Sales Forecasting Using LSTM Report

-by Tejo and Snigda

In order to perform sales forecasting, there are very few techniques, out of which the LSTM model has been proved to perform the best. Long Short-Term Memory (LSTM) is an artificial RNN (Recurrent Neural Network) used in the field of Deep Learning. LSTM networks are well suited to classifying, preprocessing and making predictions based on time-series data. This is because, in the time-series data, there can be lags of unknown duration between important events.

This Deep Learning model has been developed in three phases.

1) Data Cleaning
2) Data Transformation
3) Building the data model and evaluation

Phase 1: Data Cleaning

The diamond dataset of the Jewelry Suite has been taken for sales forecast. Imported this dataset along with the required libraries and converted this csv into pandas dataframe.

As our task is to forecast monthly sales, the data had to be aggregated to the monthly level by summing up the sales column.

In [9]:
```
df_sales
```

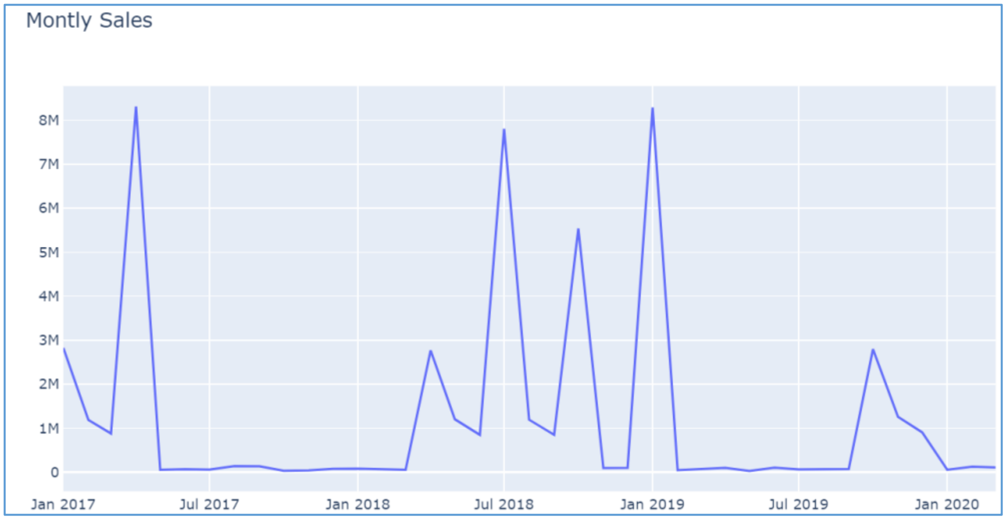
Out[9]:

	Date	Sales amount
0	2017-01-01	2.825349e+06
1	2017-02-01	1.186427e+06
2	2017-03-01	8.781764e+05
3	2017-04-01	8.314381e+06
4	2017-05-01	5.370173e+04
5	2017-06-01	6.766301e+04
6	2017-07-01	5.786559e+04
7	2017-08-01	1.384289e+05
8	2017-09-01	1.340508e+05
9	2017-10-01	2.992774e+04
10	2017-11-01	4.010043e+04
11	2017-12-01	7.408933e+04
12	2018-01-01	7.887067e+04
13	2018-02-01	7.157332e+04
14	2018-03-01	5.218237e+04

Phase 2: Data Transformation

To model the forecast easier and more accurate, the following transformations have been done:

