

# aapl prediction

June 4, 2023

```
[48]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
[2]: aapl=pd.read_csv("C:/Users/jakey/OneDrive/Documents/personal projects/
↳predicting AAPL stock price\AAPL.csv",index_col='Date')
aapl
#10409 days
```

```
[2]:
```

	Open	High	Low	Close	Adj Close	\
Date						
1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100323	
1980-12-15	0.122210	0.122210	0.121652	0.121652	0.095089	
1980-12-16	0.113281	0.113281	0.112723	0.112723	0.088110	
1980-12-17	0.115513	0.116071	0.115513	0.115513	0.090291	
1980-12-18	0.118862	0.119420	0.118862	0.118862	0.092908	
...	...	...	...	...	...	
2022-03-18	160.509995	164.479996	159.759995	163.979996	163.979996	
2022-03-21	163.509995	166.350006	163.009995	165.380005	165.380005	
2022-03-22	165.509995	169.419998	164.910004	168.820007	168.820007	
2022-03-23	167.990005	172.639999	167.649994	170.210007	170.210007	
2022-03-24	171.059998	174.139999	170.210007	174.070007	174.070007	
	Volume					
Date						
1980-12-12	469033600					
1980-12-15	175884800					
1980-12-16	105728000					
1980-12-17	86441600					
1980-12-18	73449600					
...	...					
2022-03-18	123351200					

```

2022-03-21    95811400
2022-03-22    81532000
2022-03-23    98062700
2022-03-24    90018700

```

```
[10409 rows x 6 columns]
```

```
[3]: aapl.describe()
```

```

[3]:
      count      Open      High      Low      Close      Adj Close  \
count  10409.000000  10409.000000  10409.000000  10409.000000  10409.000000
mean     13.959910     14.111936     13.809163     13.966757     13.350337
std      30.169244     30.514878     29.835055     30.191696     29.911132
min       0.049665      0.049665      0.049107      0.049107      0.038384
25%       0.281964      0.287946      0.274554      0.281250      0.234799
50%       0.468750      0.477679      0.459821      0.468750      0.386853
75%      14.217857     14.364286     14.043571     14.206071     12.188149
max      182.630005     182.940002     179.119995     182.009995     181.778397

      Volume
count  1.040900e+04
mean   3.321778e+08
std    3.393344e+08
min    0.000000e+00
25%    1.247604e+08
50%    2.199680e+08
75%    4.126108e+08
max    7.421641e+09

```

```

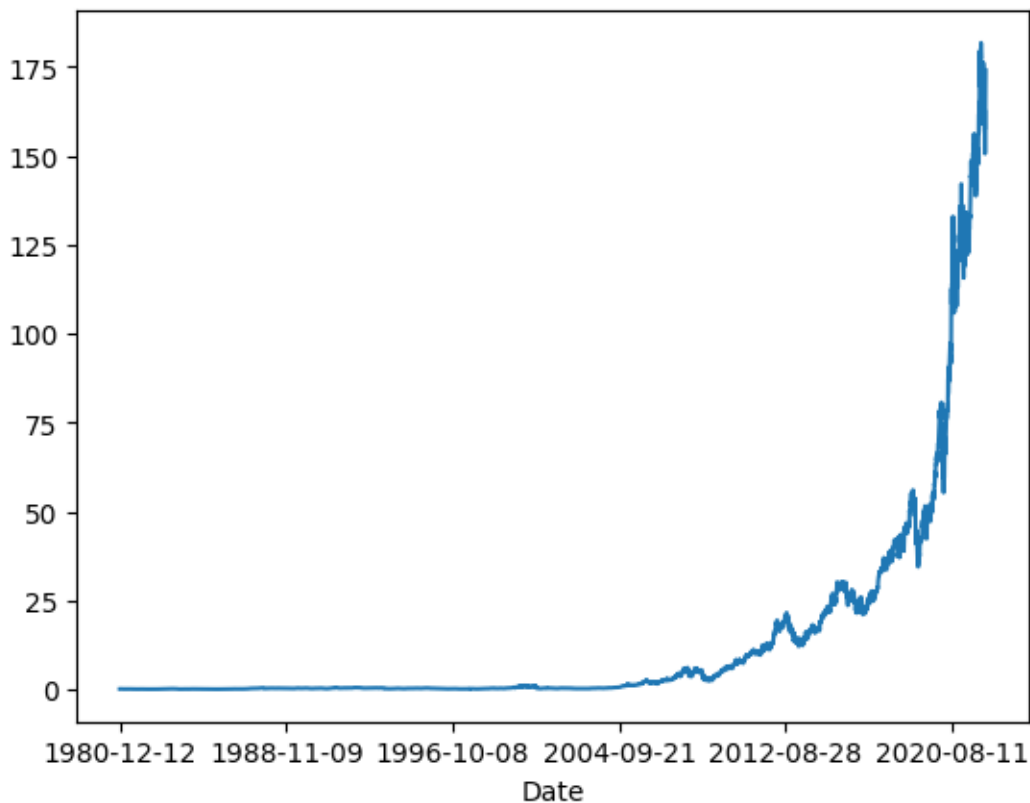
[4]: #Check for any missing values values
aapl.isnull().values.any()

