Stock Price Prediction Using LSTM

Introduction:

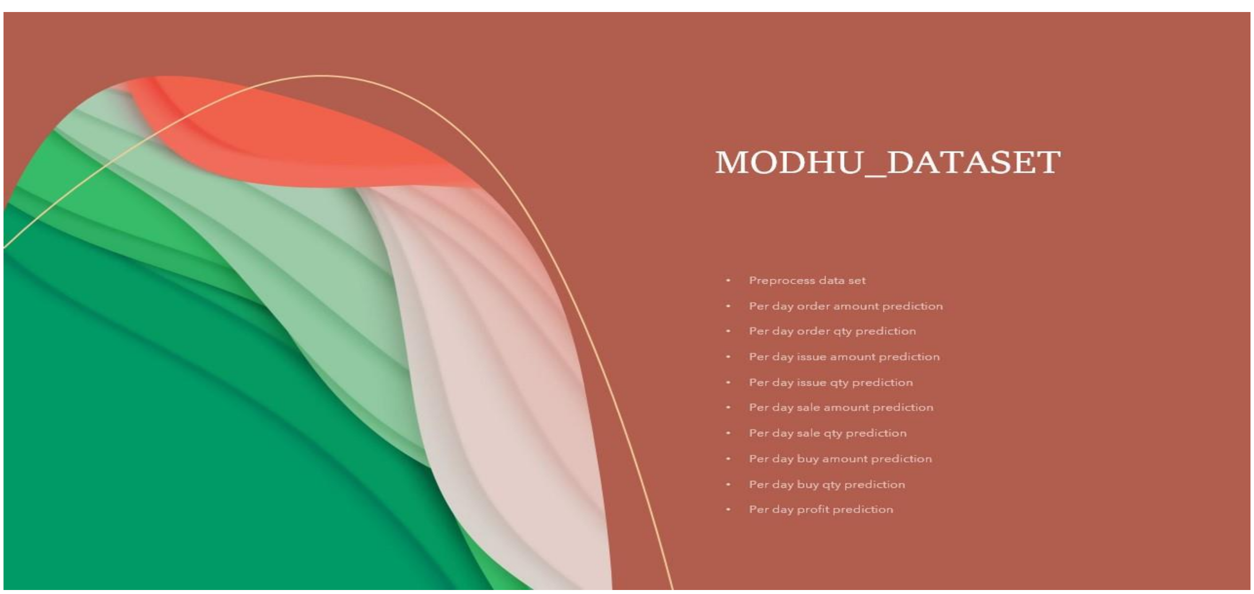
In this task, the future stock of local shop of Bangladesh are predicted using the **LSTM** Recurrent Neural Network. Our task is to predict stock prices, sale prices and sale quantity for a few days, which is a time series problem. The **LSTM** model is very popular in time-series forecasting, and this is the reason why this model is chosen in this task. The historical prices of local shops are collected manually. We have used 30 days of historical price data, from 01.01.2020 to 31.01.2020.

This data set contains 54249 observations with 40 attributes. After preprocessing, only Date, Amount, Rate and Qty columns, a total of 4 columns, are taken as these columns have main significance in the dataset. The **LSTM** model is trained on this entire dataset. The stock prices for this new duration will be predicted by the already trained **LSTM** model, and the predicted prices will be plotted against the original prices to visualize the model's accuracy.

Long short-term memory (LSTM): LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections. It can not only process single data points, but also entire sequences of data (such as speech or video inputs). LSTM models are able to store information over a period of time.

Modules: Keras, Tensorflow, Pandas, Scikit-Learn & Numpy.

