```
##Everything together
globals().clear
import cv2

import holidays
import pandas as pd
from math import sqrt
import numpy as np
from numpy import concatenate
from matplotlib import pyplot
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline
import openpyxl as xlss
import pandas as pd
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from sklearn import preprocessing, metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import models, layers, Sequential
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.optimizers    import Adam
from datetime import datetime


stock = pd.read_excel('merged_onehot_test.xlsx')
max=stock['TSLA_close'].max()
min=stock['TSLA_close'].min()


stock_norm = pd.read_excel('merged_normalization.xlsx')


t1=stock_norm
t1.index=t1['Date']
t1.drop(columns=t1.columns[0:2],
        axis=1,
        inplace=True)
t1.head()
```

Date	TSLA_close	TSLA_volume	TSLA_vol_4_ave	TSLA_vwap	TSLA_vwap_4_ave	TSLA_trai
2020-06-01 10:30:00	0.003937	0.160478	0.510144	0.003262	0.001627	
2020-06-01 10:45:00	0.004076	0.128409	0.377649	0.003393	0.002439	
2020-06-01 11:00:00	0.003900	0.096654	0.285393	0.003786	0.002903	
2020-06-01 11:15:00	0.002882	0.100758	0.213820	0.003165	0.003116	
2020-06-01 11:30:00	0.002812	0.070695	0.196724	0.003028	0.003298	

5 rows × 67 columns

```
def mean_absolute_percentage_error(y_true, y_pred):
    dfff = pd.DataFrame({'tr':y_true,'pr':y_pred})
    subsetf = dfff.loc[dfff['tr'] > 0]
    y_true_, y_pred_ = np.array(subsetf['tr']), np.array(subsetf['pr'])
    return np.mean(np.abs(y_true_ - y_pred_) / y_true_) * 100


def mean_percentage_error(y_true, y_pred):
    dfff = pd.DataFrame({'tr':y_true,'pr':y_pred})
    subsetf = dfff.loc[dfff['tr'] > 0]
    y_true_, y_pred_ = np.array(subsetf['tr']), np.array(subsetf['pr'])
    return np.mean((y_true_ - y_pred_) / y_true_) * 100


train_df = t1.loc['2020-06-01 10:30:00':'2021-12-31 16:00:00']
test_df = t1.loc['2022-01-01 09:30:00':'2022-05-27 16:00:00']


import time


start = time.time()
predictions = list()
a=1
count_time=list()
```

```
for i in test_df['week_label'].unique():

  st = time.time()

  test_subset = test_df[test_df['week_label']==i]
  print(train_df.index[0])
  print(train_df.index[-1])
  print(test_subset.index[0])
  print(test_subset.index[-1])
  train_stand = train_df.copy()
  test_stand = test_subset.copy()

  X_train, y_train = np.array(train_stand.iloc[:,1:65]), train_stand.iloc[:,0]
  X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))


  # design network
  model = Sequential()
  model.add(layers.LSTM(50,return_sequences = True, input_shape=(X_train.shape[1], X_train.sh
  model.add(layers.LSTM(25))
  model.add(layers.Dense(1))
  opt = Adam(amsgrad = True, learning_rate = 0.001, beta_1 = 0.79, beta_2 = 0.999)
  model.compile(loss = 'mse', optimizer = opt)
  # fit network
  history  = model.fit(X_train, y_train, epochs=30, batch_size=65, verbose=2, shuffle=False)


  X_test, y_test = np.array(test_stand.iloc[:,1:65]), test_stand.iloc[:,0]
  X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))
  y_hat=model.predict(X_test)
  predictions.append(y_hat)
  et = time.time()
  used_time=et-st
  count_time.append(used_time)

  train_df = train_df.append(test_df[test_df['week_label']==i])
  train_df=train_df.drop(train_df[train_df['week_label']==a].index)
  a+=1
  print(train_df.index[0])
  print(train_df.index[-1])
  print('Time taken:'+str(used_time))
  print('--------------------------')

end = time.time()
print("total used time"+str(end-start))
```

```
167/167 - 1s - loss: 3.1358e-04 - 621ms/epoch - 4ms/step
Epoch 28/30
167/167 - 1s - loss: 2.8009e-04 - 626ms/epoch - 4ms/step
Epoch 29/30
167/167 - 1s - loss: 2.5070e-04 - 640ms/epoch - 4ms/step
Epoch 30/30
167/167 - 1s - loss: 2.2911e-04 - 631ms/epoch - 4ms/step
```

```
2020-06-15 09:30:00
2022-01-14 16:00:00
Time taken:30.467204809188843
---------------------------
2020-06-15 09:30:00
2022-01-14 16:00:00
2022-01-18 09:30:00
2022-01-21 16:00:00
Epoch 1/30
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: T
  super(Adam, self).__init__(name, **kwargs)
167/167 - 4s - loss: 0.0028 - 4s/epoch - 26ms/step
Epoch 2/30
167/167 - 1s - loss: 0.0064 - 624ms/epoch - 4ms/step
Epoch 3/30
167/167 - 1s - loss: 0.0047 - 640ms/epoch - 4ms/step
Epoch 4/30
167/167 - 1s - loss: 0.0031 - 649ms/epoch - 4ms/step
Epoch 5/30
167/167 - 1s - loss: 0.0022 - 660ms/epoch - 4ms/step
Epoch 6/30
167/167 - 1s - loss: 0.0018 - 638ms/epoch - 4ms/step
Epoch 7/30
167/167 - 1s - loss: 0.0016 - 647ms/epoch - 4ms/step
Epoch 8/30
167/167 - 1s - loss: 0.0015 - 613ms/epoch - 4ms/step
Epoch 9/30
167/167 - 1s - loss: 0.0014 - 627ms/epoch - 4ms/step
Epoch 10/30
167/167 - 1s - loss: 0.0012 - 620ms/epoch - 4ms/step
Epoch 11/30
167/167 - 1s - loss: 0.0011 - 622ms/epoch - 4ms/step
Epoch 12/30
167/167 - 1s - loss: 0.0011 - 636ms/epoch - 4ms/step
Epoch 13/30
167/167 - 1s - loss: 9.6601e-04 - 628ms/epoch - 4ms/step
Epoch 14/30
167/167 - 1s - loss: 8.9222e-04 - 609ms/epoch - 4ms/step
Epoch 15/30
167/167 - 1s - loss: 8.3273e-04 - 604ms/epoch - 4ms/step
Epoch 16/30
167/167 - 1s - loss: 7.8699e-04 - 622ms/epoch - 4ms/step
Epoch 17/30
167/167 - 1s - loss: 7.5159e-04 - 613ms/epoch - 4ms/step
Epoch 18/30
167/167 - 1s - loss: 7.2188e-04 - 617ms/epoch - 4ms/step
Epoch 19/30
167/167 - 1s - loss: 6.9394e-04 - 599ms/epoch - 4ms/step
Epoch 20/30
167/167 - 1s - loss: 6.6527e-04 - 634ms/epoch - 4ms/step
Epoch 21/30
```

