2 Dropout Layers	0.44610342383384705	1.00
ReLU + adam + 6 layers + early stopping + model checkpointing + 3 Dropout Layers	0.588869035243988	1.00
ReLU + SGD + 4 layers + early stopping + Model Checkpointing	29.908370971679688	-0.00
Tanh + adam + 4 layers + early stopping + model checkpointing	2.475679636001587	0.99
Tanh + adam + 5 layers + early stopping + model checkpointing + 2 Dropout Layer	10.592331886291504	0.87
Tanh + SGD + 4 layers + early stopping + Model Checkpointing	29.803329467773438	0.01
Sigmoid + adam + 4 layers + early stopping + model checkpointing	29.908071517944336	-0.00
Sigmoid + adam + 5 layers + early stopping + model checkpointing + 2 Dropout Layer	29.908069610595703	0.004
Sigmoid + adam + 2 layers + early stopping + model checkpointing + 1 Dropout Layer	11.4020729064941	0.85
Sigmoid + SGD + 4 layers + early stopping + model checkpointing	0.9340808987617493	1.00
LSTM		
1- LSTM layer (64 UNITS) + 1 Dense Layer + adam	1.4032532490615899	0.9976412881560534
2- LSTM layers(80 and 50 UNITS in first and second lstm layer) + adam + 2 Dropout Layers	1.4680909946360896	0.9974678447605176
CNN		
CNN - 1 Conv2D layer(32 neurons) + Kernel size (1,3) + Dropout layer + 1 FC Layer	2.1363725872518033	0.994032355663934
CNN - 3 Conv2D layer (128 , 64 and 32 neurons) + Kernel size (1,3) + Dropout Layers + Max pool layers + 3 Fully Connected Layers	2.0277379697960916	0.9946303289150841

3.5 Graph Comparing RMSE Score for All Models (Task1, Task2 and Task3)

RMSE score for Fully Connected Network vs LSTM vs CNN



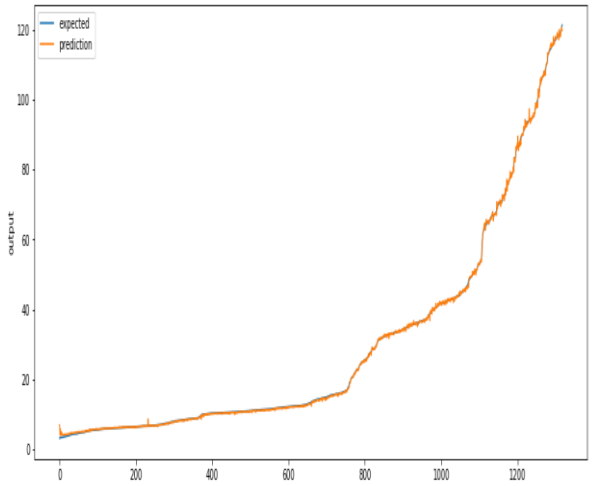
RMSE score comparison for ass FCN models with different comparison- TASK1



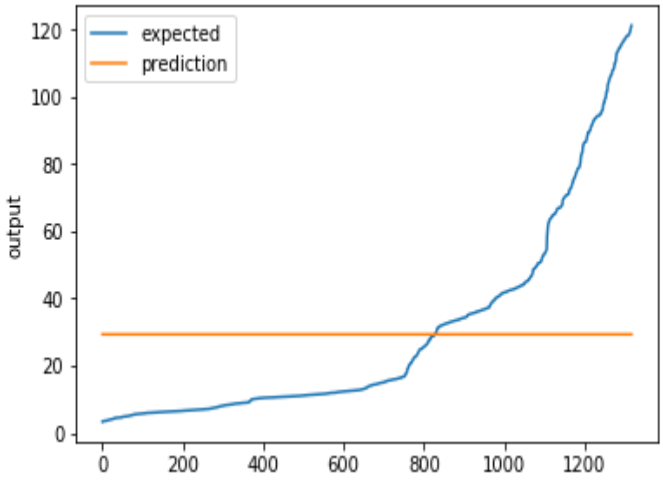
3.6 Regression Lift Charts for All Models (Task1, Task2, Task3)

