```
#For getting a Realtime Data From the Yahoo Finance
!pip install yfinance
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting yfinance
  Downloading yfinance-0.2.11-py2.py3-none-any.whl (59 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.2/59.2 KB 2.5 MB/s eta 0:00:00
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.8/dist-packages (from yfinance) (2022.7.1)
Collecting requests>=2.26
  Downloading requests-2.28.2-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.8/62.8 KB 5.3 MB/s eta 0:00:00
Collecting frozendict>=2.3.4
  Downloading frozendict-2.3.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (110 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 111.0/111.0 KB 11.5 MB/s eta 0:00:00
Collecting html5lib>=1.1
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.2/112.2 KB 11.9 MB/s eta 0:00:00
Requirement already satisfied: lxml>=4.9.1 in /usr/local/lib/python3.8/dist-packages (from yfinance) (4.9.2)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.21.6)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.8/dist-packages (from yfinance) (0.0.11)
Collecting cryptography>=3.3.2
  Downloading cryptography-39.0.1-cp36-abi3-manylinux_2_28_x86_64.whl (4.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.2/4.2 MB 53.5 MB/s eta 0:00:00
Collecting beautifulsoup4>=4.11.1
  Downloading beautifulsoup4-4.11.2-py3-none-any.whl (129 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.4/129.4 KB 13.1 MB/s eta 0:00:00
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.3.5)
Requirement already satisfied: appdirs>=1.4.4 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.4.4)
Collecting soupsieve>1.2
  Downloading soupsieve-2.4-py3-none-any.whl (37 kB)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.8/dist-packages (from cryptography>=3.3.2->yfinance) (1.15.1)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.8/dist-packages (from html5lib>=1.1->yfinance) (1.15.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.8/dist-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.8/dist-packages (from pandas>=1.3.0->yfinance) (2.8
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (2022.1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (1.2
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (2.10)
Requirement already satisfied: pycparser in /usr/local/lib/python3.8/dist-packages (from cffi>=1.12->cryptography>=3.3.2->yfinance)
Installing collected packages: soupsieve, requests, html5lib, frozendict, cryptography, beautifulsoup4, yfinance
  Attempting uninstall: requests
    Found existing installation: requests 2.25.1
    Uninstalling requests-2.25.1:
      Successfully uninstalled requests-2.25.1
  Attempting uninstall: html5lib
    Found existing installation: html5lib 1.0.1
    Uninstalling html5lib-1.0.1:
      Successfully uninstalled html5lib-1.0.1
  Attempting uninstall: beautifulsoup4
    Found existing installation: beautifulsoup4 4.6.3
    Uninstalling beautifulsoup4-4.6.3:
      Successfully uninstalled beautifulsoup4-4.6.3
Successfully installed beautifulsoup4-4.11.2 cryptography-39.0.1 frozendict-2.3.4 html5lib-1.1 requests-2.28.2 soupsieve-2.4 yfinan
```

```
#To get a Data from Website using YFinance Library
import yfinance as yf

msft = yf.Ticker("SI=F")

# get the historical market data Specific Date
hist = msft.history(start="2020-01-01", end="2023-12-31")
```

```
import pandas as pd
```

```
#To get a CSV File
hist.to_csv('Silver.csv')
```

```
hist.head()
```

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2020-01-02 00:00:00-05:00	17.966000	17.990000	17.966000	17.966000	2	0.0	0.0
2020-01-03 00:00:00-05:00	18.110001	18.110001	17.965000	18.068001	83	0.0	0.0
2020-01-06 00:00:00-05:00	18.025000	18.105000	18.025000	18.097000	3	0.0	0.0
2020-01-07 00:00:00-05:00	18.014999	18.344999	18.014999	18.316000	33	0.0	0.0
2020-01-08 00:00:00-05:00	18.400000	18.504999	18.070000	18.087999	31	0.0	0.0

```
df1 = hist.reset_index()['Close']
```

