-----BIG DATA ANALYTICS ASSIGNMENT-----

Kriti Rastogi- AIML B1- 20119051623

Q. Write a program to implement stock market prediction using python. Also explore the steps used in this application.

IMPORTING MODULES AND READING THE DATASET

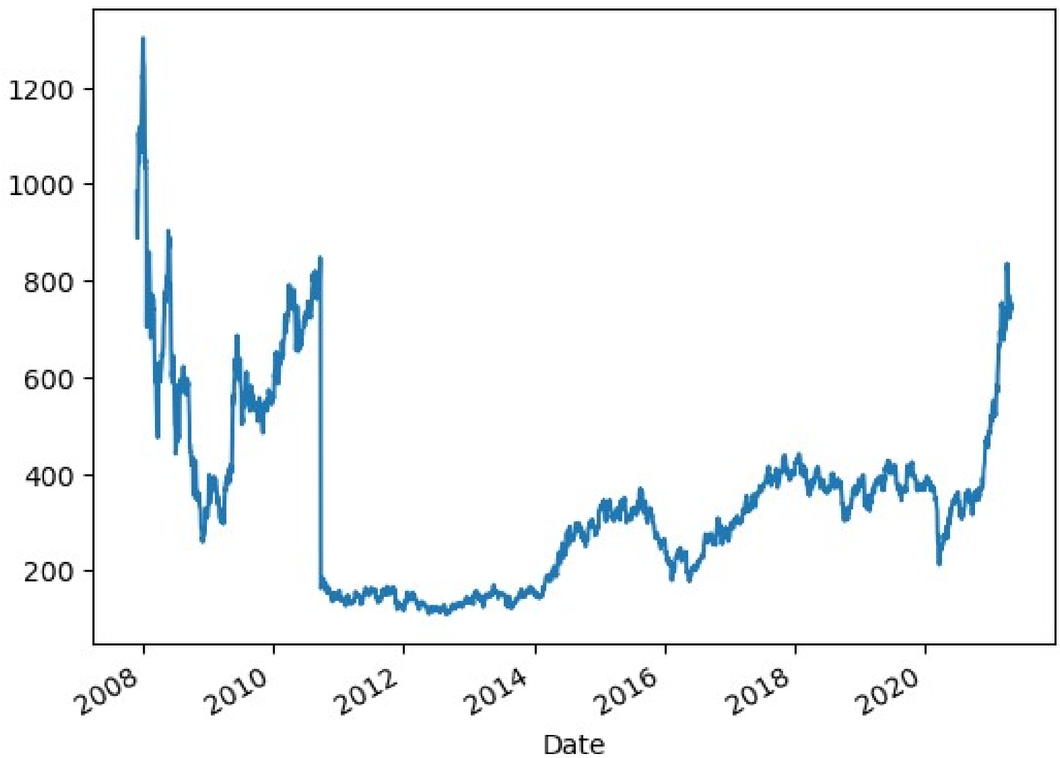
```
import numpy as np
import pandas as pd

df=pd.read_csv('ADANIPORTS.csv',na_values=['null'],index_col='Date',pa
rse_dates=True,infer_datetime_format=True)
df.head()
```

```
C:\Users\Kriti Rastogi\AppData\Local\Temp\
ipykernel_28044\1324389878.py:4: FutureWarning: The argument
'infer_datetime_format' is deprecated and will be removed in a future
version. A strict version of it is now the default, see
https://pandas.pydata.org/pdeps/0004-consistent-to-datetime-
parsing.html. You can safely remove this argument.

df=pd.read_csv('ADANIPORTS.csv',na_values=['null'],index_col='Date',pa
rse_dates=True,infer_datetime_format=True)

                Symbol Series  Prev Close    Open     High    Low
Last  \
Date

2007-11-27  MUNDRAPORT     EQ      440.00  770.00  1050.00  770.0
959.0
2007-11-28  MUNDRAPORT     EQ      962.90  984.00   990.00  874.0
885.0
2007-11-29  MUNDRAPORT     EQ      893.90  909.00   914.75  841.0
887.0
2007-11-30  MUNDRAPORT     EQ      884.20  890.00   958.00  890.0
929.0
2007-12-03  MUNDRAPORT     EQ      921.55  939.75   995.00  922.0
980.0

             Close    VWAP    Volume     Turnover  Trades  \
Date
2007-11-27  962.90  984.72  27294366  2.687719e+15     NaN
2007-11-28  893.90  941.38   4581338  4.312765e+14     NaN
2007-11-29  884.20  888.09   5124121  4.550658e+14     NaN
2007-11-30  921.55  929.17   4609762  4.283257e+14     NaN
2007-12-03  969.30  965.65   2977470  2.875200e+14     NaN

            Deliverable Volume  %Deliverble
Date
```

```
2007-11-27              9859619       0.3612
2007-11-28              1453278       0.3172
2007-11-29              1069678       0.2088
2007-11-30              1260913       0.2735
2007-12-03               816123       0.2741
```

