STOCK PRICE PREDICTION
REPORT

- **CREATION OF DATASET:-**
  For prediction of stock price, initially a few days of earlier record is essential and the current day record is taken as target. It was mentioned to take the previous 3 days and create a new dataset. When the dataset set was examined, it was noted that entry for each day from 2015 to 2020 was given with the recent entries at the top of the dataset. Hence I used reverse indexing thus starting at 3 and each time the entries are calculated as " i-3" which is taken as the data and the present open which is "i" is taken as the target. Thus this creates the custom dataset by considering only the last 3 entries as one data entry.

```
# Creating a custom data structure with 3 timesteps and 1 output
X_train_data = []
y_train_data = []
for i in range(3, 1259):
    X_train_data.append(training_set[i-3:i, :])
    y_train_data.append(training_set[i, 1])
X_train_data, y_train_data = np.array(X_train_data), np.array(y_train_data)
```

- **PREPROCESSING FOLLOWED**
  In this case I have used a standard min max scaler. This converts all the features in the range from 0 to 1. Using min max scaler the resultant would be a smaller standard deviation, which can possibly remove the outliers. I even tried without this but the loss was very much and the model was not getting trained. Hence pre-processing here is a mandatory step for learning and better performance of the model. The training data and label was preprocessed and the scaler was saved to use the same scaler for testing.

```
# PREPROCESSING FOR DATA AND LABELS
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(df_train.iloc[:,:12])
training_y_label_scaled=sc.fit_transform(np.array(df_train["Labels"]).reshape(-1,1))

# Saving the scaler for using the same normalization for testing

joblib.dump(sc, os.path.join(datapath,"models","sc_RNN.save"))
```

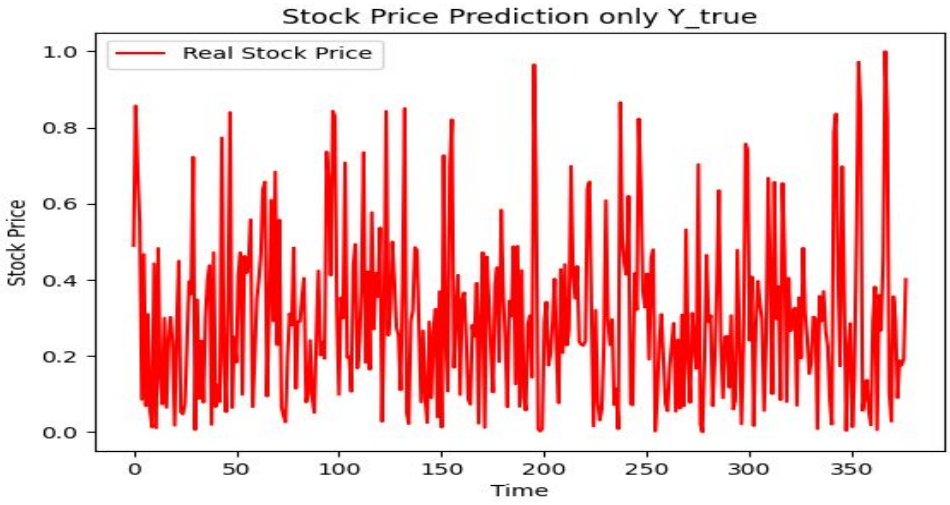
- **All design steps you went through in finding the best network in your report and how you chose your final design:-**
  Since it was a sequence related prediction, I tried with many LSTM layers with different LSTM units per layer and also used different optimizers like SGD,rmsprop. It was found that with SGD the training was quite slow. Different batch sizes were also tested.

- **Architecture of your final network, number of epochs, batch size (if needed), loss function, training algorithm:-**
  After examining various architectures, Since the created custom training dataset was around 1256, single hidden layer was essential to build the model. By trying different hidden units it was noted that 512 produced the minimum loss. The epoch used was 50 since at the beginning there were few fluctuations in the loss. Mean squared error loss function was used. The mean squared error(MSE) measures the distance between predicted and real values and out of various other loss function like RMSE with all the testing conditions the same it was noted that MSE performed better.

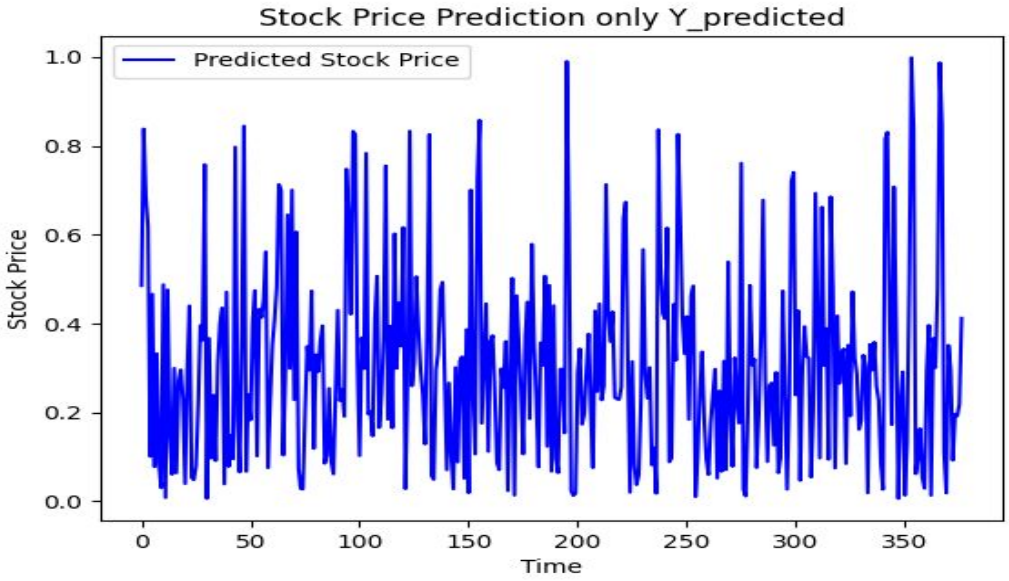
- **Output of the training loop with comments on your output.**
  By using the Mean square error the final error value obtained at the end of 50 epochs was **3.0803e-04.** I tried with different epochs and found 50 epochs was sufficient to get the lowest loss value. There was no sign of overfitting as the training loss (**2.2984e-04**) was also around the range of validation loss. This indicated that there was no overfitting even if there was a hike in validation loss at the 49th epoch.

```
Epoch 47/50
703/703 [==============================] - 1s 1ms/step - loss: 2.4501e-04 - val_loss: 4.2922e-04
Epoch 48/50
703/703 [==============================] - 1s 1ms/step - loss: 2.6028e-04 - val_loss: 3.1735e-04
Epoch 49/50
703/703 [==============================] - 1s 995us/step - loss: 2.3651e-04 - val_loss: 3.6510e-04
Epoch 50/50
703/703 [==============================] - 1s 961us/step - loss: 2.2984e-04 - val_loss: 3.0803e-04
```

- **Output from testing, including the final plot and your comment on it.**
  By analysing the final graph obtained by overlapping the true value and predicted value, it was found that the model has performed well as the predicted value almost overlapped the true value. This also gives evidence that the model has not overfit the training dataset. By trying different models it was noted that this model performed the best.



Plot for stock price with the models predicted outcome



Overlapped plot for true and predicted value