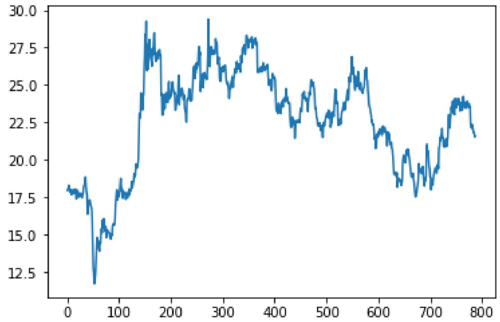
```
df1.shape
```

```
(788,)
```

```
import matplotlib.pyplot as plt
```

```
plt.plot(df1)
```

```
[<matplotlib.lines.Line2D at 0x7f80f9c32520>]
```




```
#LSTM are sensitive to the scale of the data. So we apply MinMaxSCaler
```

```
import numpy as np
```

```
from sklearn.preprocessing import MinMaxScaler

Scaler = MinMaxScaler(feature_range=(0,1))

df1 = Scaler.fit_transform(np.array(df1).reshape(-1,1))
```

```
#Splitting Dataset into Train and Test Split
```

```
Training_Size = int(len(df1)*0.80)
Test_Size = len(df1) - Training_Size
Train_data , Test_data = df1[0:Training_Size,:] , df1[Training_Size:len(df1),:1]
```

```
Training_Size , Test_Size
```

```
(630, 158)
```

```
#Converting An Array Of Value into A Dataset Matrix
import numpy
def create_dataset(dataset, time_step = 1):
    dataX , dataY = [] , []
    for i in range(len(dataset)-time_step-1):
      a = dataset[i:(i+time_step), 0 ]
      dataX.append(a)
      dataY.append(dataset[i + time_step,0])
    return numpy.array(dataX) , numpy.array(dataY)
```

```
#reshape into X=t t+1 t+2 t+3 t+4 and y=t+5
time_step=100
X_Train , Y_Train = create_dataset(Train_data,time_step)
X_Test , Y_Test = create_dataset(Test_data,time_step)
```

```
#Print the 4 timeStep Features Array Value
print(X_Train)
```

```
[[0.35277131 0.35854616 0.36018797 ... 0.31704688 0.33567342 0.33006853]
 [0.35854616 0.36018797 0.37258676 ... 0.33567342 0.33006853 0.33691903]
 [0.36018797 0.37258676 0.35967838 ... 0.33006853 0.33691903 0.34875165]
 ...
 [0.621412   0.60238916 0.60805073 ... 0.52635449 0.53111017 0.53354471]
 [0.60238916 0.60805073 0.64207665 ... 0.53111017 0.53354471 0.51304992]
 [0.60805073 0.64207665 0.64881387 ... 0.53354471 0.51304992 0.50574642]]
```

```
print(X_Train.shape) , print(Y_Train.shape)
```

```
(529, 100)
(529,)
(None, None)
```

```
print(X_Test.shape) , print(Y_Test.shape)
```

```
(57, 100)
(57,)
(None, None)
```

```
#reshape input to  be [Sample , TimeStep ,Features] Which is required for LSTM
X_Train = X_Train.reshape(X_Train.shape[0],X_Train.shape[1],1)
X_Test = X_Test.reshape(X_Test.shape[0],X_Test.shape[1],1)
```

```
#Create a The Stacked LSTM Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

```
model = Sequential()
model.add(LSTM(50,return_sequences = True , input_shape = (100,1)))
model.add(LSTM(50,return_sequences = True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error' , optimizer = 'adam')
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 lstm_1 (LSTM)               (None, 100, 50)           20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(X_Train,Y_Train,validation_data=(X_Test,Y_Test), epochs=100,batch_size=64,verbose=1)
```

