tsla-stock

September 13, 2024

1 By Prisca

[79]:
```
# TSLA  stock prediction using LSTM model
```

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
```

[76]:
```
#load the data
```

[2]:
```
df=pd.read_csv(r'C:\Users\USER\Documents\dataset\TSLA.csv')
```

[3]:
```
df.head()
```

[3]:

	Unnamed: 0	Date	Open	High	Low	Close	Volume
0	0	2010-06-29	1.266667	1.666667	1.169333	1.592667	281494500
1	1	2010-06-30	1.719333	2.028000	1.553333	1.588667	257806500
2	2	2010-07-01	1.666667	1.728000	1.351333	1.464000	123282000
3	3	2010-07-02	1.533333	1.540000	1.247333	1.280000	77097000
4	4	2010-07-06	1.333333	1.333333	1.055333	1.074000	103003500

[4]:
```
df.shape
```

[4]:
```
(3534, 7)
```

[5]:
```
df.drop(['Unnamed: 0'],axis=1,inplace=True)
```

[6]:
```
df.set_index(['Date'],inplace=True)
```

[7]:
```
df.head() # predicting stock for 2024-7-16, stock start from 2010 to 2024 july
```

[7]:

Date	Open	High	Low	Close	Volume
2010-06-29	1.266667	1.666667	1.169333	1.592667	281494500
2010-06-30	1.719333	2.028000	1.553333	1.588667	257806500
2010-07-01	1.666667	1.728000	1.351333	1.464000	123282000

```
2010-07-02  1.533333  1.540000  1.247333  1.280000   77097000
2010-07-06  1.333333  1.333333  1.055333  1.074000  103003500
```

[8]:
```
df.isnull().sum()
```

[8]:
```
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

[9]:
```
df.duplicated().sum()
```

[9]:
```
0
```

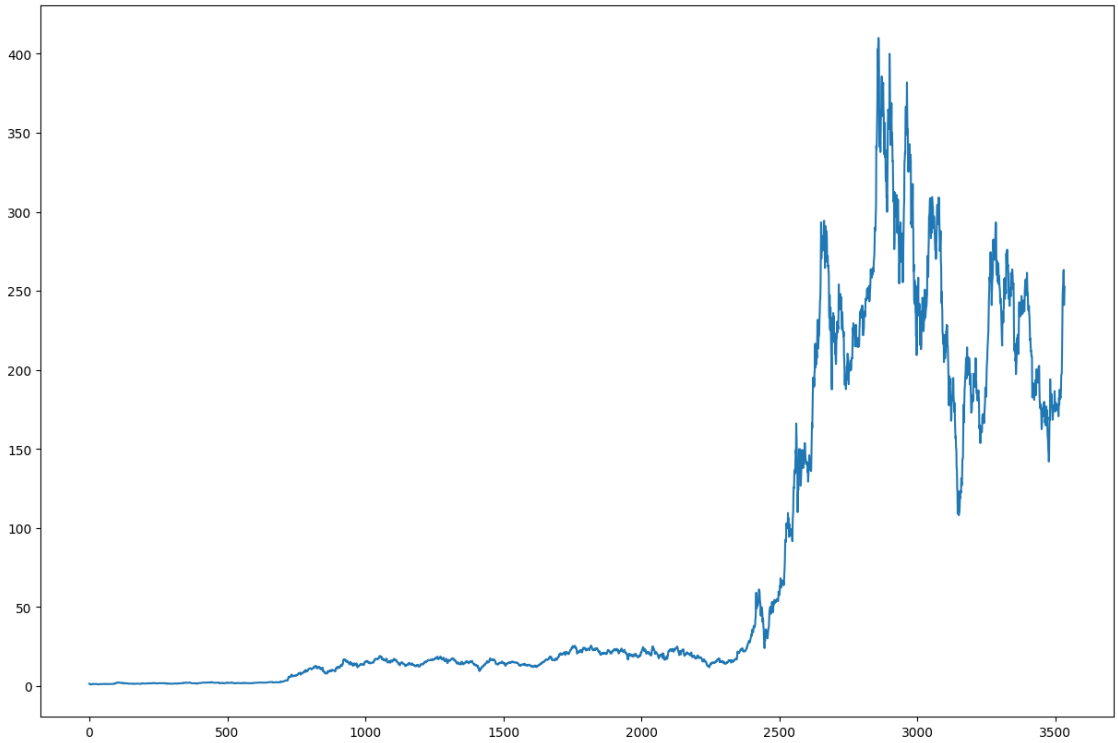
[77]:
```
# selection of target column
```

