SOURCE CODE:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import datetime
from pylab import rcParams
import matplotlib.pyplot as plt
import warnings
import itertools
import statsmodels.api as sm
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from sklearn.metrics import mean_squared_error
from keras.callbacks import ReduceLROnPlateau, EarlyStopping, ModelCheckpoint
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
import seaborn as sns
sns.set_context("paper", font_scale=1.3)
sns.set_style('white')
import math
from sklearn.preprocessing import MinMaxScaler
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
for filename in filenames:
print(os.path.join(dirname, filename))
  dateparse = lambda x: pd.datetime.strptime(x, '%b %d, %Y')
#Read csv file
from google.colab import files
uploaded = files.upload()
import io
df = pd.read_excel(io.BytesIO(uploaded['Crude Oil Prices Daily.xlsx']))
df.head()
df[:10]
#Sort dataset by column Date
df = df.sort_values('Date')
df = df.groupby('Date')['Closing Value'].sum().reset_index()
df.set_index('Date', inplace=True)
df=df.loc[datetime.date(year=2000,month=1,day=1):]
df.head()
def DfInfo(df_initial):
# gives some infos on columns types and numer of null values
tab_info = pd.DataFrame(df_initial.dtypes).T.rename(index={0: 'column type'})
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()).T.rename(ind
ex={0: 'null values (nb)'}))
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()/df_initial.shap
e[0] * 100).T.
rename(index={0: 'null values (%)'}))
 return tab_info
DfInfo(df)
```

```
df.index
y = df['Closing Value'].resample('MS').mean()
y.plot(figsize=(15, 6))
plt.show()
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
fig = decomposition.plot()
plt.show()
sc = MinMaxScaler(feature_range = (0, 1))
df = sc.fit_transform(df)
train_size = int(len(df) * 0.70)
test_size = len(df) - train_size
train, test = df[0:train_size, :], df[train_size:len(df), :]
def create_data_set(_data_set, _look_back=1):
 data_x, data_y = [], []
 for i in range(len(_data_set) - _look_back - 1):
 a = _data_set[i:(i + _look_back), 0]
 data_x.append(a)
 data_y.append(_data_set[i + _look_back, 0])
 return np.array(data_x), np.array(data_y)
look_back =90
X_train,Y_train,X_test,Ytest = [],[],[],[]
X_train,Y_train=create_data_set(train,look_back)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test,Y_test=create_data_set(test,look_back)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
regressor = Sequential()
```

```
regressor.add(LSTM(units = 60, return_sequences = True, input_shape = (X_train.shape[1], 1)))

regressor.add(Dropout(0.1))

regressor.add(LSTM(units = 60, return_sequences = True))

regressor.add(Dropout(0.1))

regressor.add(LSTM(units = 60))

regressor.add(Dropout(0.1))

regressor.add(Dense(units = 1))

regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

reduce_lr = ReduceLROnPlateau(monitor='val_loss',patience=5)

history =regressor.fit(X_train, Y_train, epochs = 20, batch_size = 15,validation_data=(X_test, Y_test), callbacks=[reduce_lr],shuffle=False)

train_predict = regressor.predict(X_train)

test_predict = regressor.predict(X_test)

train_predict = sc.inverse_transform(train_predict)

Y_train = sc.inverse_transform([Y_train])

test_predict = sc.inverse_transform(test_predict)

Y_test = sc.inverse_transform([Y_test])

print('Train Mean Absolute Error:', mean_absolute_error(Y_train[0], train_predict[:,0]))

print('Train Root Mean Squared Error:',np.sqrt(mean_squared_error(Y_train[0], train_predict[:,0])))

print('Test Mean Absolute Error:', mean_absolute_error(Y_test[0], test_predict[:,0]))

print('Test Root Mean Squared Error:',np.sqrt(mean_squared_error(Y_test[0], test_predict[:,0])))

plt.figure(figsize=(8,4))

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Test Loss')
```

```
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(loc='upper right')
plt.show();
aa=[x for x in range(180)]
plt.figure(figsize=(8,4))
plt.plot(aa, Y_test[0][:180], marker='.', label="actual")
plt.plot(aa, test_predict[:,0][:180], 'r', label="prediction")
plt.tight_layout()
sns.despine(top=True)
plt.subplots_adjust(left=0.07)
plt.ylabel('Price', size=15)
plt.xlabel('Time step', size=15)
plt.legend(fontsize=15)
plt.show();
```