Work target:



Implementation:

1. Let's import the required Libraries:

```python
# import libaray
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math

from keras.models import Sequential
from keras.layers import Dense, SimpleRNN, LSTM,Dropout
from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.metrics import mean_squared_error
from numpy import array
import seaborn as sns
```

2. Load the data:

```python
pd.set_option('display.max_columns', None)
orderdata = pd.read_excel("MODHU_DATA.xlsx", sheet_name='ORDER')
orderdata .drop(orderdata [orderdata.TRANSTP =='ANDROID APP' ].index, inplac
#orderdata=orderdata[orderdata['TRANSTP']=='ANDROID APP']
issuedata = pd.read_excel("MODHU_DATA.xlsx", sheet_name='ISSUE')
saledata = pd.read_excel("MODHU_DATA.xlsx", sheet_name='SALE')
buydata= pd.read_excel("MODHU_DATA.xlsx", sheet_name='BUY')
```

3. Preprocessing the data:

```python
data=orderdata[['TRANSDT','AMOUNT','QTY','RATE']]
#data['TRANSDT']=pd.to_datetime(data['TRANSDT'], format='%Y-%m-%d %H:%M:%S')
data=data.groupby("TRANSDT").sum()
data=data.reset_index()
```

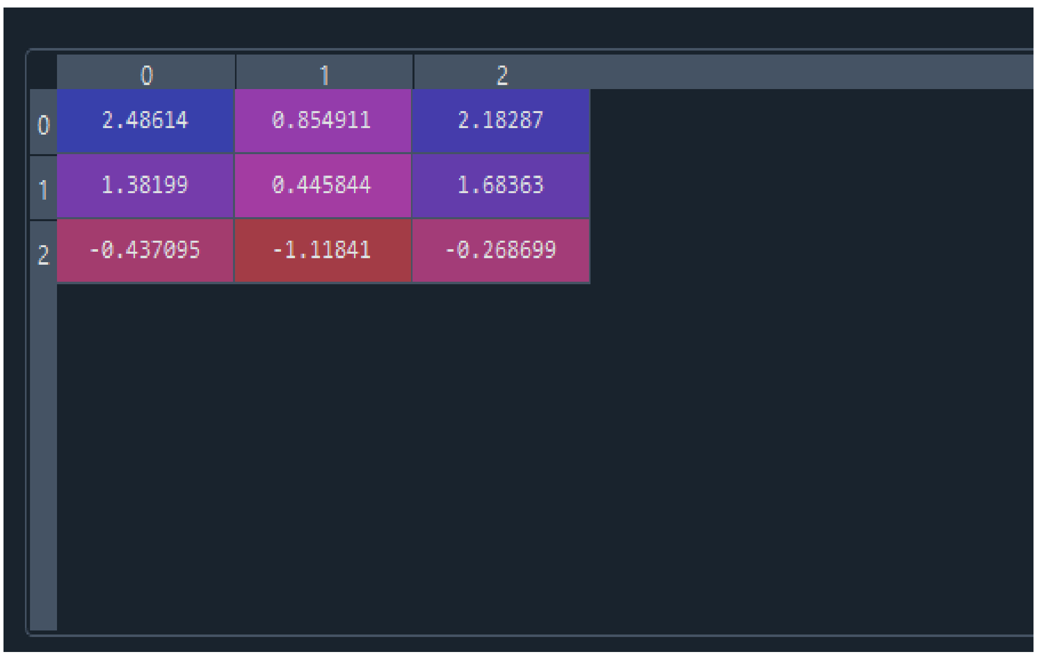
4. Visualize the dataset:

Index	TRANSDT	AMOUNT	QTY	RATE
0	2020-01-01 00:00:00	1.17651e+06	24742	266428
1	2020-01-02 00:00:00	1.02746e+06	23356	252992
2	2020-01-03 00:00:00	781905	18056	200448
3	2020-01-04 00:00:00	765468	17763.5	203744
4	2020-01-05 00:00:00	885609	25092.5	203121
5	2020-01-06 00:00:00	765427	20968	191988
6	2020-01-07 00:00:00	794933	19631.5	187961
7	2020-01-08 00:00:00	873522	22416	215699
8	2020-01-09 00:00:00	1.10808e+06	29784.5	253092