```
Epoch 1/100
9/9 [==============================] - 22s 819ms/step - loss: 0.1964 - val_loss: 0.0367
Epoch 2/100
9/9 [==============================] - 2s 195ms/step - loss: 0.0239 - val_loss: 0.0138
Epoch 3/100
9/9 [==============================] - 2s 196ms/step - loss: 0.0120 - val_loss: 0.0134
Epoch 4/100
9/9 [==============================] - 2s 198ms/step - loss: 0.0101 - val_loss: 0.0026
Epoch 5/100
9/9 [==============================] - 2s 286ms/step - loss: 0.0079 - val_loss: 0.0043
Epoch 6/100
9/9 [==============================] - 2s 248ms/step - loss: 0.0070 - val_loss: 0.0024
Epoch 7/100
9/9 [==============================] - 2s 197ms/step - loss: 0.0067 - val_loss: 0.0030
Epoch 8/100
9/9 [==============================] - 2s 197ms/step - loss: 0.0063 - val_loss: 0.0026
Epoch 9/100
9/9 [==============================] - 2s 195ms/step - loss: 0.0061 - val_loss: 0.0029
Epoch 10/100
9/9 [==============================] - 2s 200ms/step - loss: 0.0060 - val_loss: 0.0026
Epoch 11/100
9/9 [==============================] - 2s 200ms/step - loss: 0.0059 - val_loss: 0.0027
Epoch 12/100
9/9 [==============================] - 3s 333ms/step - loss: 0.0057 - val_loss: 0.0028
Epoch 13/100
9/9 [==============================] - 2s 208ms/step - loss: 0.0056 - val_loss: 0.0025
Epoch 14/100
9/9 [==============================] - 2s 201ms/step - loss: 0.0055 - val_loss: 0.0027
Epoch 15/100
9/9 [==============================] - 2s 199ms/step - loss: 0.0055 - val_loss: 0.0028
Epoch 16/100
9/9 [==============================] - 2s 198ms/step - loss: 0.0053 - val_loss: 0.0023
Epoch 17/100
9/9 [==============================] - 2s 200ms/step - loss: 0.0053 - val_loss: 0.0034
Epoch 18/100
```

```
9/9 [==============================] - 2s 254ms/step - loss: 0.0054 - val_loss: 0.0023
Epoch 19/100
9/9 [==============================] - 3s 285ms/step - loss: 0.0050 - val_loss: 0.0023
Epoch 20/100
9/9 [==============================] - 2s 198ms/step - loss: 0.0050 - val_loss: 0.0023
Epoch 21/100
9/9 [==============================] - 2s 200ms/step - loss: 0.0047 - val_loss: 0.0028
Epoch 22/100
9/9 [==============================] - 2s 201ms/step - loss: 0.0048 - val_loss: 0.0024
Epoch 23/100
9/9 [==============================] - 2s 201ms/step - loss: 0.0045 - val_loss: 0.0022
Epoch 24/100
9/9 [==============================] - 2s 204ms/step - loss: 0.0044 - val_loss: 0.0025
Epoch 25/100
9/9 [==============================] - 3s 327ms/step - loss: 0.0042 - val_loss: 0.0020
Epoch 26/100
9/9 [==============================] - 2s 220ms/step - loss: 0.0042 - val_loss: 0.0021
Epoch 27/100
9/9 [==============================] - 2s 200ms/step - loss: 0.0046 - val_loss: 0.0029
Epoch 28/100
9/9 [==============================] - 2s 200ms/step - loss: 0.0051 - val_loss: 0.0024
Epoch 29/100
9/9 [==============================] - 2s 206ms/step - loss: 0.0042 - val_loss: 0.0020
```

```
import tensorflow as tf
```

```
#let do the prediction and check the performance metrics
Train_Predict = model.predict(X_Train)
Test_Predict = model.predict(X_Test)
```

```
17/17 [==============================] - 3s 56ms/step
2/2 [==============================] - 0s 54ms/step
```

