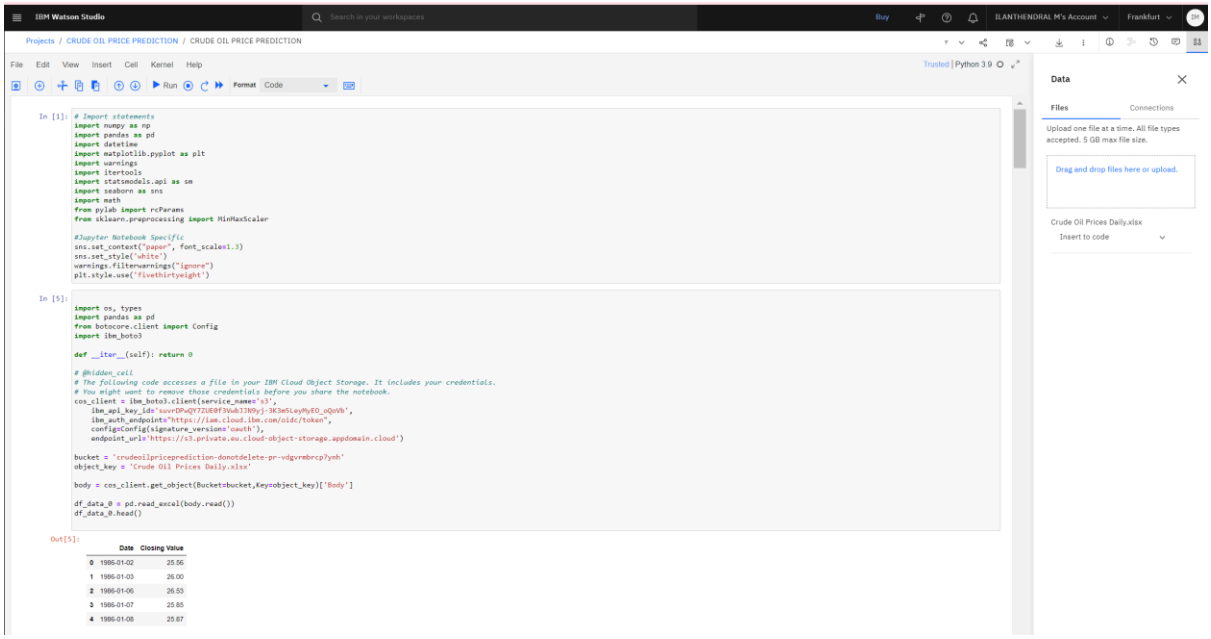
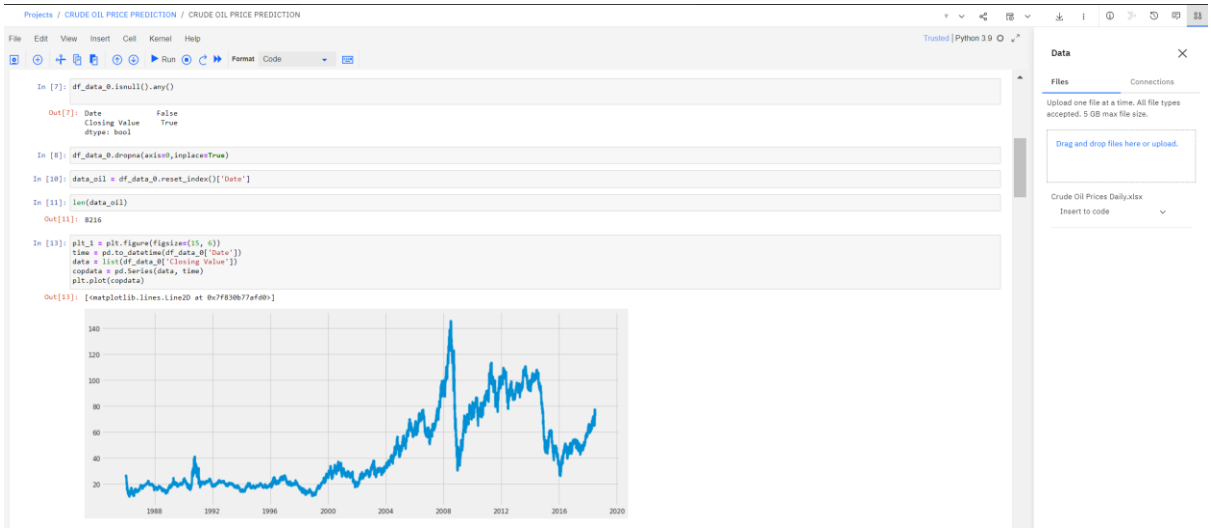