[78]:
```
new_df=df['Close'].values
```

[11]:
```
new_df=pd.DataFrame(new_df)
```

[12]:
```
plt.figure(figsize=(15,10))
plt.plot(new_df)
```

[12]:
```
[<matplotlib.lines.Line2D at 0x1324fafdaf0>]
```



[80]:
```
# data scaling
```

[13]:
```
from sklearn.preprocessing import MinMaxScaler
```

[14]:
```
scaler=MinMaxScaler(feature_range=(0,1))
```

[15]:
```
df_scaled=scaler.fit_transform(new_df)
```

[16]:
```
# split the data
```

[98]:
```
split_index=int(len(df_scaled)*0.8)
```

[99]:
```
train_df=df_scaled[:split_index]
test_df=df_scaled[split_index:]
```

[100]:
```
train_df.shape,test_df.shape
```

[100]:
```
((2827, 1), (707, 1))
```

[101]:
```
# created sequence for both train and test data
```

[102]:
```
x_train=[]
y_train=[]

for i in range(60,len(train_df)):
    x_train.append(train_df[i-60:i, 0])
    y_train.append(train_df[i,0])

x_train=np.array(x_train)
y_train=np.array(y_train)
```

[103]:
```
x_test=[]
y_test=[]

for i in range(60,len(test_df)):
    x_test.append(test_df[i-60:i,0])
    y_test.append(test_df[i,0])

x_test=np.array(x_test)
y_test=np.array(y_test)
```

[104]:
```
x_train.shape,x_test.shape
```

[104]:
```
((2767, 60), (647, 60))
```

[105]:
```
y_train.shape,y_test.shape
```

[105]:
```
((2767,), (647,))
```

2 reshape

[106]:
```
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)
```

[107]:
```
x_train.shape,x_test.shape
```

[107]:
```
((2767, 60, 1), (647, 60, 1))
```

3 build model

[29]:
```
from keras .models import Sequential
from keras .layers import Dense,Dropout,LSTM
```

[30]:
```
model=Sequential()
```

[31]:
```
model.add(LSTM(units=50,return_sequences=True,input_shape=( x_train.
↪shape[1],x_train.shape[2])))
model.add(Dropout(0.2))
```

[32]:
```
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
```

[33]:
```
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
```

[34]:
```
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
```

[35]:
```
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
```

[36]:
```
model.add(LSTM(units=50,return_sequences=False))
model.add(Dropout(0.2))
```