```
#Transform to original form
Train_Predict = Scaler.inverse_transform(Train_Predict)
Test_Predict = Scaler.inverse_transform(Test_Predict)
```

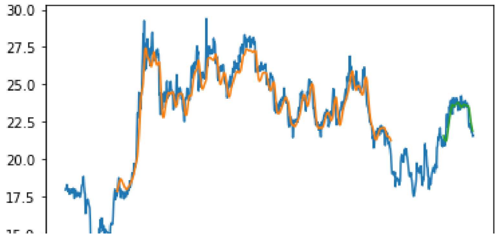
```
#Calculate RMSE Performance Metric
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(Y_Train,Train_Predict))
```

```
23.383009014321456
```

```
#Test Data RMSE
math.sqrt(mean_squared_error(Y_Test,Test_Predict))
```

```
22.398532006983164
```

```
#Plotting

#shift train predictions for plotting

look_back=100

trainPredictPlot = numpy.empty_like(df1)

trainPredictPlot[:, :] = np.nan # plot baseline and predictions

trainPredictPlot[look_back: len(Train_Predict)+look_back, ] = Train_Predict

#shift test predictions for plotting

testPredictPlot = numpy.empty_like(df1)
testPredictPlot[:, :]= numpy.nan
testPredictPlot [len(Train_Predict)+(look_back*2)+1:len (df1)-1,] = Test_Predict

plt.plot(Scaler.inverse_transform(df1))

plt.plot(trainPredictPlot)

plt.plot(testPredictPlot)

plt.show()
#Orange is Train Predict Data
#Green is Test Predict Data
```




```
len(Test_data)
```

```
158
```

```
X_Input = Test_data[58:].reshape(1,-1)
```

```
X_Input.shape
```

```
(1, 100)
```

```
temp_input = list(X_Input)
temp_input = temp_input[0].tolist()
```

```
lst_output=[]

n_steps=100
i = 0

while (i < 30) :

  if(len(temp_input)>100):

    #print(temp_input)

    x_input=np.array(temp_input[1:])

    print("{} day input {}".format(i,x_input))

    x_input=x_input.reshape(1,-1)

    x_input = x_input.reshape((1, n_steps, 1))

    #print(x_input)

    yhat = model.predict(x_input, verbose=0)

    print("{} day output {}".format(i,yhat))

    temp_input.extend(yhat[0].tolist())

    temp_input=temp_input[1:]

    #print(temp_input)

    lst_output.extend(yhat.tolist())
    i = i + 1
  else:
    X_Input = X_Input.reshape((1, n_steps,1))
    yhat = model.predict(X_Input, verbose=0)
    print(yhat[0])
    temp_input.extend(yhat[0].tolist())
    print(len(temp_input))
    lst_output.extend(yhat.tolist())
    i = i + 1

print(lst_output)
```

```
[0.5647845]
101
1 day input [0.36924644 0.39976218 0.38928834 0.4090471  0.4973107  0.52663763
 0.49504614 0.50161349 0.47879744 0.44278999 0.43548661 0.4044047
 0.40327236 0.35554545 0.3928551  0.38702367 0.3737191  0.39240221
 0.4137462  0.42116286 0.43050448 0.4384872  0.43837403 0.41963422
 0.41838872 0.44941404 0.44528111 0.43599619 0.51265361 0.51876808
 0.55154838 0.54141426 0.56287153 0.56111644 0.58636691 0.55279399
 0.55307702 0.52199511 0.52318404 0.51599393 0.52646777 0.54475458
 0.54860437 0.51944741 0.53609241 0.55568131 0.61722238 0.63992525
 0.59242479 0.588122   0.62135531 0.64134062 0.66806315 0.65022923
 0.68312292 0.69133209 0.64502066 0.64632275 0.63941567 0.69971123
 0.69637091 0.66472284 0.68102811 0.69665405 0.67530995 0.69880534
 0.68657641 0.6977297  0.68261334 0.65238063 0.68431185 0.6780841
```