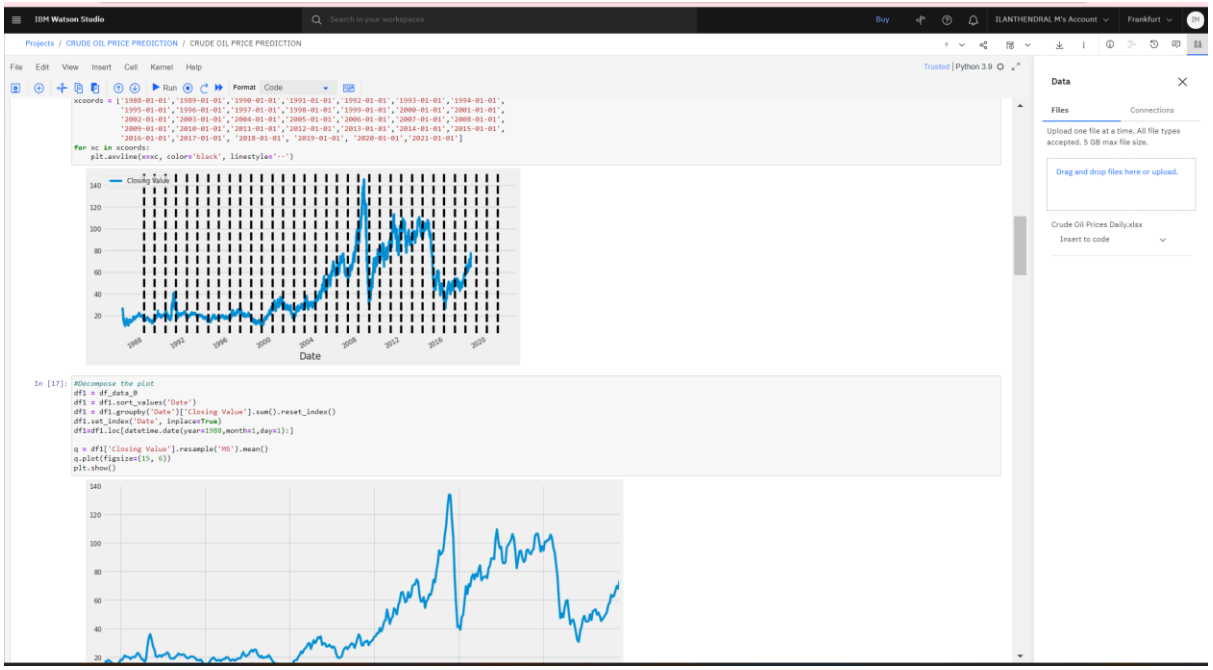
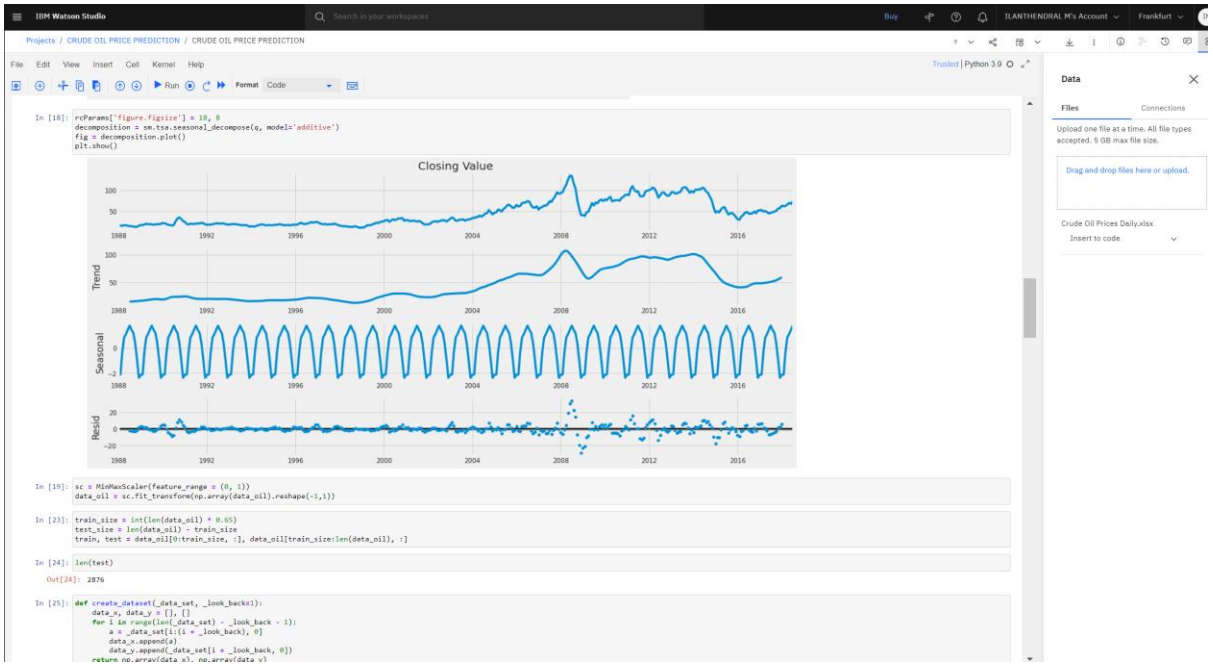