Fully Connected Network

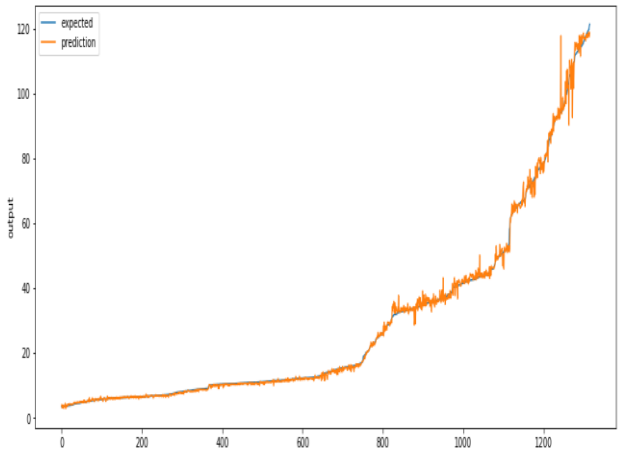
Relu + adam



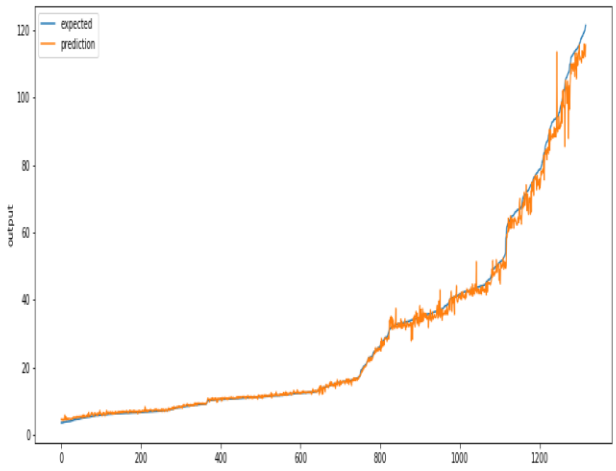
Relu + SGD



LSTM



Convolutional Neural Networks (CNN)



4. Task Division: N/A (Individually completed the project)

5. Project Reflection

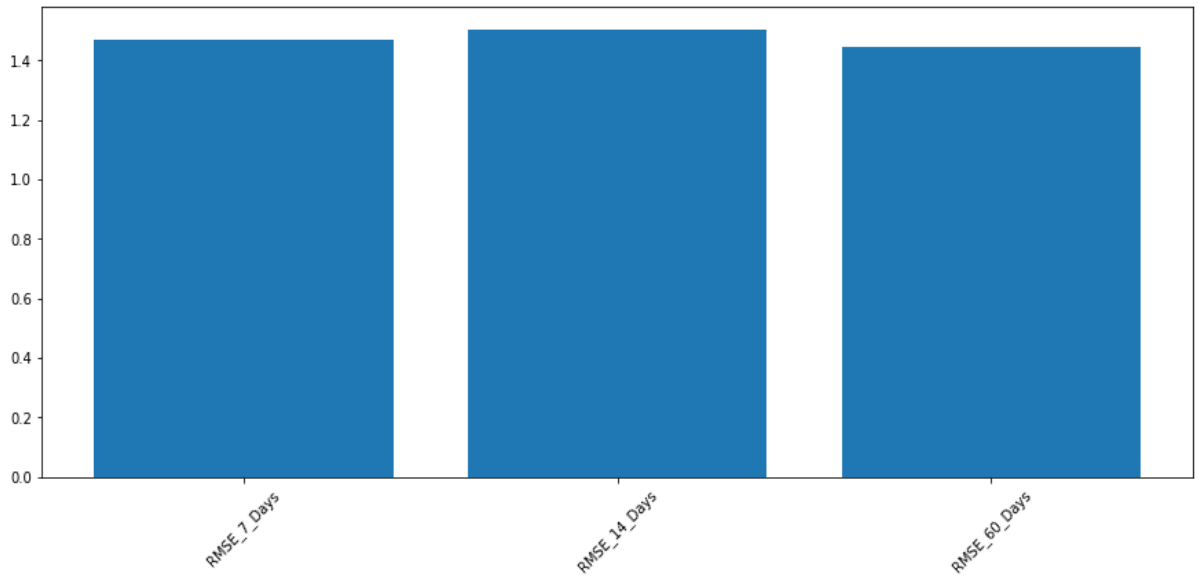
- Learnt how to implement LSTM models to predict stock price. Understood how to give the input as a sequence of inputs to the LSTM models.
- Compared the performance of the LSTM models with CNN and other FCN models.
- Also, it was good to see the results various combinations of activation functions and optimizers . For example Relu and SGD for FCN gave very bad performance.
- All Models – FCN, LSTM, CNN gave good performance
- Also experimented with the splitting the data into test and train data.
- Implemented model by splitting the test data in the order of date(sorted) and another by using method test_train_split (random). Both the models gave almost similar performance. Random input gave little RMSR score (RMSE = 0.44610342383384705) and compared to ordered split (RMSE = 0.815961480140686)