[37]:
```
model.add(Dense(units=1))
```

[ ]:
```
# compile
```

[39]:
```
model.compile(optimizer='adam',loss='mean_squared_error')
```

[40]:
```
#fit the model
```

[41]:
```
history=model.
  ↪fit(x_train,y_train,validation_data=(x_test,y_test),epochs=20,batch_size=2)
```

```
Epoch 1/20
1384/1384 [==============================] - 300s 202ms/step - loss: 0.0024 -
val_loss: 0.0149
Epoch 2/20
1384/1384 [==============================] - 281s 203ms/step - loss: 0.0014 -
val_loss: 0.0110
Epoch 3/20
1384/1384 [==============================] - 279s 201ms/step - loss: 0.0012 -
val_loss: 0.0102
Epoch 4/20
1384/1384 [==============================] - 273s 197ms/step - loss: 0.0010 -
val_loss: 0.0100
Epoch 5/20
1384/1384 [==============================] - 278s 201ms/step - loss: 9.9633e-04
- val_loss: 0.0259
Epoch 6/20
1384/1384 [==============================] - 283s 205ms/step - loss: 8.8110e-04
- val_loss: 0.0090
Epoch 7/20
1384/1384 [==============================] - 274s 198ms/step - loss: 8.3555e-04
- val_loss: 0.0090
Epoch 8/20
1384/1384 [==============================] - 281s 203ms/step - loss: 7.4407e-04
- val_loss: 0.0094
Epoch 9/20
1384/1384 [==============================] - 274s 198ms/step - loss: 8.6287e-04
- val_loss: 0.0077
Epoch 10/20
1384/1384 [==============================] - 272s 196ms/step - loss: 6.2938e-04
- val_loss: 0.0051
Epoch 11/20
1384/1384 [==============================] - 359s 260ms/step - loss: 6.6233e-04
- val_loss: 0.0186
Epoch 12/20
1384/1384 [==============================] - 524s 379ms/step - loss: 7.7214e-04
- val_loss: 0.0138
Epoch 13/20
1384/1384 [==============================] - 809s 584ms/step - loss: 6.7801e-04
- val_loss: 0.0058
Epoch 14/20
1384/1384 [==============================] - 957s 691ms/step - loss: 6.4346e-04
- val_loss: 0.0069
```

```
Epoch 15/20
1384/1384 [==============================] - 897s 648ms/step - loss: 7.0937e-04
- val_loss: 0.0085
Epoch 16/20
1384/1384 [==============================] - 843s 609ms/step - loss: 6.1364e-04
- val_loss: 0.0071
Epoch 17/20
1384/1384 [==============================] - 929s 671ms/step - loss: 5.9391e-04
- val_loss: 0.0055
Epoch 18/20
1384/1384 [==============================] - 942s 681ms/step - loss: 6.0241e-04
- val_loss: 0.0068
Epoch 19/20
1384/1384 [==============================] - 383s 276ms/step - loss: 5.7224e-04
- val_loss: 0.0087
Epoch 20/20
1384/1384 [==============================] - 281s 203ms/step - loss: 5.4193e-04
- val_loss: 0.0087
```