```

```
[4]: False
```

```
[5]: aapl['Adj Close'].plot()
```

```
[5]: <Axes: xlabel='Date'>
```



```
[6]: #target variable=dependent variable
target=pd.DataFrame(aapl['Adj Close'])
#feature variables=independent variables
features=['Open','High','Low','Volume']
#Scaling data between 0 and 1 for precision and memory consumption
scaler=MinMaxScaler(feature_range=(0,1))
features_transform=scaler.fit_transform(aapl[features])
feature_transform=pd.
    ↪DataFrame(columns=features,data=features_transform,index=aapl.index)
feature_transform
```

```
[6]:
```

	Open	High	Low	Volume
Date				
1980-12-12	0.000431	0.000433	0.000443	0.063198
1980-12-15	0.000397	0.000397	0.000405	0.023699
1980-12-16	0.000348	0.000348	0.000355	0.014246
1980-12-17	0.000361	0.000363	0.000371	0.011647
1980-12-18	0.000379	0.000381	0.000390	0.009897
...	...	...	...	...
2022-03-18	0.878848	0.899065	0.891886	0.016620
2022-03-21	0.895279	0.909290	0.910036	0.012910

```

2022-03-22  0.906233  0.926076  0.920646  0.010986
2022-03-23  0.919816  0.943682  0.935947  0.013213
2022-03-24  0.936631  0.951884  0.950243  0.012129

```

[10409 rows x 4 columns]

```

[51]: #Split Training and Test sets with 80/20
test_ratio=.1
training_ratio=1-test_ratio
test_size=int(test_ratio*len(aapl))
training_size=int(training_ratio*len(aapl))
x_train,x_test=feature_transform[:
    ↪training_size],feature_transform[training_size:]
y_train,y_test=target[:training_size],target[training_size:]
train_x,test_x=np.array(x_train),np.array(x_test)
#n x m -> n arrays containing 1 array with m elements
x_train=train_x.reshape(x_train.shape[0],1,x_train.shape[1])
x_test=test_x.reshape(x_test.shape[0],1,x_test.shape[1])

```

```

[52]: lstm=Sequential()
lstm.add(LSTM(32,input_shape=(1,train_x.
    ↪shape[1]),activation='relu',return_sequences=False))
lstm.add(Dense(1))
lstm.compile(loss='mean_squared_error',optimizer='adam')
history=lstm.
    ↪fit(x_train,y_train,epochs=100,batch_size=32,verbose=1,shuffle=False)

```

```

Epoch 1/100
293/293 [=====] - 1s 716us/step - loss: 88.7918
Epoch 2/100
293/293 [=====] - 0s 680us/step - loss: 76.9560
Epoch 3/100
293/293 [=====] - 0s 681us/step - loss: 68.5523
Epoch 4/100
293/293 [=====] - 0s 677us/step - loss: 63.2379
Epoch 5/100
293/293 [=====] - 0s 685us/step - loss: 58.4186
Epoch 6/100
293/293 [=====] - 0s 677us/step - loss: 54.8505
Epoch 7/100
293/293 [=====] - 0s 698us/step - loss: 50.5184
Epoch 8/100
293/293 [=====] - 0s 683us/step - loss: 46.0803
Epoch 9/100
293/293 [=====] - 0s 714us/step - loss: 41.1552
Epoch 10/100
293/293 [=====] - 0s 719us/step - loss: 35.5283
Epoch 11/100