IBM Watson Studio
Search in your workspaces
Buy
ILANTHENDRAL M's Account
Frankfurt
Projects / CRUDE OIL PRICE PREDICTION / CRUDE OIL PRICE PREDICTION
File Edit View Insert Cell Kernel Help
Trusted | Python 3.9
Run
Format Code
In [1]:
# Import statements
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
import warnings
import itertools
import statsmodels.api as sm
import seaborn as sns
import math
from pylab import rcParams
from sklearn.preprocessing import MinMaxScaler
#Jupyter Notebook Specific
sns.set_context("paper", font_scale=1.3)
sns.set_style('white')
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
In [5]:
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3
def __iter__(self): return 0
# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
ibm_api_key_id='uvr0PwQY7ZUE0f9ub17N0yj-3K3mSLeyMyEO_oQaVb',
ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
config=Config(signature_version='oauth'),
endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')
bucket = 'crudeoilpricepredictiion-donotdelete-pr-vdgvrebrcp7ynh'
object_key = 'Crude Oil Prices Daily.xlsx'
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
df_data_0 = pd.read_excel(body.read())
df_data_0.head()
Out[5]:
Date Closing Value
0 1986-01-02 25.56
1 1986-01-03 26.00
2 1986-01-06 26.53
3 1986-01-07 25.85
4 1986-01-08 25.87
Data
Files
Connections
Upload one file at a time. All file types accepted. 5 GB max file size.
Drag and drop files here or upload.
Crude Oil Prices Daily.xlsx
Insert to code