```
 0.66647793 0.65628711 0.6866331  0.70746762 0.69121892 0.66778012
 0.68046194 0.68476474 0.66336406 0.67451734 0.68527432 0.69042631
 0.66761026 0.6742909  0.68012222 0.66727054 0.66800657 0.6000679
 0.59117929 0.58778239 0.60193627 0.58665005 0.58308329 0.57085436
 0.57221303 0.55545487 0.55681365 0.56478453]
1 day output [[0.56181073]]
2 day input [0.39976218 0.38928834 0.4090471  0.4973107  0.52663763 0.49504614
 0.50161349 0.47879744 0.44278999 0.43548661 0.4044047  0.40327236
 0.35554545 0.3928551  0.38702367 0.3737191  0.39240221 0.4137462
 0.42116286 0.43050448 0.4384872  0.43837403 0.41963422 0.41838872
 0.44941404 0.44528111 0.43599619 0.51265361 0.51876808 0.55154838
 0.54141426 0.56287153 0.56111644 0.58636691 0.55279399 0.55307702
 0.52199511 0.52318404 0.51599393 0.52646777 0.54475458 0.54860437
 0.51944741 0.53609241 0.55568131 0.61722238 0.63992525 0.59242479
 0.588122   0.62135531 0.64134062 0.66806315 0.65022923 0.68312292
 0.69133209 0.64502066 0.64632275 0.63941567 0.69971123 0.69637091
 0.66472284 0.68102811 0.69665405 0.67530995 0.69880534 0.68657641
 0.6977297  0.68261334 0.65238063 0.68431185 0.6780841  0.66647793
 0.65628711 0.6866331  0.70746762 0.69121892 0.66778012 0.68046194
 0.68476474 0.66336406 0.67451734 0.68527432 0.69042631 0.66761026
 0.6742909  0.68012222 0.66727054 0.66800657 0.6000679  0.59117929
 0.58778239 0.60193627 0.58665005 0.58308329 0.57085436 0.57221303
 0.55545487 0.55681365 0.56478453 0.56181073]
2 day output [[0.56032264]]
3 day input [0.38928834 0.4090471  0.4973107  0.52663763 0.49504614 0.50161349
 0.47879744 0.44278999 0.43548661 0.4044047  0.40327236 0.35554545
 0.3928551  0.38702367 0.3737191  0.39240221 0.4137462  0.42116286
 0.43050448 0.4384872  0.43837403 0.41963422 0.41838872 0.44941404
 0.44528111 0.43599619 0.51265361 0.51876808 0.55154838 0.54141426
 0.56287153 0.56111644 0.58636691 0.55279399 0.55307702 0.52199511
 0.52318404 0.51599393 0.52646777 0.54475458 0.54860437 0.51944741
 0.53609241 0.55568131 0.61722238 0.63992525 0.59242479 0.588122
 0.62135531 0.64134062 0.66806315 0.65022923 0.68312292 0.69133209
 0.64502066 0.64632275 0.63941567 0.69971123 0.69637091 0.66472284
 0.68102811 0.69665405 0.67530995 0.69880534 0.68657641 0.6977297
 0.68261334 0.65238063 0.68431185 0.6780841  0.66647793 0.65628711
 0.6866331  0.70746762 0.69121892 0.66778012 0.68046194 0.68476474
 0.66336406 0.67451734 0.68527432 0.69042631 0.66761026 0.6742909
 0.68012222 0.66727054 0.66800657 0.6000679  0.59117929 0.58778239
 0.60193627 0.58665005 0.58308329 0.57085436 0.57221303 0.55545487
 0.55681365 0.56478453 0.56181073 0.56032264]
3 day output [[0.55994261]]
```

```
Day_New = np.arange(1,101)
Day_Pred = np.arange(101,131)
```

```
import matplotlib.pyplot as plt
```

```
len(df1)
```

```
788
```

```
df3 = df1.tolist()
df3.extend(lst_output)
```

```
plt.plot(Day_New,Scaler.inverse_transform(df1[688:]))
plt.plot(Day_Pred,Scaler.inverse_transform(lst_output))
```

```
[<matplotlib.lines.Line2D at 0x7f809041ae20>]
```