5. Normalize the data:

	0	1	2
0	2.48614	0.854911	2.18287
1	1.38199	0.445844	1.68363
2	-0.437095	-1.11841	-0.268699
3	-0.558861	-1.20474	-0.14621
4	0.331145	0.958358	-0.169365
5	-0.559161	-0.258955	-0.583019
6	-0.34058	-0.653412	-0.732643
7	0.241607	0.168411	0.297964
8	1.97917	2.34316	1.68734
9	-0.140891	-0.574461	-0.0966012

6. Split data trainX and trainY:



	0	1	2
0	2.48614	0.854911	2.18287
1	1.38199	0.445844	1.68363
2	-0.437095	-1.11841	-0.268699

	0
0	-1.20474
1	0.958358
2	-0.258955
3	-0.653412
4	0.168411
5	2.34316
6	-0.574461
7	2.16785
8	-1.51464
9	-1.0855

7. Create LSTM model:

```
trainY shape == (28, 1).
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 3, 50)             10800

 lstm_1 (LSTM)               (None, 3, 50)             20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 51,251
Trainable params: 51,251
```

8. Compile and train the model:

```
0.8453 - val_loss: 0.4004 - val_mean_absolute_error: 0.5076
Epoch 4/100
25/25 [==============================] - 0s 440us/sample - loss: 1.1030 - mean_absolute_error:
0.8451 - val_loss: 0.4020 - val_mean_absolute_error: 0.5083
Epoch 5/100
25/25 [==============================] - 0s 400us/sample - loss: 1.1025 - mean_absolute_error:
0.8449 - val_loss: 0.4035 - val_mean_absolute_error: 0.5090
Epoch 6/100
25/25 [==============================] - 0s 440us/sample - loss: 1.1020 - mean_absolute_error:
0.8447 - val_loss: 0.4051 - val_mean_absolute_error: 0.5097
Epoch 7/100
25/25 [==============================] - 0s 360us/sample - loss: 1.1015 - mean_absolute_error:
0.8446 - val_loss: 0.4067 - val_mean_absolute_error: 0.5104
Epoch 8/100
25/25 [==============================] - 0s 480us/sample - loss: 1.1010 - mean_absolute_error:
0.8446 - val_loss: 0.4083 - val_mean_absolute_error: 0.5110
```

9. Prediction the data:

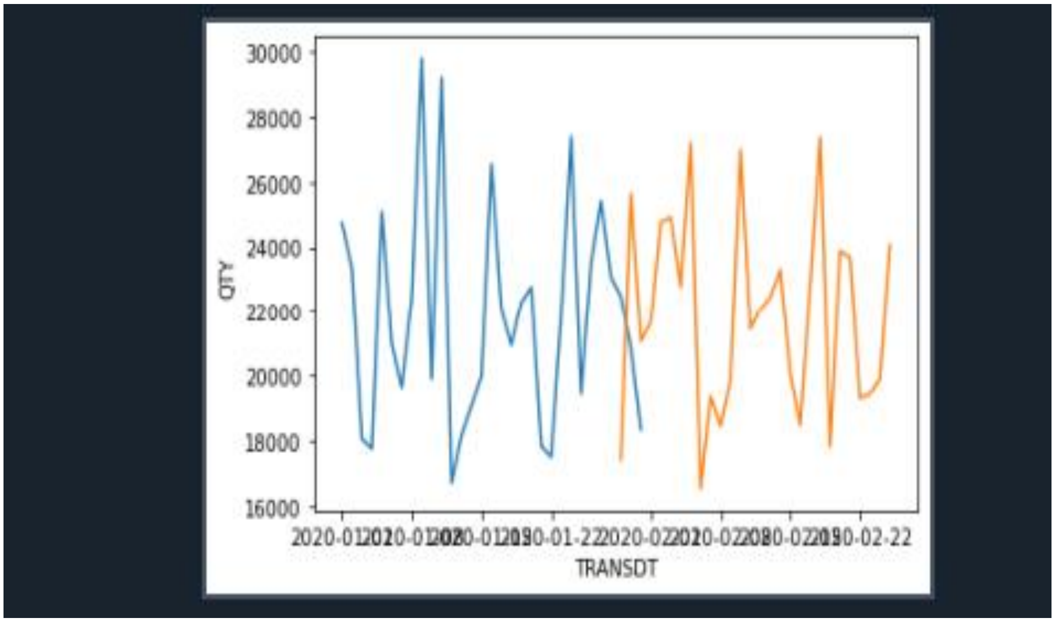
```
history = model.fit(trainX, trainY, epochs=100,batch_size=64,validation_split=0.1, ver
n_days_for_prediction=28
#n_past = 14

predict_period_dates = pd.date_range(list(train_dates)[-n_past], periods=n_days_for_pr
prediction = model.predict(trainX[-n_days_for_prediction:])


prediction_copies = np.repeat(prediction, df_for_training.shape[1], axis=-1)
y_pred_future = scaler.inverse_transform(prediction_copies)[:,1]
```