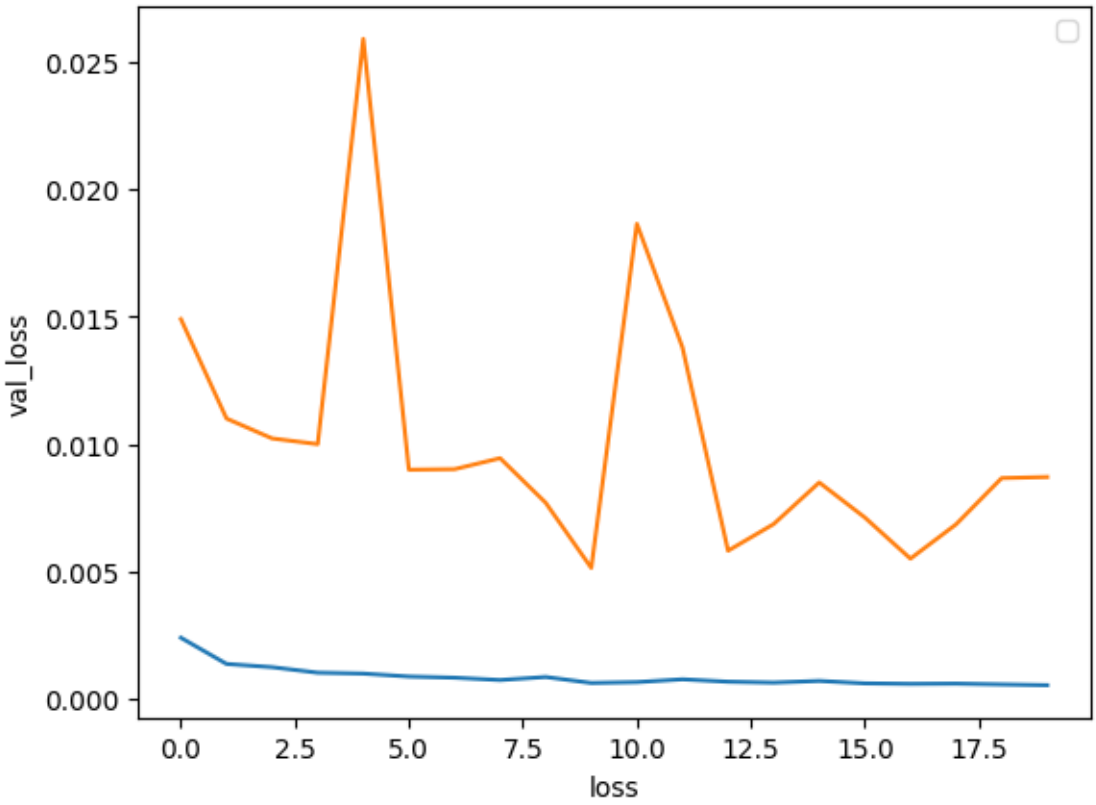
[90]:
```
#model evaluation
```

[108]:
```
loss_t=(history.history['loss'])
loss_v=(history.history['val_loss'])
```

[109]:
```
plt.plot(loss_t)
plt.plot(loss_v)
plt.xlabel('loss')
plt.ylabel('val_loss')
plt.legend()  # as the val loss is going down, the loss is low too
```

```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```

[109]:
```
<matplotlib.legend.Legend at 0x1325c6f8a00>
```



4 prediction

[110]:
```
predictions=model.predict(x_test)
```

