kaviyadevi 20106064

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

In [2]:
```
#to import dataset
data=pd.read_csv(r"C:\Users\user\Downloads\14_Iris - 14_Iris.csv")
data
```

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]:
```
#to display top 5 rows
data.head()
```

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

DATA CLEANING AND PREPROCESSING

In [4]:
```
#
data.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 [5]:
```
#to display summary of statistics(here to know min max value)
data.describe()
```

Out[5]:

	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 [6]:
```
#to display the column heading
data.columns
```

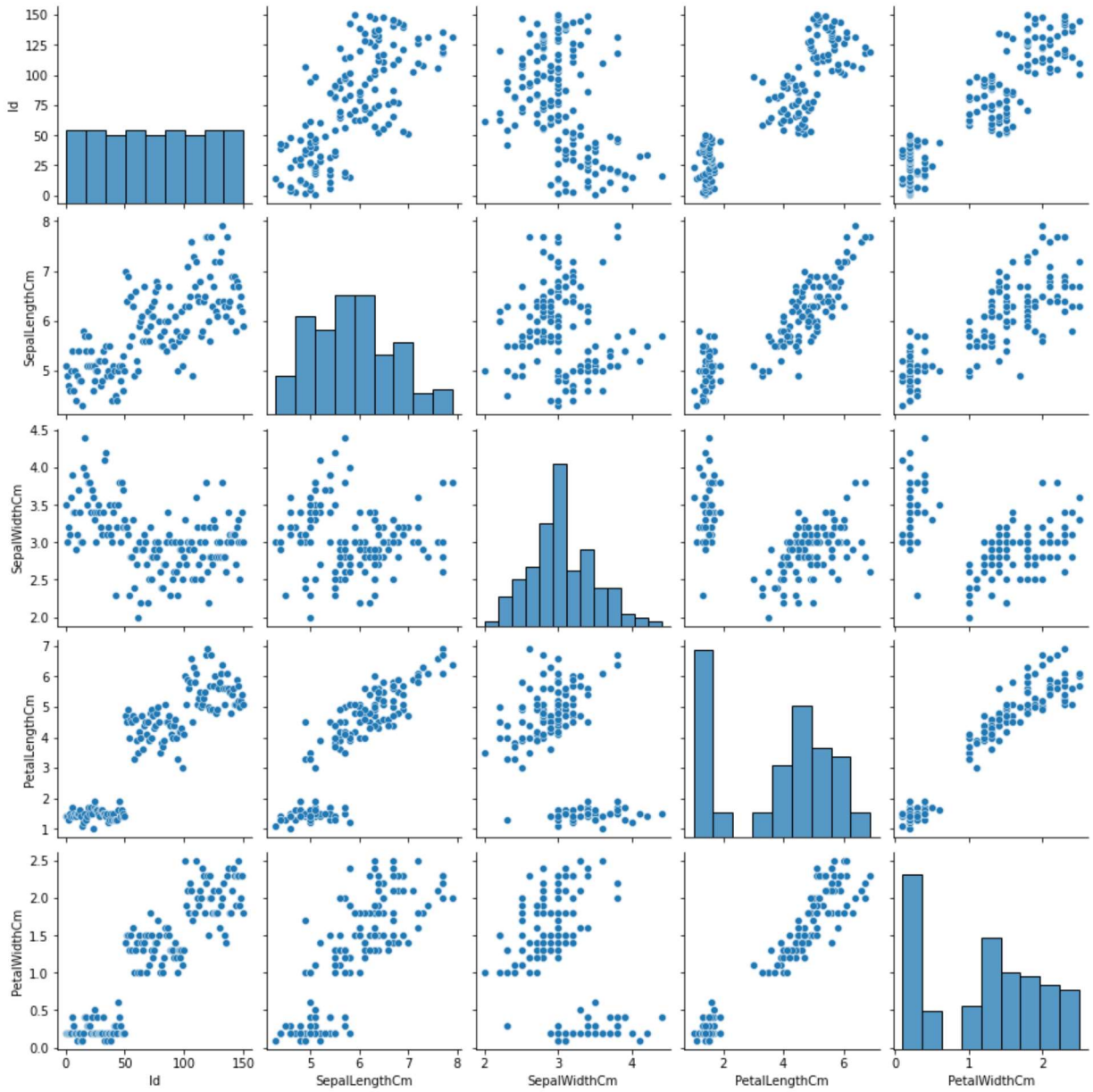
Out[6]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [7]:
```
#here there is no missing values (identified through info() 5000 data are describ
```

EDA and DATA VISUALIZATION

In [8]:
```
sns.pairplot(data)
```

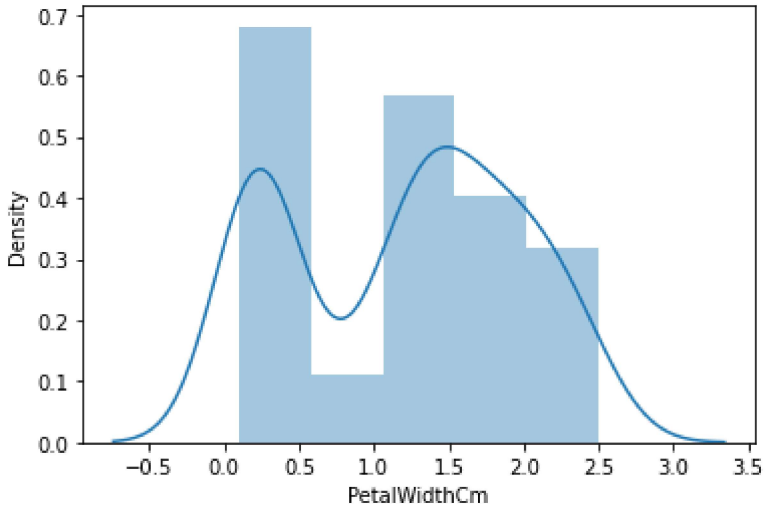
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1ece29f42b0>
```