Result:

1. Per day order Qty prediction:

Index	TRANSDT	AMOUNT
0	2020-01-29 00:00:00	764929.4
1	2020-01-30 00:00:00	876666.1
2	2020-01-31 00:00:00	767181.0
3	2020-02-01 00:00:00	757286.5
4	2020-02-02 00:00:00	851677.06
5	2020-02-03 00:00:00	1048555.56
6	2020-02-04 00:00:00	824392.1
7	2020-02-05 00:00:00	787274.8
8	2020-02-06 00:00:00	657978.1

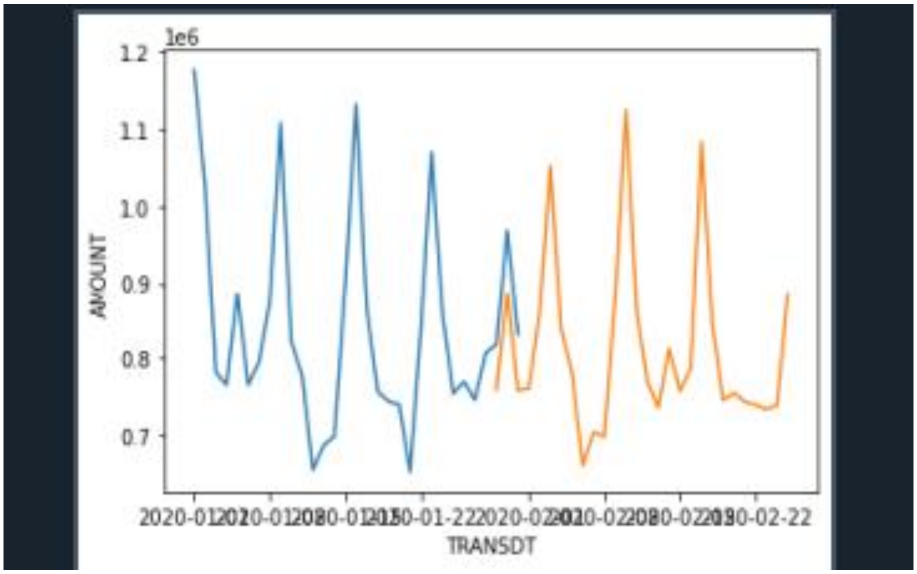
Visualize:



2. **Per day order amount prediction:**

Index	TRANSDT	AMOUNT
0	2020-01-29 00:00:00	759186.06
1	2020-01-30 00:00:00	886000.44
2	2020-01-31 00:00:00	758461.5
3	2020-02-01 00:00:00	760484.4
4	2020-02-02 00:00:00	860115.5
5	2020-02-03 00:00:00	1052632.1
6	2020-02-04 00:00:00	838278.6
7	2020-02-05 00:00:00	781407.7
8	2020-02-06 00:00:00	660712.8

