Data collection and preprocessing

November 15, 2024

1 TEAD ID: SWTID1726659740

1.0.1 Project Name- Crude oil price prediction

2 DATA PREPROCESSING

2.1 Importing the libraries

[18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pyt
import seaborn as sns
import tensorflow as tf
```

[19]:
```
pwd
```

[19]:
```
'C:\\Users\\lenovo\\Desktop'
```

[20]:
```
import pandas as pd
data= pd.read_csv(r"C:\Users\lenovo\Downloads\Crude Oil Prices Daily.csv")
print(data.head())
```

```
       Date  Closing Value
0  1/2/1986          25.56
1  1/3/1986          26.00
2  1/6/1986          26.53
3  1/7/1986          25.85
4  1/8/1986          25.87
```

2.2 Analyze the Data

[21]:
```
data.head()
```

[21]:
```
        Date  Closing Value
0  1/2/1986          25.56
1  1/3/1986          26.00
2  1/6/1986          26.53
3  1/7/1986          25.85
4  1/8/1986          25.87
```

[22]:
```
data.tail()
```

[22]:
```
           Date  Closing Value
8218  7/3/2018          74.19
8219  7/4/2018            NaN
8220  7/5/2018          73.05
8221  7/6/2018          73.78
8222  7/9/2018          73.93
```

[23]:
```
data.describe()
```

[23]:
```
       Closing Value
count    8216.000000
mean       43.492139
std        29.616804
min        10.250000
25%        19.577500
50%        29.610000
75%        63.402500
max       145.310000
```

[24]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8223 entries, 0 to 8222
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           8223 non-null   object
 1   Closing Value  8216 non-null   float64
dtypes: float64(1), object(1)
memory usage: 128.6+ KB
```

2.3 Handling missing values

[25]:
```
data.isnull().any()
```

[25]:
```
Date             False
Closing Value     True
dtype: bool
```

[26]:
```
data.isnull().sum()
```

[26]:
```
Date             0
Closing Value    7
dtype: int64
```

[27]:
```
data.dropna(axis=0,inplace=True)
```

[28]:
```
data.isnull().sum()
```

[28]:
```
Date             0
Closing Value    0
dtype: int64
```

[29]:
```
data_oil = data.reset_index()['Closing Value']
```

[30]:
```
data_oil
```

[30]:
```
0       25.56
1       26.00
2       26.53
3       25.85
4       25.87
        ...
8211    73.89
8212    74.19
8213    73.05
8214    73.78
8215    73.93
Name: Closing Value, Length: 8216, dtype: float64
```

[31]:
```
print(data_oil.isnull().sum())
print(data_oil.shape)
```

```
0
(8216,)
```

[32]:
```
data_oil.dropna(inplace=True)
print(data_oil.isnull().sum())
print(data_oil.shape)
```

```
0
(8216,)
```

[33]:
```
print(data_oil.isnull().any())
```

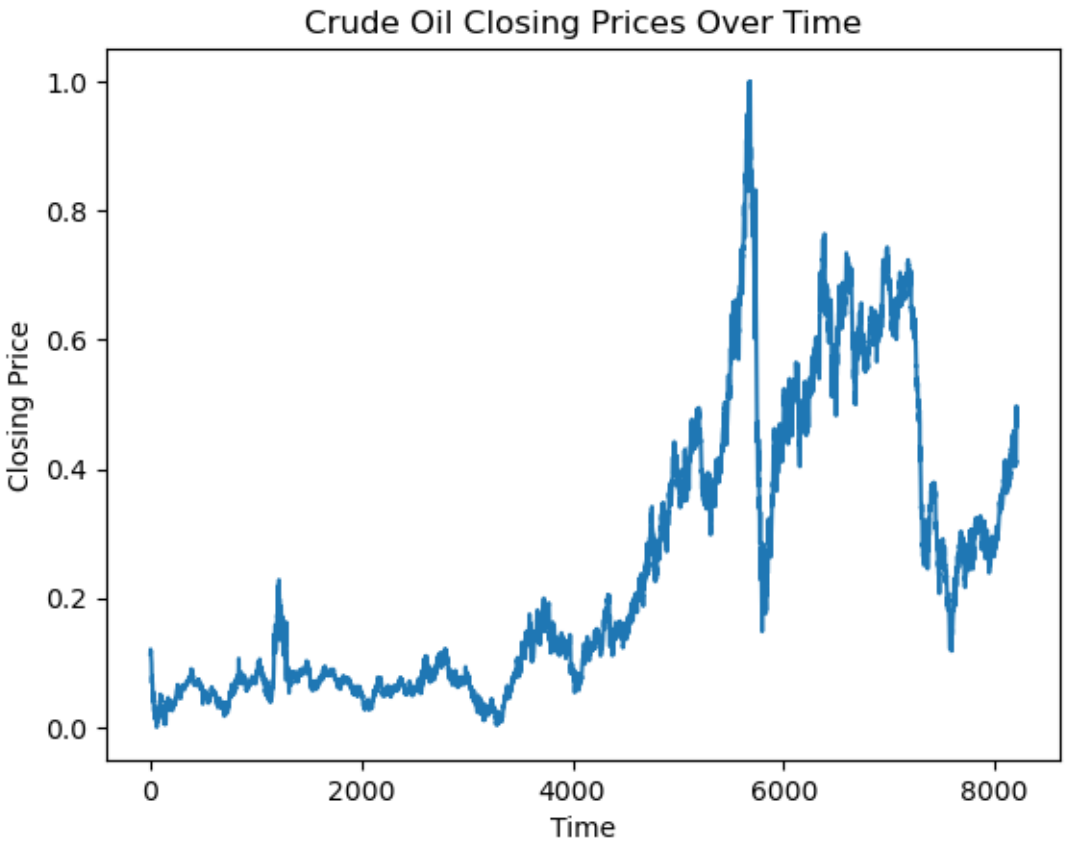
```
False
```

2.4 Feature Scaling

[34]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
data_oil = scaler.fit_transform(np.array(data_oil).reshape(-1, 1))
```

2.5 Data Visualization

[36]:
```
import matplotlib.pyplot as plt  # Use plt as the standard alias
plt.title('Crude Oil Closing Prices Over Time')
plt.plot(data_oil)
plt.xlabel('Time')
plt.ylabel('Closing Price')
plt.show()
```



2.6 Splitting Data into Train and Test

[37]:
```
training_size = int(len(data_oil)*0.65)
test_size = len(data_oil) - training_size
train_data,test_data = data_oil[0:training_size,:],data_oil[training_size:
 ↪len(data_oil),:1]
```

[38]:
```
training_size, test_size
(5340, 2876)
```

[38]: (5340, 2876)

[39]:
```
train_data.shape
(5340)
```

[39]:
```
5340
```

[40]:
```
import numpy as np

def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - time_step - 1):
        a = dataset[i:(i + time_step), 0]
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)
```

[41]:
```
time_step = 10
X_train, y_train = create_dataset(train_data, time_step)
X_test, ytest = create_dataset(test_data, time_step)
```

[42]:
```
print(X_train.shape,y_train.shape)
```

```
(5329, 10) (5329,)
```

[43]:
```
print(X_test.shape,y_train.shape)
```

```
(2865, 10) (5329,)
```

[44]:
```
X_train
```

[44]:
```
array([[0.11335703, 0.11661484, 0.12053902, …, 0.10980305, 0.1089886 ,
        0.11054346],
       [0.11661484, 0.12053902, 0.11550422, …, 0.1089886 , 0.11054346,
        0.10165852],
       [0.12053902, 0.11550422, 0.1156523 , …, 0.11054346, 0.10165852,
        0.09906708],
       …,
       [0.36731823, 0.35176958, 0.36080261, …, 0.36391234, 0.37042796,
        0.37042796],
       [0.35176958, 0.36080261, 0.35354657, …, 0.37042796, 0.37042796,
        0.37879461],
       [0.36080261, 0.35354657, 0.35295424, …, 0.37042796, 0.37879461,
        0.37916482]])
```

[ ]:
```
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)
```