DAY-10 ¶

Sales

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\sales.csv")[0:500]
df
```

Out[2]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0
...	...	...	...	...	...	...	...	...	...
495	10.2016	1.0	Italy	64983.0	Milano	3.0	other	47.205	0.0
496	10.2016	1.0	Italy	64983.0	Milano	4.0	Fish	2451.513	0.0
497	10.2016	1.0	Italy	64983.0	Milano	5.0	Fruits & Vegetables	1944.846	0.0
498	10.2016	1.0	Italy	64983.0	Milano	6.0	Meat	11980.629	122.0
499	10.2016	1.0	Italy	64983.0	Milano	13.0	Food	23665.44	122.0

500 rows × 14 columns

In [3]:
```
df.head(10)
```

Out[3]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	3
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	4
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	3
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	1
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0	17
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.0	31
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.0	2
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	0.0	
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	0.0	

In [4]:
```
df.describe()
```

Out[4]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	500.0	500.000000	500.000000	500.000000	5.000000e+02	5.000000e+02	0.0
mean	1.0	57412.764000	9.406000	31.520000	9.397837e+05	3.153113e+06	NaN
std	0.0	32104.273482	5.350366	142.134408	1.486945e+06	5.165524e+06	NaN
min	1.0	15552.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	NaN
25%	1.0	20891.000000	5.000000	0.000000	5.200250e+04	2.345122e+05	NaN
50%	1.0	71991.000000	9.000000	0.000000	2.555375e+05	7.053345e+05	NaN
75%	1.0	88253.000000	14.000000	0.000000	8.903900e+05	2.542147e+06	NaN
max	1.0	96857.000000	18.000000	1896.000000	7.476680e+06	2.571973e+07	NaN

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      500 non-null    object 
 1   Time index     500 non-null    float64
 2   Country        500 non-null    object 
 3   StoreID        500 non-null    float64
 4   City           500 non-null    object 
 5   Dept_ID        500 non-null    float64
 6   Dept. Name     500 non-null    object 
 7   HoursOwn       500 non-null    object 
 8   HoursLease     500 non-null    float64
 9   Sales units    500 non-null    float64
 10  Turnover       500 non-null    float64
 11  Customer       0 non-null      float64
 12  Area (m2)      500 non-null    object 
 13  Opening hours  500 non-null    object 
dtypes: float64(7), object(7)
memory usage: 54.8+ KB
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours'],
      dtype='object')
```

In [7]:
```
x=df[[ 'Time index',  'StoreID', 'Dept_ID', 'HoursLease', 'Sales units']]
y=df[ 'Turnover']
```

In [8]:
```
#to split my dataset into traning and test data

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [9]:
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
```

Out[9]:
```
LinearRegression()
```

In [10]:
```
print(lr.intercept_)
```

```
-70712.87685843837
```

In [11]:
```
print(lr.score(x_test,y_test))
```

```
0.9415037948859145
```

In [12]:
```
lr.score(x_train,y_train)
```

Out[12]:
```
0.8492861847269577
```

Ridge Regression

In [13]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [14]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[14]:
```
Ridge(alpha=10)
```

In [15]:
```
rr.score(x_test,y_test)
```

Out[15]:
```
0.9415032800366477
```

Lasso Regression

In [16]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[16]:
```
Lasso(alpha=10)
```

In [17]:
```
la.score(x_test,y_test)
```

Out[17]:
```
0.9415037787691132
```

In [18]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[18]:
```
ElasticNet()
```

In [19]:
```
print(en.intercept_)
```

```
-68989.20192453172
```

In [20]:
```
print(en.coef_)
```

```
[0.00000000e+00 1.45422191e+00 1.06068309e+04 2.43623236e+02
 3.19580207e+00]
```

In [30]:
```
predict=(en.predict(x_test))
```

In [22]:
```
print(en.score(x_test,y_test))
```

```
0.9414949041109082
```

Evaluation Metrix

In [27]:
```
from sklearn import metrics
```

In [31]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute Error: 674863.1452323053
```

In [32]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Square Error: 1409599193799.0828
```

In [34]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,predi
```

```
Root Mean Square Error: 1187265.42685243
```

In [ ]:
```

```