6. Additional Features:

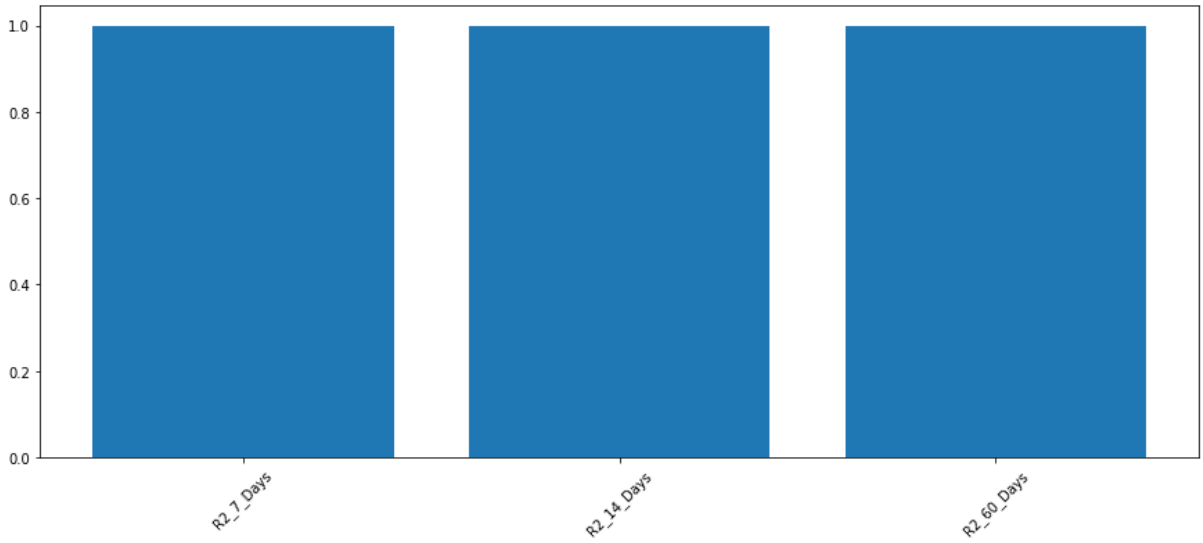
6.1 Experiment with different values of N – LSTM

- I wanted to see how the performance of the models changes by training the model for 2 weeks (14 days) data vs 2 months data (60 days)
- N = 60 days gave a little better RMSE score than others.

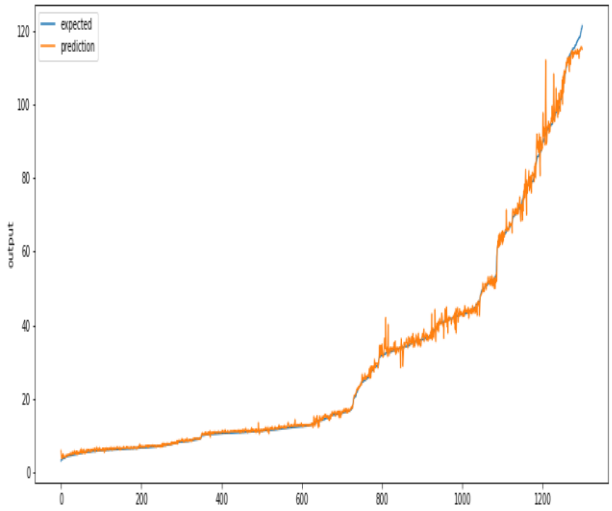
RMSE score for three N values: 7 days vs 14 days vs 60 days



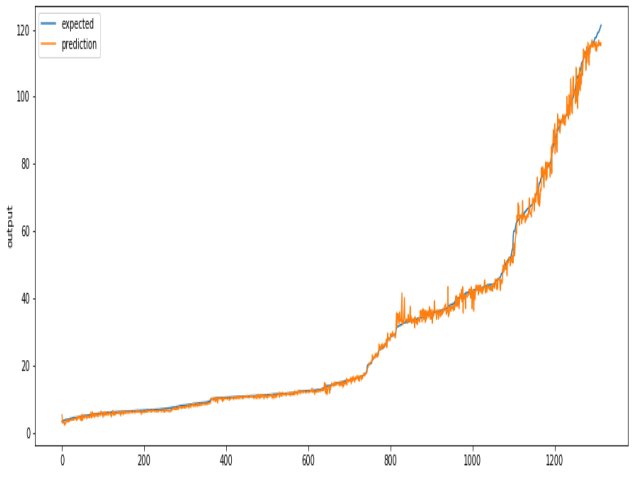
R2 score for three N values: 7-Days vs 14-Days vs 60 -Days