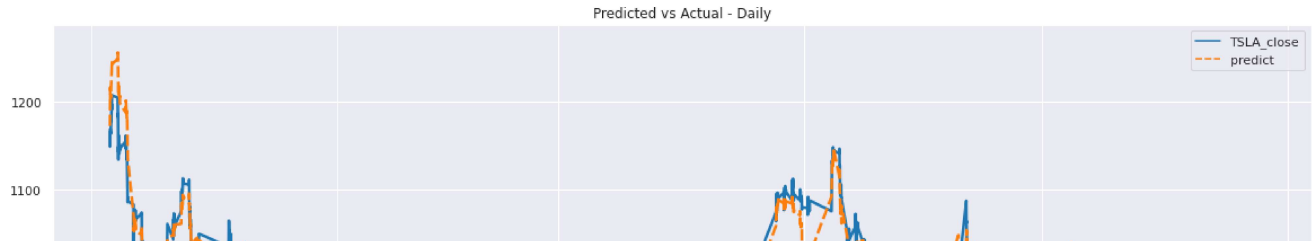
```
df_expe = pd.DataFrame(test_df.iloc[:,0])
pred_list= list()
```

```
for i in range(len(predictions)):
  pred_list=pred_list+predictions[i].tolist()


df_pred = pd.DataFrame(pred_list,index=test_df.index,columns= ['predict'])

df_Result = pd.concat([df_expe,df_pred],axis=1)
df_Result= df_Result.multiply(max-min).add(min)

df_Result
```

Date	TSLA_close	predict
2022-01-03 09:30:00	1165.198	1172.587546
2022-01-03 09:45:00	1149.000	1191.350045
2022-01-03 10:00:00	1156.590	1193.729311
2022-01-03 10:15:00	1152.005	1203.561883
2022-01-03 10:30:00	1153.290	1216.540359
...	...	...
2022-05-27 15:00:00	756.960	727.600061
2022-05-27 15:15:00	758.700	729.880771
2022-05-27 15:30:00	757.650	730.899521
2022-05-27 15:45:00	759.660	732.047602
2022-05-27 16:00:00	759.500	731.091483

2754 rows × 2 columns

```
plt.figure(figsize=(20,10))
linep = sns.lineplot(data=df_Result, palette="tab10", linewidth=2.5)
linep.set(xlabel='Date', ylabel='Price', title='Predicted vs Actual - Daily')
plt.show()
```




```
mean_absolute_percentage_error(df_Result['TSLA_close'],df_Result['predict'])
```

```
3.0420197314694484
```

```
mean_percentage_error(df_Result['TSLA_close'],df_Result['predict'])
```

```
-0.09792384211271628
```

```
from sklearn.metrics import mean_squared_error
mean_squared_error(df_Result['TSLA_close'],df_Result['predict'],squared=False)
```

```
35.143548752315176
```

✓ 0s completed at 4:58 PM