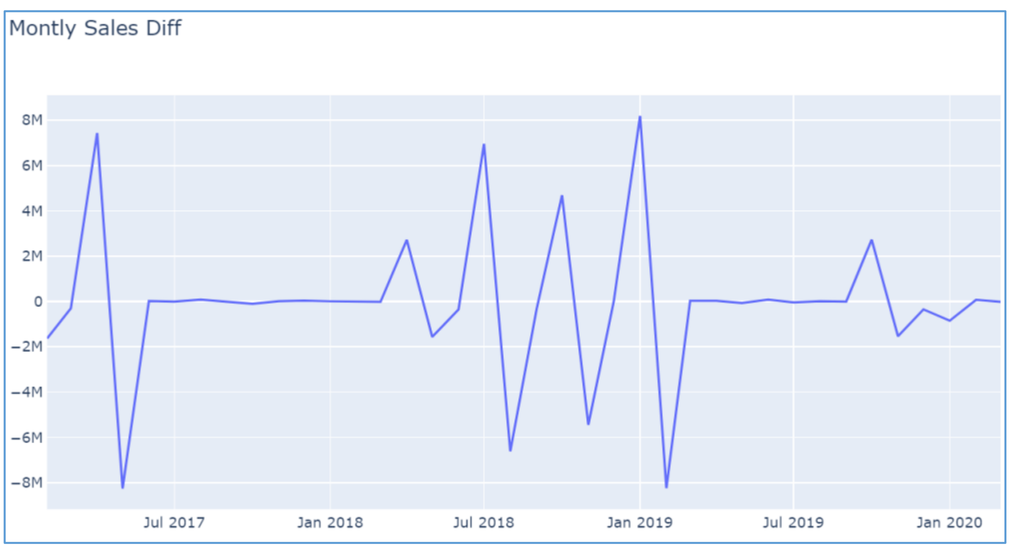
a. Checked if the data is stationary.



It is not stationary and has no particular trend over the months.

b. Conversion from time series to supervised

In order to model our forecast easy and accurate, the data has to be converted to stationary, so got the difference in sales compared to previous month and built the model on it.



Now, this converted data is suitable for building a feature set.

Previous monthly sales data would be used to forecast the next ones. The look back period to be considered will be 12 for this model.

Next step is to create columns from lag_1 to lag_12 and assign values. The values were assigned using the shift() method I python.

In [19]:
```
df_supervised.head()
```

Out[19]:

	Date	Sales amount	diff	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	lag_8
0	2018-02-01	7.157332e+04	-7.297354e+03	4.781345e+03	3.398889e+04	10172.6991	-104123.0513	-4378.1111	80563.3108	-9797.4239	13961.2768
1	2018-03-01	5.218237e+04	-1.939094e+04	-7.297354e+03	4.781345e+03	33988.8910	10172.6991	-104123.0513	-4378.1111	80563.3108	-9797.4239
2	2018-04-01	2.767992e+06	2.715809e+06	-1.939094e+04	-7.297354e+03	4781.3446	33988.8910	10172.6991	-104123.0513	-4378.1111	80563.3108
3	2018-05-01	1.202955e+06	-1.565037e+06	2.715809e+06	-1.939094e+04	-7297.3537	4781.3446	33988.8910	10172.6991	-104123.0513	-4378.1111
4	2018-06-01	8.480074e+05	-3.549473e+05	-1.565037e+06	2.715809e+06	-19390.9419	-7297.3537	4781.3446	33988.8910	10172.6991	-104123.0513

This is our feature set.

By trying to implement the model using different combinations and number of lags with R-Squared as the loss function, combination of lag1 and lag2 gave the best score.

c. Feature Scaling

Took Min Max Scaler for the purpose of scaling each feature between -1 and 1.

Phase 3: Building the LSTM model and evaluation

Now, created feature and label sets from the scaled datasets. Then, fit the LSTM model with one dense layer, ‘mean squared error’ as the loss function and adam optimizer. Ran this model through 96 epochs to get the best accuracy with least loss possible.

Achieved an accuracy of 87% (loss 13%) for this model.

```
In [35]: model = Sequential()
         model.add(LSTM(4, batch_input_shape=(1, X_train.shape[1], X_train.shape[2]), stateful=True))
         model.add(Dense(1))
         model.compile(loss='mean_squared_error', optimizer='adam')
         model.fit(X_train, y_train, nb_epoch=96, batch_size=1, verbose=1, shuffle=False)

         Epoch 88/96
         20/20 [==============================] - 0s 2ms/step - loss: 0.1329
         Epoch 89/96
         20/20 [==============================] - 0s 3ms/step - loss: 0.1325
         Epoch 90/96
         20/20 [==============================] - 0s 2ms/step - loss: 0.1322
         Epoch 91/96
         20/20 [==============================] - 0s 2ms/step - loss: 0.1313
         Epoch 92/96
         20/20 [==============================] - 0s 2ms/step - loss: 0.1303
         Epoch 93/96
         20/20 [==============================] - 0s 2ms/step - loss: 0.1302
         Epoch 94/96
         20/20 [==============================] - 0s 3ms/step - loss: 0.1307
         Epoch 95/96
         20/20 [==============================] - 0s 2ms/step - loss: 0.1309
         Epoch 96/96
         20/20 [==============================] - 0s 2ms/step - loss: 0.1307

Out[35]: <keras.callbacks.History at 0x25fe3437be0>
```

We can see that the loss is approx 13%, which implies that the accuracy of our model is approx 87% which is good but just not enough.