```

293/293 [=====] - 0s 740us/step - loss: 29.3769  
 Epoch 12/100  
 293/293 [=====] - 0s 712us/step - loss: 23.1234  
 Epoch 13/100  
 293/293 [=====] - 0s 711us/step - loss: 17.2399  
 Epoch 14/100  
 293/293 [=====] - 0s 705us/step - loss: 12.1046  
 Epoch 15/100  
 293/293 [=====] - 0s 710us/step - loss: 7.9437  
 Epoch 16/100  
 293/293 [=====] - 0s 698us/step - loss: 4.8293  
 Epoch 17/100  
 293/293 [=====] - 0s 701us/step - loss: 2.6983  
 Epoch 18/100  
 293/293 [=====] - 0s 684us/step - loss: 1.3810  
 Epoch 19/100  
 293/293 [=====] - 0s 701us/step - loss: 0.6531  
 Epoch 20/100  
 293/293 [=====] - 0s 686us/step - loss: 0.2975  
 Epoch 21/100  
 293/293 [=====] - 0s 677us/step - loss: 0.1457  
 Epoch 22/100  
 293/293 [=====] - 0s 683us/step - loss: 0.0900  
 Epoch 23/100  
 293/293 [=====] - 0s 679us/step - loss: 0.0732  
 Epoch 24/100  
 293/293 [=====] - 0s 679us/step - loss: 0.0699  
 Epoch 25/100  
 293/293 [=====] - 0s 672us/step - loss: 0.0704  
 Epoch 26/100  
 293/293 [=====] - 0s 692us/step - loss: 0.0715  
 Epoch 27/100  
 293/293 [=====] - 0s 682us/step - loss: 0.0724  
 Epoch 28/100  
 293/293 [=====] - 0s 711us/step - loss: 0.0729  
 Epoch 29/100  
 293/293 [=====] - 0s 730us/step - loss: 0.0728  
 Epoch 30/100  
 293/293 [=====] - 0s 718us/step - loss: 0.0723  
 Epoch 31/100  
 293/293 [=====] - 0s 698us/step - loss: 0.0712  
 Epoch 32/100  
 293/293 [=====] - 0s 710us/step - loss: 0.0697  
 Epoch 33/100  
 293/293 [=====] - 0s 715us/step - loss: 0.0681  
 Epoch 34/100  
 293/293 [=====] - 0s 687us/step - loss: 0.0666  
 Epoch 35/100

293/293 [=====] - 0s 706us/step - loss: 0.0653  
Epoch 36/100  
293/293 [=====] - 0s 684us/step - loss: 0.0643  
Epoch 37/100  
293/293 [=====] - 0s 687us/step - loss: 0.0637  
Epoch 38/100  
293/293 [=====] - 0s 700us/step - loss: 0.0634  
Epoch 39/100  
293/293 [=====] - 0s 688us/step - loss: 0.0633  
Epoch 40/100  
293/293 [=====] - 0s 712us/step - loss: 0.0634  
Epoch 41/100  
293/293 [=====] - 0s 680us/step - loss: 0.0637  
Epoch 42/100  
293/293 [=====] - 0s 688us/step - loss: 0.0640  
Epoch 43/100  
293/293 [=====] - 0s 696us/step - loss: 0.0643  
Epoch 44/100  
293/293 [=====] - 0s 720us/step - loss: 0.0647  
Epoch 45/100  
293/293 [=====] - 0s 701us/step - loss: 0.0652  
Epoch 46/100  
293/293 [=====] - 0s 702us/step - loss: 0.0656  
Epoch 47/100  
293/293 [=====] - 0s 673us/step - loss: 0.0661  
Epoch 48/100  
293/293 [=====] - 0s 687us/step - loss: 0.0666  
Epoch 49/100  
293/293 [=====] - 0s 694us/step - loss: 0.0672  
Epoch 50/100  
293/293 [=====] - 0s 696us/step - loss: 0.0677  
Epoch 51/100  
293/293 [=====] - 0s 699us/step - loss: 0.0684  
Epoch 52/100  
293/293 [=====] - 0s 698us/step - loss: 0.0690  
Epoch 53/100  
293/293 [=====] - 0s 683us/step - loss: 0.0696  
Epoch 54/100  
293/293 [=====] - 0s 670us/step - loss: 0.0702  
Epoch 55/100  
293/293 [=====] - 0s 676us/step - loss: 0.0708  
Epoch 56/100  
293/293 [=====] - 0s 681us/step - loss: 0.0714  
Epoch 57/100  
293/293 [=====] - 0s 671us/step - loss: 0.0719  
Epoch 58/100  
293/293 [=====] - 0s 682us/step - loss: 0.0724  
Epoch 59/100