DRAWING THE PLOT

```
df['VWAP'].plot()

<Axes: xlabel='Date'>
```



CREATING A DATAFRAME

```
output_var=pd.DataFrame(df['VWAP'])
features=['Open','High','Low','Volume']
```

NORMALISING THE DATASET'S FEATURE VALUES INTO A SPECIFIC RANGE

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
feature_transform=scaler.fit_transform(df[features])
```

```
feature_transform=pd.DataFrame(columns=features,data=feature_transform
,index=df.index)
feature_transform.head()
```

	Open	High	Low	Volume
Date				
2007-11-27	0.550634	0.774216	0.570576	0.279227
2007-11-28	0.728634	0.724774	0.659896	0.046763
2007-11-29	0.666251	0.662766	0.631554	0.052318
2007-11-30	0.650447	0.698406	0.673638	0.047054
2007-12-03	0.691828	0.728895	0.701121	0.030347

```
feature_transform.shape
```

```
(3322, 4)
```

PERFORMING TIME SERIES CROSS VALIDATION

```
from sklearn.model_selection import TimeSeriesSplit
timesplit=TimeSeriesSplit(n_splits=10)
for train_index,test_index in timesplit.split(feature_transform):

X_train,X_test=feature_transform[:len(train_index)],feature_transform[
len(train_index):(len(train_index)+len(test_index))]

y_train,y_test=output_var[:len(train_index)].values.ravel(),output_var
[len(train_index):(len(train_index)+len(test_index))].values.ravel()

print(X_train.shape)
print(X_test.shape)
```

```
(3020, 4)
(302, 4)
```

PREPARING DATA FOR SEQUENCE BASED DL MODELS (LSTM HERE)

```
trainX=np.array(X_train)
testX=np.array(X_test)
X_train=trainX.reshape(X_train.shape[0],1,X_train.shape[1])
X_test=testX.reshape(X_test.shape[0],1,X_test.shape[1])
print(X_train.shape)
print(X_test.shape)
```

```
(3020, 1, 4)
(302, 1, 4)
```

IMPLEMENTING AN LSTM REGRESSION MODEL

```
import tensorflow as tf
```

```
lstm=tf.keras.Sequential()
lstm.add(tf.keras.layers.LSTM(32,input_shape=(1,trainX.shape[1]),activ
ation='relu',return_sequences=False))
lstm.add(tf.keras.layers.Dense(1))
lstm.compile(loss='mean_squared_error',optimizer='adam')
tf.keras.utils.plot_model(lstm,show_shapes=True,show_layer_names=True)
```

```
You must install graphviz (see instructions at
https://graphviz.gitlab.io/download/) for `plot_model` to work.
```

```
C:\Users\Kriti Rastogi\AppData\Roaming\Python\Python312\site-packages\
keras\src\layers\rnn\rnn.py:199: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)
```

```
history=lstm.fit(X_train,y_train,epochs=100,batch_size=8,verbose=1,shu
ffle=False)
```

```
Epoch 1/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - loss: 151030.6875
Epoch 2/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 145412.7031
Epoch 3/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 135326.4375
Epoch 4/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 121324.6406
Epoch 5/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 105430.6016
Epoch 6/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 89206.4297
Epoch 7/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 73719.7109
Epoch 8/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 59689.9180
Epoch 9/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 47561.0703
Epoch 10/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 37539.5898
Epoch 11/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 29623.6719
Epoch 12/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 23638.1699
Epoch 13/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 19279.6934
Epoch 14/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 16172.3506
Epoch 15/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 13928.4648
```