Lift Chart N = 14 Days



Lift Chart N = 60 Days



6.2 LSTM – Continuous Time period – Nest 5 days prediction

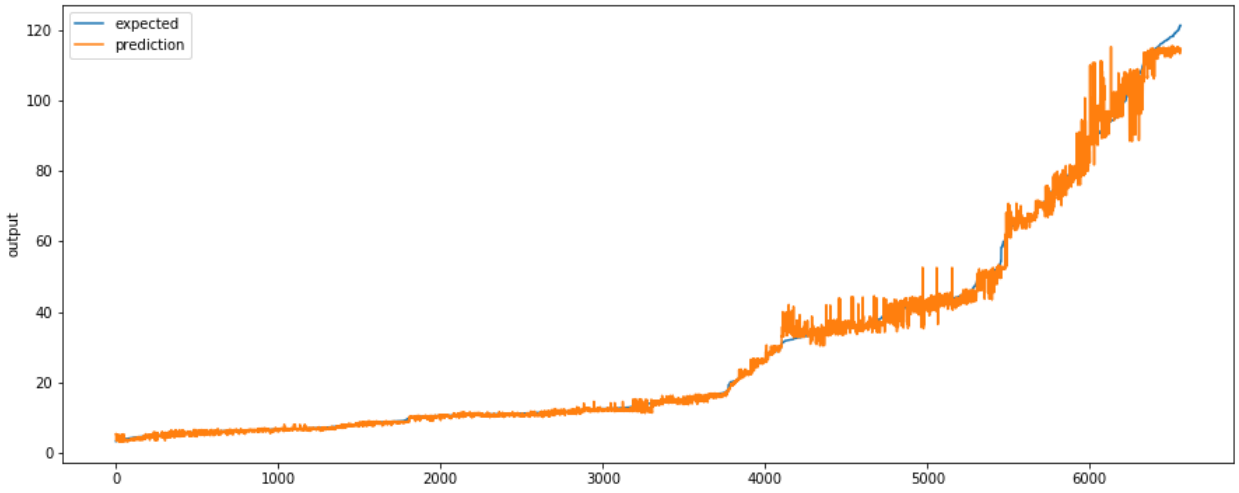
- Implemented this model by giving 5 output neurons to the output layer (Multi output Regressor)
- Created the sequence of 5 days for output and 15 days for input features.
- Below in an Example of the actual and predicted values for 5 continuous days by the Model.

Day 1
Actual Value: [11.278437, 11.211143, 11.397026, 11.44823, 11.464707],
predicted Value: [13.7275, 13.8875, 14.0025, 14.065, 13.9675]

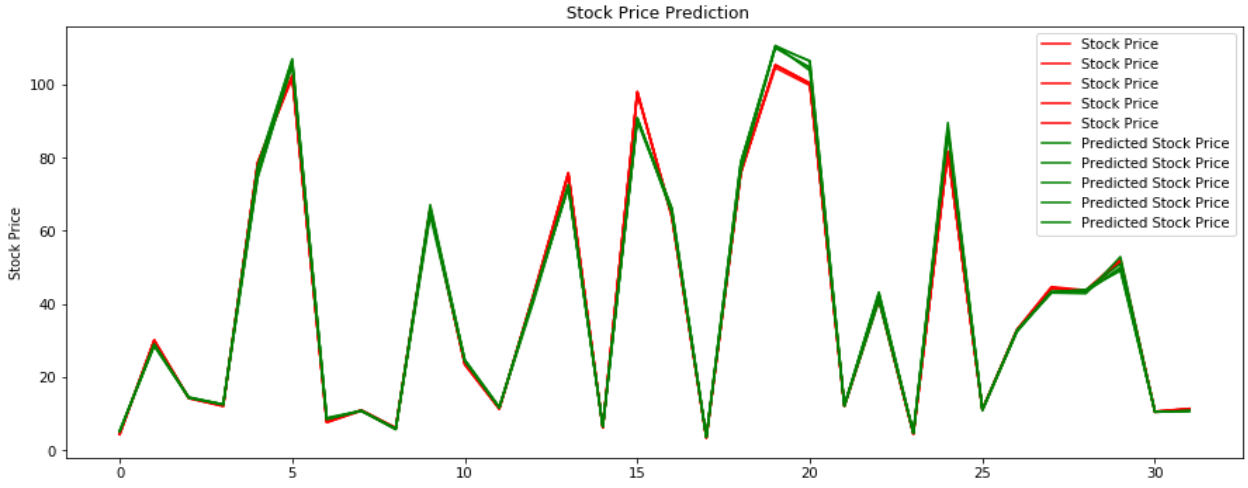
Day 2
Actual Value: [6.6527963, 6.595593, 6.756828, 6.8500204, 6.7705684],
predicted Value: [7.1, 6.9125, 6.75, 7.2125, 6.85]

