DAY-10

Iris

In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

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

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [3]:

```python
df.head(10)
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

In [4]:

```python
df.describe()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [5]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [8]:

```
x=df[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]
y=df['PetalWidthCm']
```

In [9]:

```
#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 [10]:

```
from sklearn.linear_model import LinearRegression

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

Out[10]:

```
LinearRegression()
```

In [11]:

```
print(lr.intercept_)
```

```
-0.3995021843654347
```

In [12]:

```
print(lr.score(x_test,y_test))
```

```
0.9370898687255358
```

In [13]:

```
lr.score(x_train,y_train)
```

Out[13]:

```
0.9488841211306531
```

Ridge Regression

In [14]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [15]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[15]:

```
Ridge(alpha=10)
```

In [16]:

```
rr.score(x_test,y_test)
```

Out[16]:

```
0.9259706159482798
```

Lasso Regression

In [17]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[17]:
```
Lasso(alpha=10)
```

In [18]:
```
la.score(x_test,y_test)
```

Out[18]:
```
0.6789035012371226
```

In [19]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[19]:
```
ElasticNet()
```

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

```
0.025257675016l9569
```

In [21]:
```
print(en.coef_)
```

```
[0.01530862 0.         0.         0.        ]
```

In [22]:
```
predict=(en.predict(x_test))
```

In [23]:
```
print(en.score(x_test,y_test))
```

```
0.7904357858002837
```

Evaluation Metrix

In [24]:
```
from sklearn import metrics
```

In [25]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute Error: 0.29976426633626574
```

In [26]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Square Error: 0.1276385773952405
```

In [27]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,predict))
```

```
Root Mean Square Error: 0.3572654158958582
```

In [ ]:
```
```