Projects / CRUDE OIL PRICE PREDICTION / CRUDE OIL PRICE PREDICTION
File Edit View Insert Cell Kernel Help
Trusted | Python 3.9
Run
Format Code
In [7]:
df_data_0.isnull().any()
Out[7]:
Date False
Closing Value True
dtype: bool
In [8]:
df_data_0.dropna(axis=0,inplace=True)
In [10]:
data_oil = df_data_0.reset_index()['Date']
In [11]:
len(data_oil)
Out[11]:
8216
In [13]:
plt_1 = plt.figure(figsize=(15, 6))
time = pd.to_datetime(df_data_0['Date'])
data = list(df_data_0['Closing Value'])
copdata = pd.Series(data, time)
plt.plot(copdata)
Out[13]:
<matplotlib.lines.Line2D at 0x7f830b77afd0>
140
120
100
80
60
40
20
1988
1992
1996
2000
2004
2008
2012
2016
2020
Data
Files
Connections
Upload one file at a time. All file types accepted. 5 GB max file size.
Drag and drop files here or upload.
Crude Oil Prices Daily.xlsx
Insert to code


IBM Watson Studio
Projects / CRUDE OIL PRICE PREDICTION / CRUDE OIL PRICE PREDICTION
File Edit View Insert Cell Kernel Help
Run Format Code
Trusted | Python 3.9
Data
Files Connections
Upload one file at a time. All file types accepted. 5 GB max file size.
Drag and drop files here or upload.
Crude Oil Prices Daily.xlsx
Insert to code
Closing Value
Date
In [17]:
#Decompose the plot
df1 = df_data_0
df1 = df1.sort_values('Date')
df1 = df1.groupby('Date')['Closing Value'].sum().reset_index()
df1.set_index('Date', inplace=True)
df1=df1.loc[datetime.date(year=1988,month=1,day=1):]
q = df1['Closing Value'].resample('MS').mean()
q.plot(figsize=(15, 6))
plt.show()


IBM Watson Studio
Projects / CRUDE OIL PRICE PREDICTION / CRUDE OIL PRICE PREDICTION
File Edit View Insert Cell Kernel Help
Run Format Code
Trusted | Python 3.9
Data
Files Connections
Upload one file at a time. All file types accepted. 5 GB max file size.
Drag and drop files here or upload.
Crude Oil Prices Daily.xlsx
Insert to code
In [18]:
rcParams['figure.figsize'] = 10, 8
decomposition = sm.tsa.seasonal_decompose(q, model='additive')
fig = decomposition.plot()
plt.show()
Closing Value
Trend
Seasonal
Resid
In [19]:
sc = MinMaxScaler(feature_range = (0, 1))
data_oil = sc.fit_transform(np.array(data_oil).reshape(-1,1))
In [23]:
train_size = int(len(data_oil) * 0.65)
test_size = len(data_oil) - train_size
train, test = data_oil[0:train_size, :], data_oil[train_size:len(data_oil), :]
In [24]:
len(test)
Out[24]:
2876
In [25]:
def create_dataset(_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)

IBM Watson Studio | Search in your workspaces | Buy | ILANTHENDRAL M's Account | Frankfurt