```
Epoch 16/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 12204.7988
Epoch 17/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 10740.2910
Epoch 18/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 9368.3818
Epoch 19/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 8012.3247
Epoch 20/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 6669.9429
Epoch 21/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 5384.9736
Epoch 22/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 4213.4468
Epoch 23/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 3200.3811
Epoch 24/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 2370.6843
Epoch 25/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 1727.7900
Epoch 26/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 1255.8876
Epoch 27/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 925.0824
Epoch 28/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 699.2754
Epoch 29/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 544.7745
Epoch 30/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 435.9872
Epoch 31/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 356.2154
Epoch 32/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 295.4507
Epoch 33/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 247.9344
Epoch 34/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 210.3154
Epoch 35/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 180.4693
Epoch 36/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 156.8597
Epoch 37/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 138.2637
Epoch 38/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 123.6805
Epoch 39/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 112.2974
Epoch 40/100
```

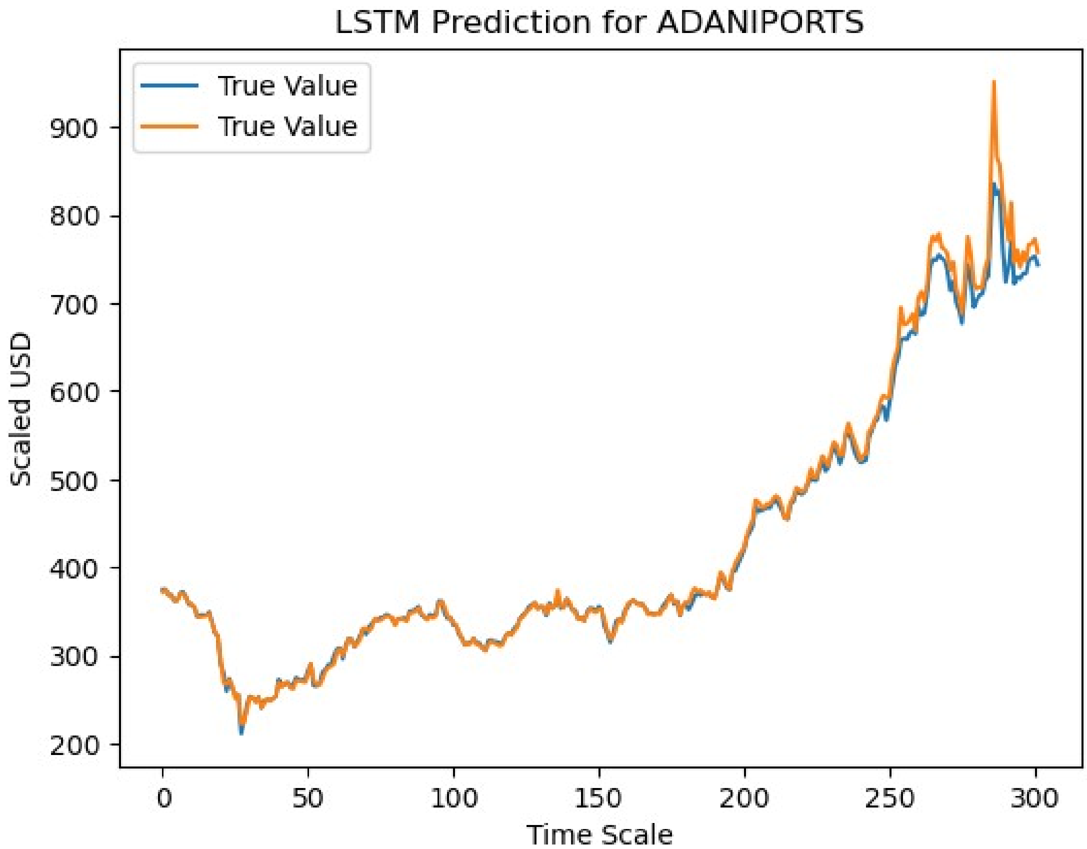
```
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 103.4742
Epoch 41/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 96.7057
Epoch 42/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 91.5917
Epoch 43/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 87.8036
Epoch 44/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 85.0644
Epoch 45/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 83.1375
Epoch 46/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 81.8210
Epoch 47/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 80.9480
Epoch 48/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 80.3831
Epoch 49/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 80.0221
Epoch 50/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 79.7846
Epoch 51/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 79.6124
Epoch 52/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 79.4639
Epoch 53/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 79.3101
Epoch 54/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 79.1317
Epoch 55/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 78.9161
Epoch 56/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 78.6561
Epoch 57/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 78.3467
Epoch 58/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 77.9856
Epoch 59/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 77.5712
Epoch 60/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 77.1028
Epoch 61/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 76.5797
Epoch 62/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 76.0015
Epoch 63/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 75.3676
Epoch 64/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 74.6776
```

```
Epoch 65/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 73.9305
Epoch 66/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 73.1258
Epoch 67/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 72.2628
Epoch 68/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 71.3407
Epoch 69/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 70.3597
Epoch 70/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 69.3196
Epoch 71/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 68.2212
Epoch 72/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 67.0653
Epoch 73/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 65.8535
Epoch 74/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 64.5882
Epoch 75/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 63.2725
Epoch 76/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 61.9099
Epoch 77/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 60.5047
Epoch 78/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 59.0621
Epoch 79/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 57.5875
Epoch 80/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 56.0871
Epoch 81/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 54.5673
Epoch 82/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 53.0353
Epoch 83/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 51.4978
Epoch 84/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 49.9624
Epoch 85/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 48.4360
Epoch 86/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 46.9257
Epoch 87/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 45.4385
Epoch 88/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 43.9810
Epoch 89/100
```

```
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 42.5591
Epoch 90/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 41.1786
Epoch 91/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 39.8444
Epoch 92/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 38.5611
Epoch 93/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 37.3325
Epoch 94/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 36.1618
Epoch 95/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 35.0515
Epoch 96/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 34.0036
Epoch 97/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 33.0191
Epoch 98/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 32.0989
Epoch 99/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 31.2427
Epoch 100/100
378/378 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 30.4503

y_pred=lstm.predict(X_test)

10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
```

PREPARING THE FINAL PLOT

```
import matplotlib.pyplot as plt
plt.plot(y_test,label='True Value')
plt.plot(y_pred,label='True Value')
plt.title("LSTM Prediction for ADANIPORTS")
plt.xlabel('Time Scale')
plt.ylabel('Scaled USD')
plt.legend()
plt.show()
```


LSTM Prediction for ADANIPORTS
True Value
True Value
Scaled USD
Time Scale
900
800
700
600
500
400
300
200
0
50
100
150
200
250
300

-----END OF CODE-----