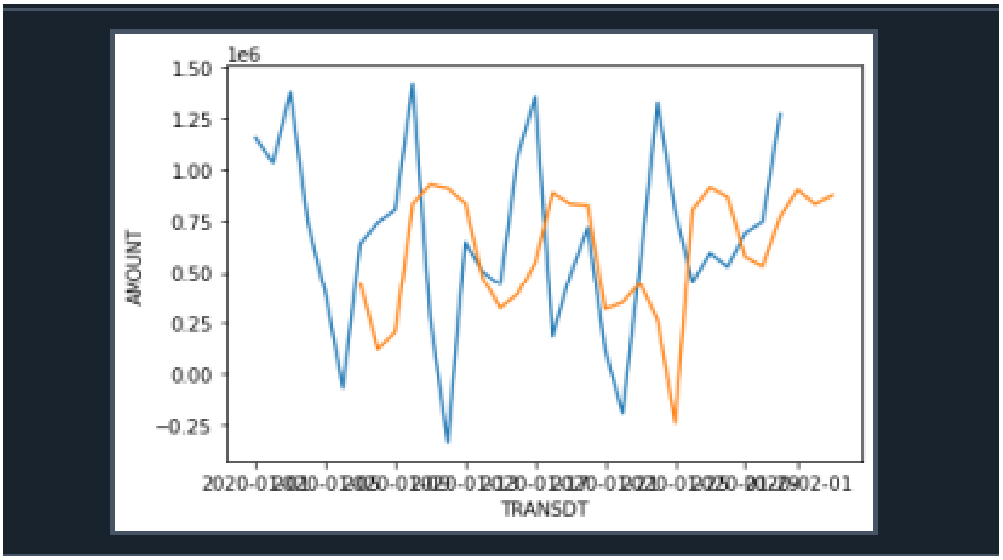
Visualize:



3. Per day profit prediction:

Index	TRANSDT	AMOUNT
0	2020-01-07 00:00:00	444141.97
1	2020-01-08 00:00:00	118016.71
2	2020-01-09 00:00:00	201681.95
3	2020-01-10 00:00:00	834517.9
4	2020-01-11 00:00:00	928307.4
5	2020-01-12 00:00:00	910630.0
6	2020-01-13 00:00:00	836272.44
7	2020-01-14 00:00:00	470576.97
8	2020-01-15 00:00:00	318117.28

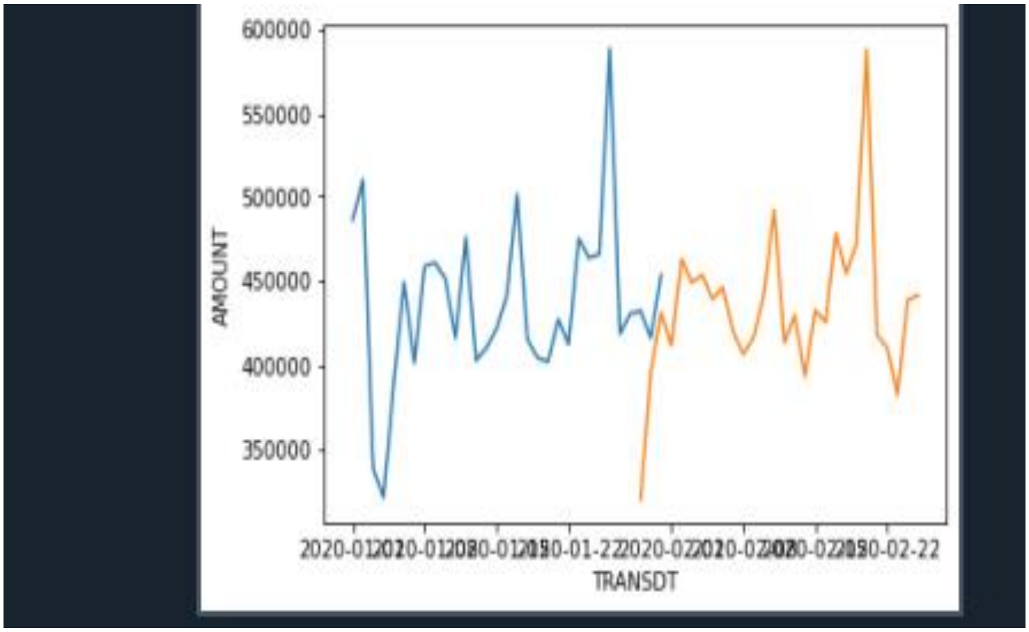
Visualize:



4. Per day Issue amount prediction:

Index	TRANSDT	QTY
0	2020-01-29 00:00:00	13215.493
1	2020-01-30 00:00:00	13984.177
2	2020-01-31 00:00:00	15664.446
3	2020-02-01 00:00:00	17175.6
4	2020-02-02 00:00:00	16325.6455
5	2020-02-03 00:00:00	15262.869
6	2020-02-04 00:00:00	15630.73
7	2020-02-05 00:00:00	15075.101
8	2020-02-06 00:00:00	14143.564

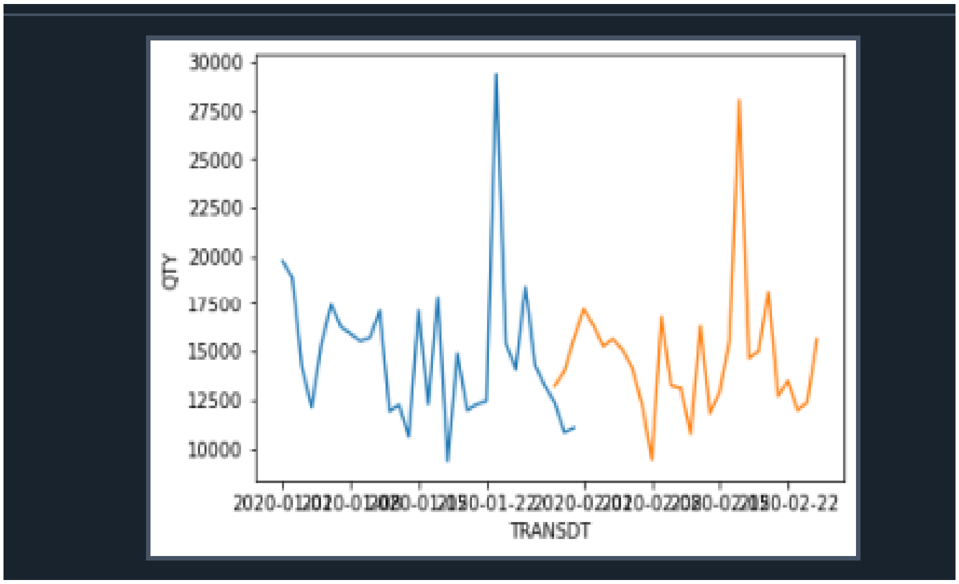
Visualize:



5. **per day Issue qty prediction:**

Index	TRANSDT	QTY
0	2020-01-29 00:00:00	13215.493
1	2020-01-30 00:00:00	13984.177
2	2020-01-31 00:00:00	15664.446
3	2020-02-01 00:00:00	17175.6
4	2020-02-02 00:00:00	16325.6455
5	2020-02-03 00:00:00	15262.869
6	2020-02-04 00:00:00	15630.73
7	2020-02-05 00:00:00	15075.101
8	2020-02-06 00:00:00	14143.564

Visualize:



30000
27500
25000
22500
20000
17500
15000
12500
10000
QTY
TRANSDT
2020-01-22
2020-02-22