In [9]:
```
sns.distplot(data['PetalWidthCm'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

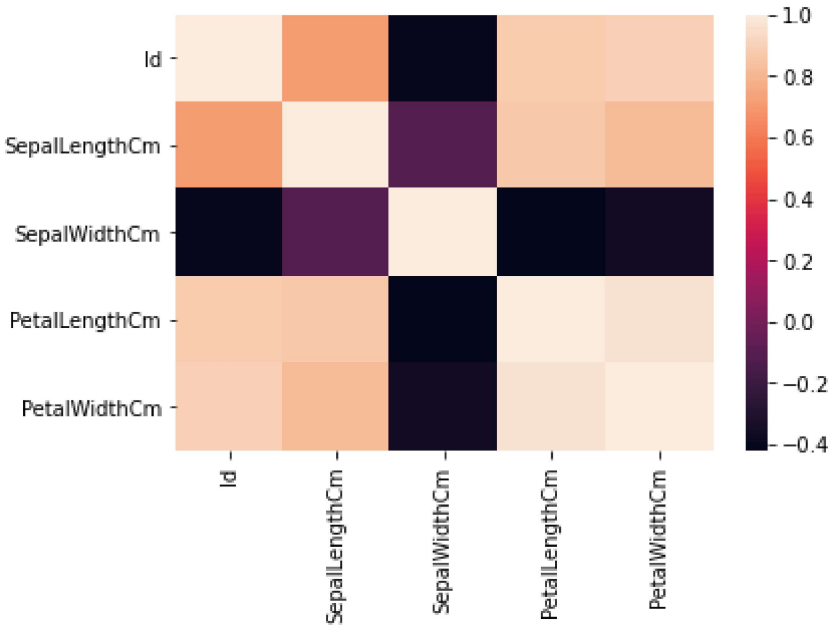
Out[9]: <AxesSubplot:xlabel='PetalWidthCm', ylabel='Density'>



In [10]:
```
df=data[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species']]
```

In [11]:
```
sns.heatmap(df.corr())
```

Out[11]: <AxesSubplot:>



TRAINING MODEL

In [12]:
```
x=df[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]
y=df['PetalWidthCm']
```

In [13]:
```
#to split my dataset into trainning and test

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 [14]:
```
from sklearn.linear_model import LinearRegression

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

Out[14]:
```
LinearRegression()
```

In [15]:
```
#to find intercept
print(lr.intercept_)
```

```
-0.48623284502873965
```

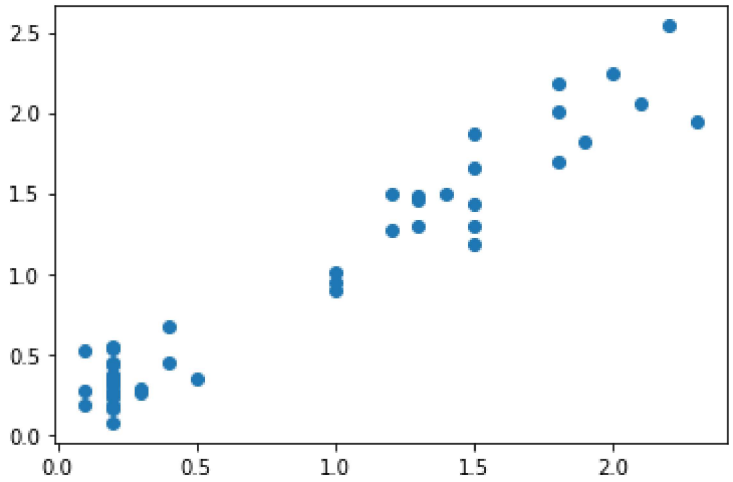
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Id	0.003894
SepalLengthCm	-0.205713
SepalWidthCm	0.313455
PetalLengthCm	0.441963

In [17]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1ece539d6a0>
```



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

```
0.9227710113297463
```

RIDGE AND LASSO 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]:
```
0.9336312567866796
```

In [22]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[22]:
```
Lasso(alpha=10)
```

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

Out[23]:
```
0.580268936212565
```

In [24]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[24]:
```
ElasticNet()
```

In [25]:
```
print(en.coef_)
```

```
[0.01509033 0.         0.         0.        ]
```

In [26]:
```
print(en.predict(x_test))
```

```
[1.87229275 0.4688925  2.03828633 0.12181502 0.54434413 1.13286681
 0.48398282 0.92160226 0.61979575 1.49503462 1.29886039 1.46485397
 1.99301536 0.21235697 0.74051835 1.55539592 0.96687323 0.89142161
 1.94774438 0.27271827 0.31798925 0.83106031 0.78578933 1.05741519
 0.10672469 0.68015705 0.60470543 2.11373796 0.5896151  1.01214421
 0.33307957 1.43467332 0.80087966 0.77069901 1.37431202 1.22340877
 1.85720243 1.7817508  0.45380217 0.51416347 2.29482186 0.55943445
 1.16304746 0.24253762 0.19726664]
```

In [27]:
```
print(en.score(x_test,y_test))
```

```
0.7764691528942963
```

In [28]:
```
from sklearn import metrics
```

In [29]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 0.16121836420927024
```

In [30]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 0.03962361978708484
```

In [31]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 0.19905682552247447
```