Day 3
Actual Value: [6.820395, 6.761118, 6.927421, 7.018788, 6.941057],
predicted Value: [6.975, 7., 6.8975, 7., 6.9125]

Lift Chart for continuous value prediction for N = 5



5-days(continuous) prediction and actual price for 30 Records

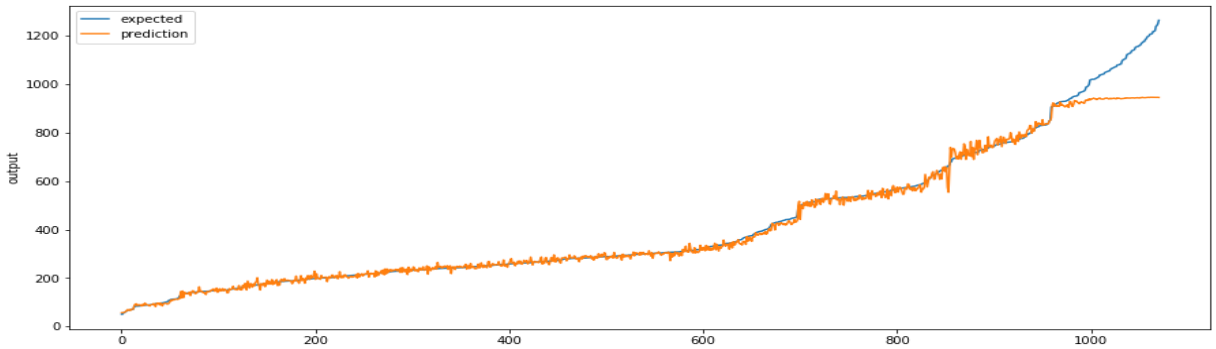


6.3 Other Companies - LSTM

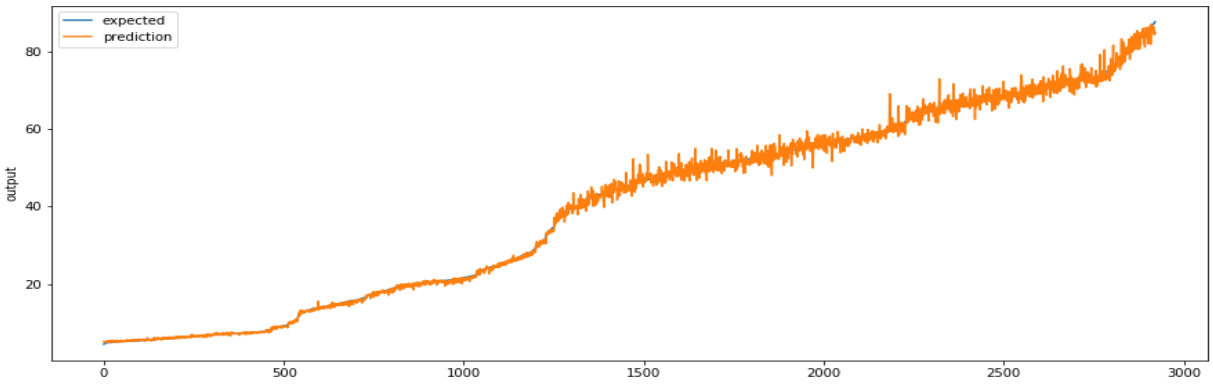
- Implemented LSTM model for 3 other Companies
- Below is the lift chart and Model RMSE score for each model for each company

Models	RMSE	R2
LSTM Models		
Google	49.57149827681670	0.963399847873463
JP Morgan	1.1376898416597356	0.9978158257269987
oyal Dutch Shell	1.3297917361282259	0.996820632677273

Google Lift Chart



Royal Dutch Shell - Lift chart



JP Morgan – Lift Chart