293/293 [=====] - 0s 692us/step - loss: 0.0729  
Epoch 60/100  
293/293 [=====] - 0s 682us/step - loss: 0.0734  
Epoch 61/100  
293/293 [=====] - 0s 722us/step - loss: 0.0739  
Epoch 62/100  
293/293 [=====] - 0s 708us/step - loss: 0.0743  
Epoch 63/100  
293/293 [=====] - 0s 695us/step - loss: 0.0746  
Epoch 64/100  
293/293 [=====] - 0s 698us/step - loss: 0.0750  
Epoch 65/100  
293/293 [=====] - 0s 685us/step - loss: 0.0753  
Epoch 66/100  
293/293 [=====] - 0s 712us/step - loss: 0.0755  
Epoch 67/100  
293/293 [=====] - 0s 679us/step - loss: 0.0758  
Epoch 68/100  
293/293 [=====] - 0s 698us/step - loss: 0.0760  
Epoch 69/100  
293/293 [=====] - 0s 685us/step - loss: 0.0762  
Epoch 70/100  
293/293 [=====] - 0s 684us/step - loss: 0.0763  
Epoch 71/100  
293/293 [=====] - 0s 699us/step - loss: 0.0764  
Epoch 72/100  
293/293 [=====] - 0s 689us/step - loss: 0.0766  
Epoch 73/100  
293/293 [=====] - 0s 700us/step - loss: 0.0767  
Epoch 74/100  
293/293 [=====] - 0s 710us/step - loss: 0.0767  
Epoch 75/100  
293/293 [=====] - 0s 694us/step - loss: 0.0768  
Epoch 76/100  
293/293 [=====] - 0s 714us/step - loss: 0.0768  
Epoch 77/100  
293/293 [=====] - 0s 688us/step - loss: 0.0769  
Epoch 78/100  
293/293 [=====] - 0s 700us/step - loss: 0.0769  
Epoch 79/100  
293/293 [=====] - 0s 692us/step - loss: 0.0769  
Epoch 80/100  
293/293 [=====] - 0s 704us/step - loss: 0.0769  
Epoch 81/100  
293/293 [=====] - 0s 709us/step - loss: 0.0768  
Epoch 82/100  
293/293 [=====] - 0s 681us/step - loss: 0.0768  
Epoch 83/100

```

293/293 [=====] - 0s 681us/step - loss: 0.0768
Epoch 84/100
293/293 [=====] - 0s 697us/step - loss: 0.0767
Epoch 85/100
293/293 [=====] - 0s 690us/step - loss: 0.0767
Epoch 86/100
293/293 [=====] - 0s 691us/step - loss: 0.0766
Epoch 87/100
293/293 [=====] - 0s 704us/step - loss: 0.0766
Epoch 88/100
293/293 [=====] - 0s 684us/step - loss: 0.0765
Epoch 89/100
293/293 [=====] - 0s 689us/step - loss: 0.0765
Epoch 90/100
293/293 [=====] - 0s 699us/step - loss: 0.0764
Epoch 91/100
293/293 [=====] - 0s 701us/step - loss: 0.0763
Epoch 92/100
293/293 [=====] - 0s 685us/step - loss: 0.0763
Epoch 93/100
293/293 [=====] - 0s 679us/step - loss: 0.0762
Epoch 94/100
293/293 [=====] - 0s 681us/step - loss: 0.0761
Epoch 95/100
293/293 [=====] - 0s 693us/step - loss: 0.0760
Epoch 96/100
293/293 [=====] - 0s 713us/step - loss: 0.0760
Epoch 97/100
293/293 [=====] - 0s 700us/step - loss: 0.0759
Epoch 98/100
293/293 [=====] - 0s 715us/step - loss: 0.0758
Epoch 99/100
293/293 [=====] - 0s 712us/step - loss: 0.0757
Epoch 100/100
293/293 [=====] - 0s 710us/step - loss: 0.0756

```

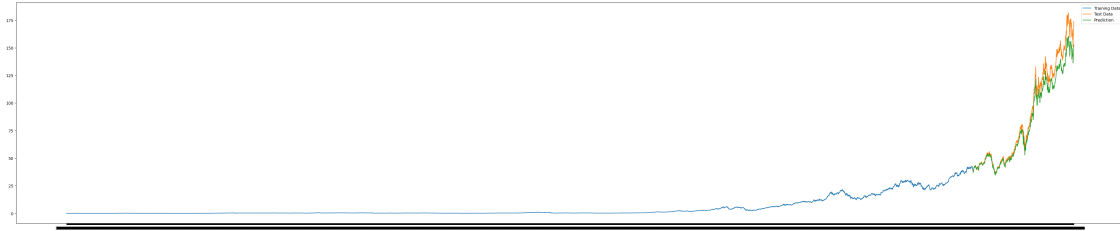
```
[55]: y_pred=lstm.predict(x_test)
```

```
33/33 [=====] - 0s 512us/step
```

```
[59]: y_test['Predicted']=y_pred
```

```
[60]: plt.figure(figsize=(50,10))
plt.plot(y_train['Adj Close'],label='Training Data')
plt.plot(y_test['Adj Close'],label='Test Data')
plt.plot(y_test['Predicted'],label='Prediction')
plt.legend()
plt.show()
```





```
[62]: #more test data=more accuracy
print('The Mean Squared Error: ',mean_squared_error(y_test['Adj Close'].
↪values,y_test['Predicted'].values))
print('The Mean Absolute Error: ',mean_absolute_error(y_test['Adj Close'].
↪values,y_test['Predicted'].values))
print('The Root Mean Squared Error: ',np.sqrt(mean_squared_error(y_test['Adj_
↪Close'].values,y_test['Predicted'].values)))
```

The Mean Squared Error: 76.6663812472651

The Mean Absolute Error: 6.302792272016746

The Root Mean Squared Error: 8.755934059097584