```
21/21 [==============================] - 2s 88ms/step
```

[111]:
```
predictions=scaler.inverse_transform(predictions)
```

[112]:
```
predictions.shape
```

[112]:
```
(647, 1)
```

[113]:
```
# Reshape y_test
```

[114]:
```
y_test=y_test.reshape(-1,1)
```

[115]:
```
y_test.shape
```

[115]:
```
(647, 1)
```

[116]:
```
y_test_sc=scaler.inverse_transform(y_test)
```

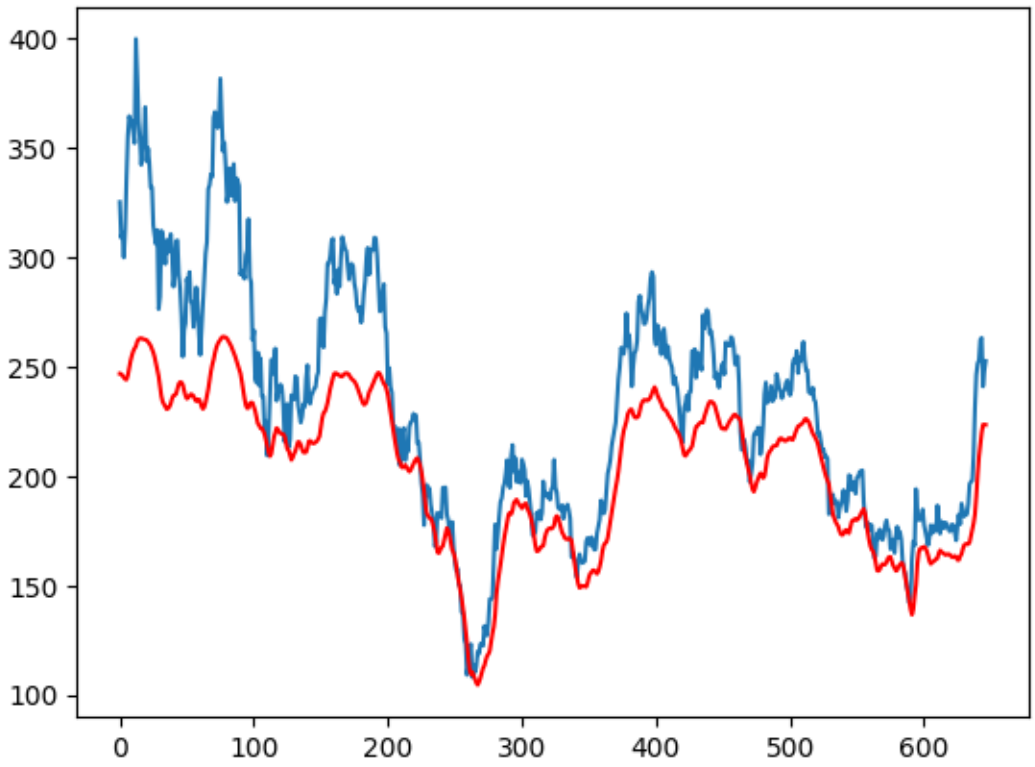
[117]:
```
y_test_sc.shape,predictions.shape
```

[117]:
```
((647, 1), (647, 1))
```

5 stock visualization

[119]:
```
plt.plot(y_test_sc)
plt.plot(predictions,color='red') # trend highly copied
```

[119]:
```
[<matplotlib.lines.Line2D at 0x13253b46310>]
```



[56]:
```
#evaluation
```

[120]:
```
from sklearn .metrics import mean_squared_error
```

[121]:
```
rmse=np.sqrt(mean_squared_error(y_test_sc,predictions))
```

[122]:
```
rmse # the error is high but what matter is the trend is quite captured,if i␣
 ↪train it more, the error will reduce
```

[122]:
```
38.148803216812006
```

6 lets predict the next day that is 2024-7-16

[123]:
```
# reload the data
```

[124]:
```
df=pd.read_csv(r'C:\Users\USER\Documents\dataset\TSLA.csv')
```

[125]:
```
df.tail()
```

[125]:

	Unnamed: 0	Date	Open	High	Low	Close	Volume
3529	3529	2024-07-09	251.000000	265.609985	250.300003	262.329987	160210900
3530	3530	2024-07-10	262.799988	267.589996	257.859985	263.260010	128519400
3531	3531	2024-07-11	263.299988	271.000000	239.649994	241.029999	221707300
3532	3532	2024-07-12	235.800003	251.839996	233.089996	248.229996	155694400
3533	3533	2024-07-15	255.964996	265.579987	251.729996	252.639999	142831728

[87]:
```
# chose the target varaible
```

[126]:
```
new_df=df['Close']
```

[127]:
```
# select the last 60 days
```

[128]:
```
last_60_days=new_df[-60:]
```

[129]:
```
last=pd.DataFrame(last_60_days)
```

[130]:
```
last.shape
```

[130]:
```
(60, 1)
```

[131]:
```
last_days=scaler.transform(last)
```

```
C:\Users\USER\anaconda3\envs\tf_env\lib\site-packages\sklearn\base.py:458:
UserWarning: X has feature names, but MinMaxScaler was fitted without feature
names
  warnings.warn(
```

[89]:
```
#created sequence and append the 60 days
```

[132]:
```
x_test=[]
x_test.append(last_days)

x_test=np.array(x_test)
```

[133]:
```
x_test.shape
```

[133]:
```
(1, 60, 1)
```

[134]:
```
#reshape
```

[135]:
```
x_predict=model.predict(x_test)
```

```
1/1 [==============================] - 0s 71ms/step
```

[136]:
```
x_predict=scaler.inverse_transform(x_predict)
```

[138]:
```
x_predict #stock price for 2024-7-16
```

[138]:
```
array([[223.94717]], dtype=float32)
```

[ ]:

[ ]: