

DAY-9

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	31
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	1
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	41
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	31
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	11
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	1
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	0.0	1

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

In [14]: *#to split my dataset into training 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 [15]: `from sklearn.linear_model import LinearRegression`

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

Out[15]: LinearRegression()

In [16]: `print(lr.intercept_)`

-68742.91895086737

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

```
0.7073917318410916
```

```
In [18]: lr.score(x_train,y_train)
```

```
Out[18]: 0.9187980321742718
```

Ridge Regression

```
In [19]: from sklearn.linear_model import Ridge,Lasso
```

```
In [20]: rr=Ridge(alpha=10)  
rr.fit(x_train,y_train)
```

```
Out[20]: Ridge(alpha=10)
```

```
In [21]: rr.score(x_test,y_test)
```

```
Out[21]: 0.7073961651647841
```

Lasso Regression

```
In [23]: la=Lasso(alpha=10)  
la.fit(x_train,y_train)
```

```
Out[23]: Lasso(alpha=10)
```

```
In [24]: la.score(x_test,y_test)
```

```
Out[24]: 0.7073917818118092
```

```
In [ ]:
```