Projects / CRUDE OIL PRICE PREDICTION / CRUDE OIL PRICE PREDICTION

File Edit View Insert Cell Kernel Help | Trusted | Python 3.9

Run | Format Code

In [28]:
```
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')
# early_stopping = EarlyStopping(monitor='val_loss',patience=20)
history =regressor.fit(X_train, Y_train, epochs = 20, batch_size = 64,validation_data=(X_test, Y_test),verbose=1)
# history =regressor.fit(X_train, Y_train, epochs = 100, batch_size = 64,validation_data=(X_test, Y_test), callbacks=[early_stopping],shuffle=False,verbo
```

```
Epoch 1/20
84/84 [==============================] - 7s 37ms/step - loss: 0.0095 - val_loss: 4.6844e-06
Epoch 2/20
84/84 [==============================] - 2s 27ms/step - loss: 5.5193e-04 - val_loss: 0.0010
Epoch 3/20
84/84 [==============================] - 2s 24ms/step - loss: 4.8525e-04 - val_loss: 1.3721e-04
Epoch 4/20
84/84 [==============================] - 2s 28ms/step - loss: 4.8271e-04 - val_loss: 2.8378e-04
Epoch 5/20
84/84 [==============================] - 2s 25ms/step - loss: 4.4876e-04 - val_loss: 2.1780e-04
Epoch 6/20
84/84 [==============================] - 2s 27ms/step - loss: 4.0779e-04 - val_loss: 1.3980e-04
Epoch 7/20
84/84 [==============================] - 2s 23ms/step - loss: 4.1373e-04 - val_loss: 2.0241e-04
Epoch 8/20
84/84 [==============================] - 2s 26ms/step - loss: 3.5424e-04 - val_loss: 3.2804e-04
Epoch 9/20
84/84 [==============================] - 2s 26ms/step - loss: 3.7755e-04 - val_loss: 6.8814e-05
Epoch 10/20
84/84 [==============================] - 2s 29ms/step - loss: 3.5605e-04 - val_loss: 2.8357e-04
Epoch 11/20
84/84 [==============================] - 2s 28ms/step - loss: 3.2133e-04 - val_loss: 5.7816e-04
Epoch 12/20
84/84 [==============================] - 2s 28ms/step - loss: 3.2639e-04 - val_loss: 2.7981e-04
Epoch 13/20
84/84 [==============================] - 2s 28ms/step - loss: 3.0551e-04 - val_loss: 0.0014
Epoch 14/20
84/84 [==============================] - 2s 29ms/step - loss: 3.2560e-04 - val_loss: 9.1151e-04
Epoch 15/20
84/84 [==============================] - 2s 24ms/step - loss: 2.9090e-04 - val_loss: 6.9674e-05
```

Data

Files | Connections

Upload one file at a time. All file types accepted. 5 GB max file size.

Drag and drop files here or upload.

Crude Oil Prices Daily.xlsx

Insert to code

IBM Watson Studio | Search in your workspaces | Buy | ILANTHENDRAL M's Account | Frankfurt

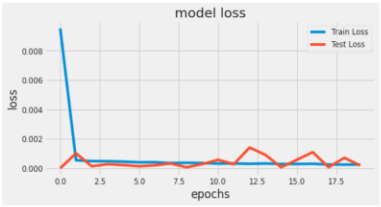
Projects / CRUDE OIL PRICE PREDICTION / CRUDE OIL PRICE PREDICTION

File Edit View Insert Cell Kernel Help | Trusted | Python 3.9

Run | Format Code

In [32]:
```
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();
```

```
Train Mean Absolute Error: 651452159318545.8
Train Root Mean Squared Error: 768561729052668.4
Test Mean Absolute Error: 1.1510667509622272e+16
Test Root Mean Squared Error: 1.4298628039561374e+16
```



In [33]:
```
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();
```

Data

Files | Connections

Upload one file at a time. All file types accepted. 5 GB max file size.

Drag and drop files here or upload.

Crude Oil Prices Daily.xlsx

Insert to code

IBM Watson Studio

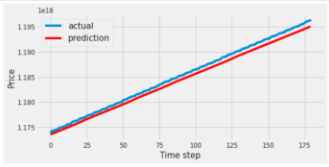
Projects / CRUDE OIL PRICE PREDICTION / CRUDE OIL PRICE PREDICTION

File Edit View Insert Cell Kernel Help

Trusted | Python 3.9

Run Format Code

```
In [33]: 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();
```




```
In [34]: regressor.save("crude_oil.h5")
```

```
In [35]: #test the model
         look_back = 10
         trainPredictPlot = np.empty_like(data_oil)
         trainPredictPlot[:,:] = np.nan
         trainPredictPlot[look_back:len(train_predict)+look_back,:] = train_predict

         testPredictPlot = np.empty_like(data_oil)
         testPredictPlot[:,:] = np.nan
         testPredictPlot[len(train_predict)+(look_back*2)+1:len(data_oil)-1,:] = test_predict

         plt.plot(sc.inverse_transform(data_oil))
         plt.plot(trainPredictPlot)
         plt.plot(testPredictPlot)
         plt.show()
```

Data

Files Connections

Upload one file at a time. All file types accepted. 5 GB max file size.