Performed prediction and the results were:

```
In [36]: y_pred = model.predict(X_test,batch_size=1)
         #for multistep prediction, you need to replace X_test values with the predictions coming from t-1

In [37]: y_pred

Out[37]: array([[ 0.23077591],
                [-0.03210291],
                [-0.02857352],
                [ 0.05579389],
                [ 0.11870001],
                [ 0.13360031]], dtype=float32)

In [38]: y_test

Out[38]: array([[ 0.33468832],
                [-0.18401758],
                [-0.03978432],
                [-0.10027156],
                [ 0.01166307],
                [ 0.00098872]])
```

This is on the scaled data so this is not the actual prediction. So, performed inverse transformation for the scaling and built a dataframe that has only the dates and the predictions. The predicted sales after de-transforming is below.

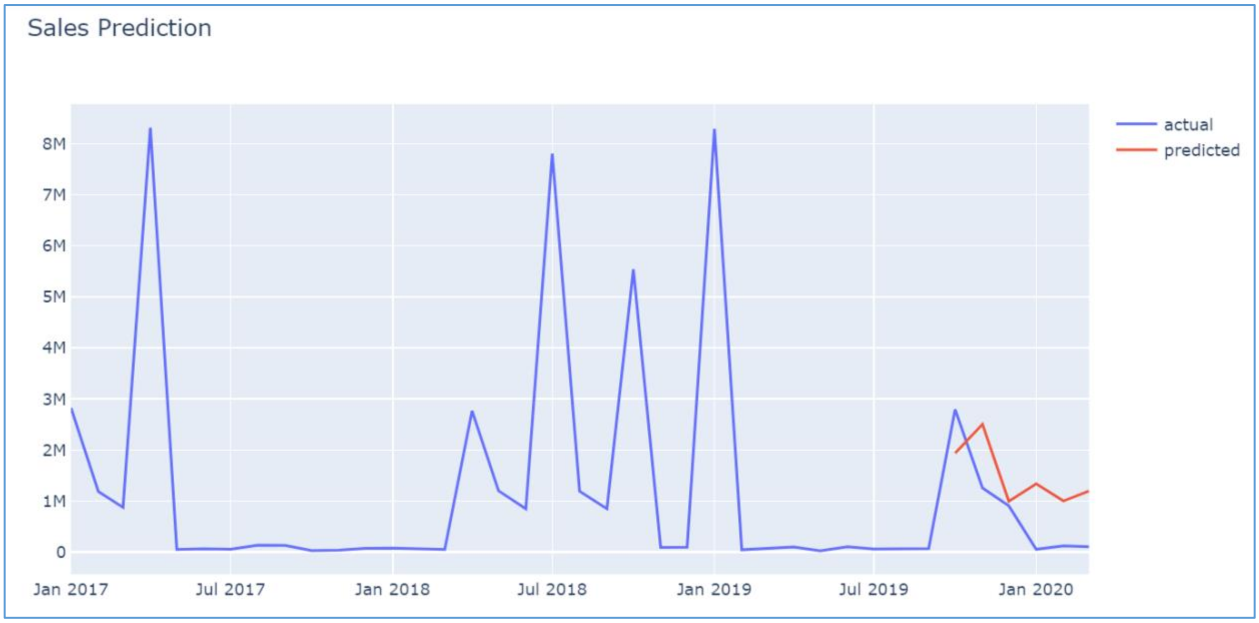
```
In [46]: df_result

Out[46]:
```

	Date	pred_value
0	2019-10-01	1941684
1	2019-11-01	2505932
2	2019-12-01	997164
3	2020-01-01	1337705
4	2020-02-01	1004912
5	2020-03-01	1197393

So, here prediction has been done for the next six months.

Now, checking them in the plot to see how good the model is:



Achieved the best possible fit with the sales data present.

Note: This model can be further improvised to get more accurate results. To achieve this, sales data should be at least 1000 times more than the present sales data.