Drag and drop files here or upload.

Crude Oil Prices Daily.xlsx

Insert to code

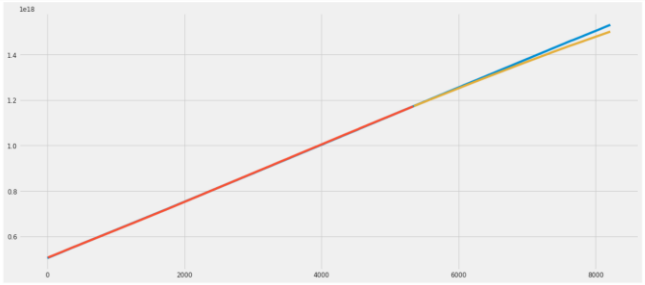
IBM Watson Studio

Projects / CRUDE OIL PRICE PREDICTION / CRUDE OIL PRICE PREDICTION

File Edit View Insert Cell Kernel Help

Trusted | Python 3.9

Run Format Code




```
In [36]: len(test)
```

```
Out[36]: 2876
```

```
In [43]: x_input = test[2865:].reshape(1,-1)
         x_input.shape
```

```
Out[43]: (1, 11)
```

```
In [44]: temp_input = list(x_input[0])
         temp_list = temp_input[0].tolist()
```

```
In [45]: temp_input
```

```
Out[45]: [0.9985685415964971,
          0.9988211519029975,
          0.9989053553384977,
          0.9989895587739979,
          0.9990737622094981,
          0.9991579656449983,
          0.9994105759514988,
          0.9994947793869988,
          0.9996631862579992,
          0.9997473896934994,
          1.0]
```

Data

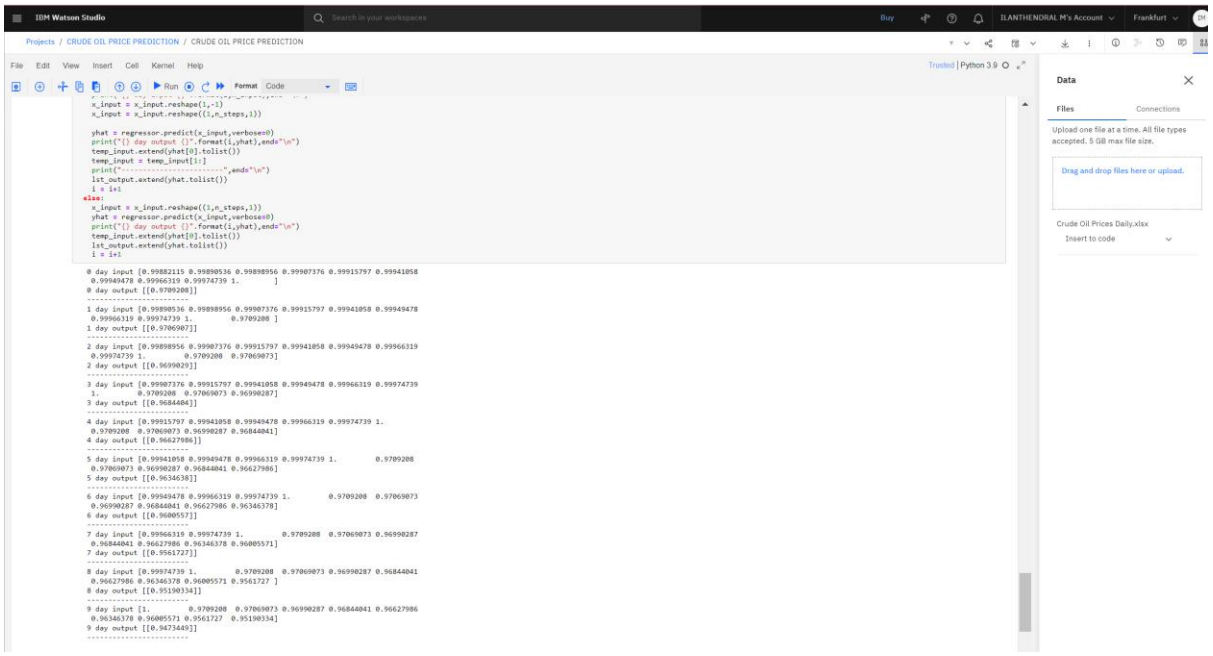
Files Connections

Upload one file at a time. All file types accepted. 5 GB max file size.

Drag and drop files here or upload.

Crude Oil Prices Daily.xlsx

Insert to code


IBM Watson Studio
Search in your workspaces
Buy
ILANTHENDRAL M's Account
Frankfurt
Projects / CRUDE OIL PRICE PREDICTION / CRUDE OIL PRICE PREDICTION
File Edit View Insert Cell Kernel Help
Trusted | Python 3.9
Run
Format Code
x_input = x_input.reshape(1,-1)
x_input = x_input.reshape((1,n_steps,1))
yhat = regressor.predict(x_input,verbose=0)
print("{} day output {}".format(i,yhat),ends="\n")
temp_input.extend(yhat[0].tolist())
temp_input = temp_input[1:]
print("-----------",ends="\n")
lst_output.extend(yhat.tolist())
i = i+1
else:
x_input = x_input.reshape((1,n_steps,1))
yhat = regressor.predict(x_input,verbose=0)
print("{} day output {}".format(i,yhat),ends="\n")
temp_input.extend(yhat[0].tolist())
lst_output.extend(yhat.tolist())
i = i+1
0 day input [0.99882115 0.99890536 0.99898956 0.99907376 0.99915797 0.99941058
0.99949478 0.99966319 0.99974739 1.]
0 day output [[0.9709208]]
1 day input [0.99890536 0.99898956 0.99907376 0.99915797 0.99941058 0.99949478
0.99966319 0.99974739 1. 0.9709208]
1 day output [[0.97069073]]
2 day input [0.99898956 0.99907376 0.99915797 0.99941058 0.99949478 0.99966319
0.99974739 1. 0.9709208 0.97069073]
2 day output [[0.9699029]]
3 day input [0.99907376 0.99915797 0.99941058 0.99949478 0.99966319 0.99974739
1. 0.9709208 0.97069073 0.96990287]
3 day output [[0.9684404]]
4 day input [0.99915797 0.99941058 0.99949478 0.99966319 0.99974739 1.
0.9709208 0.97069073 0.96990287 0.96844041]
4 day output [[0.96627986]]
5 day input [0.99941058 0.99949478 0.99966319 0.99974739 1. 0.9709208
0.97069073 0.96990287 0.96844041 0.96627986]
5 day output [[0.9634638]]
6 day input [0.99949478 0.99966319 0.99974739 1. 0.9709208 0.97069073
0.96990287 0.96844041 0.96627986 0.96346378]
6 day output [[0.9600557]]
7 day input [0.99966319 0.99974739 1. 0.9709208 0.97069073 0.96990287
0.96844041 0.96627986 0.96346378 0.96005571]
7 day output [[0.9561727]]
8 day input [0.99974739 1. 0.9709208 0.97069073 0.96990287 0.96844041
0.96627986 0.96346378 0.96005571 0.9561727]
8 day output [[0.95190334]]
9 day input [1. 0.9709208 0.97069073 0.96990287 0.96844041 0.96627986
0.96346378 0.96005571 0.9561727 0.95190334]
9 day output [[0.9473449]]
Data
Files
Connections
Upload one file at a time. All file types accepted. 5 GB max file size.
Drag and drop files here or upload.
Crude Oil Prices Daily.xlsx
Insert to code