D11

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

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\14_Iris.csv")
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]:
```
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]:

```
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 [5]:

```
df.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]:

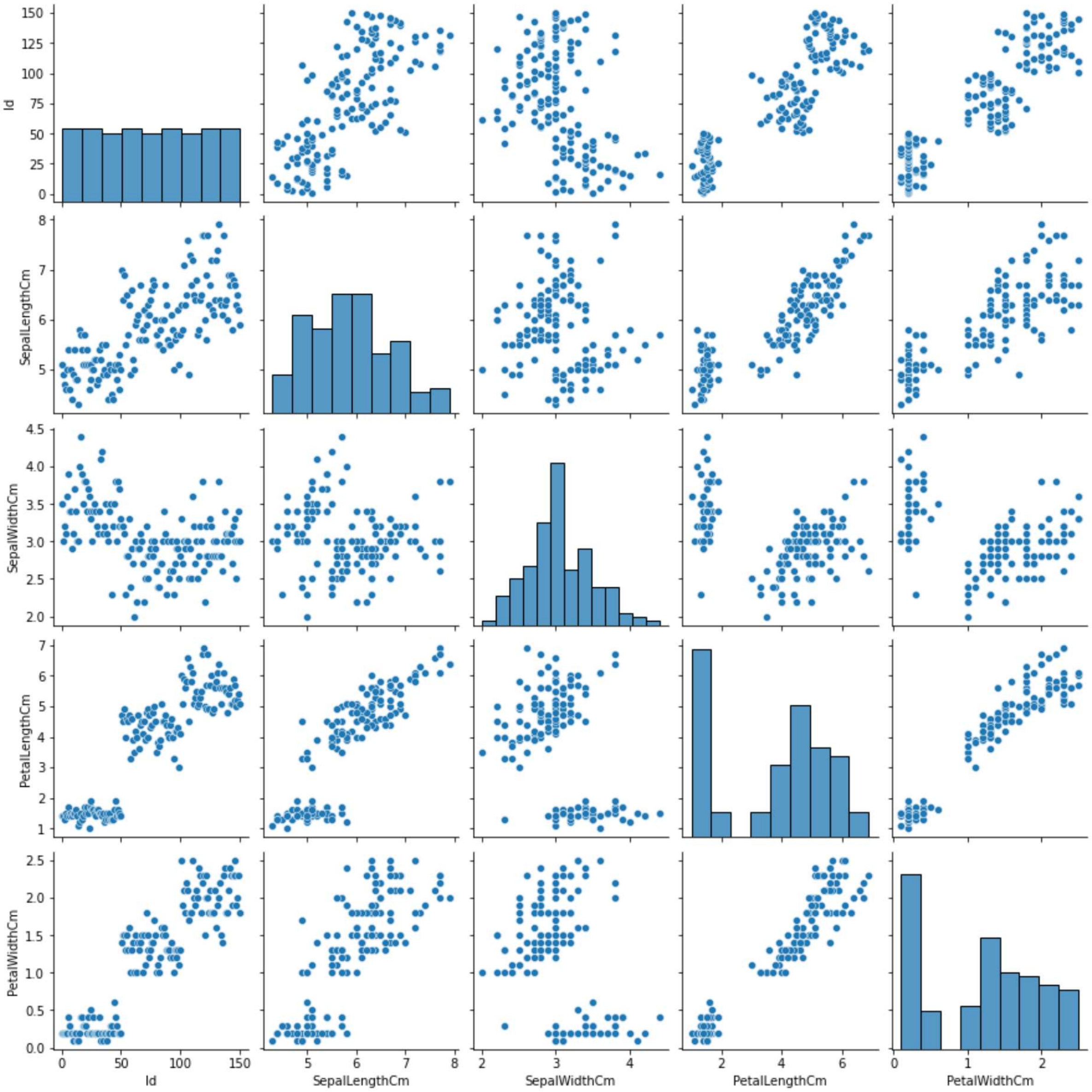
```
df.columns
```

Out[6]:

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

In [7]:
```
sns.pairplot(df)
```

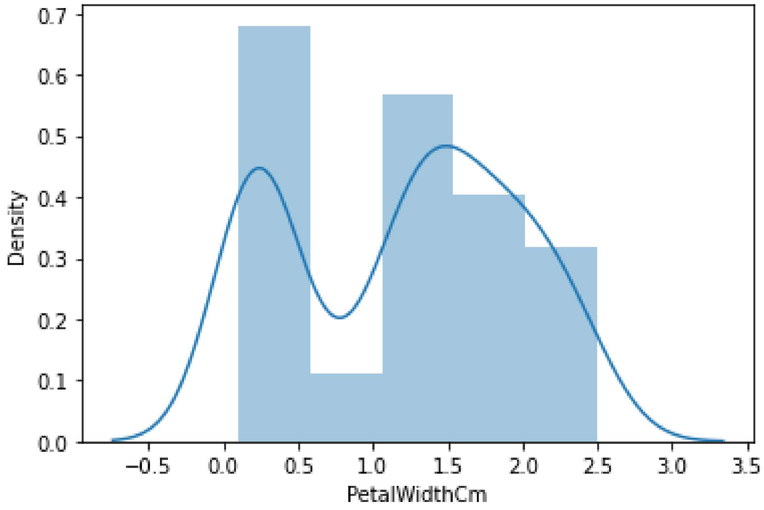
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x23856714a30>
```



In [8]:
```
sns.distplot(df['PetalWidthCm'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

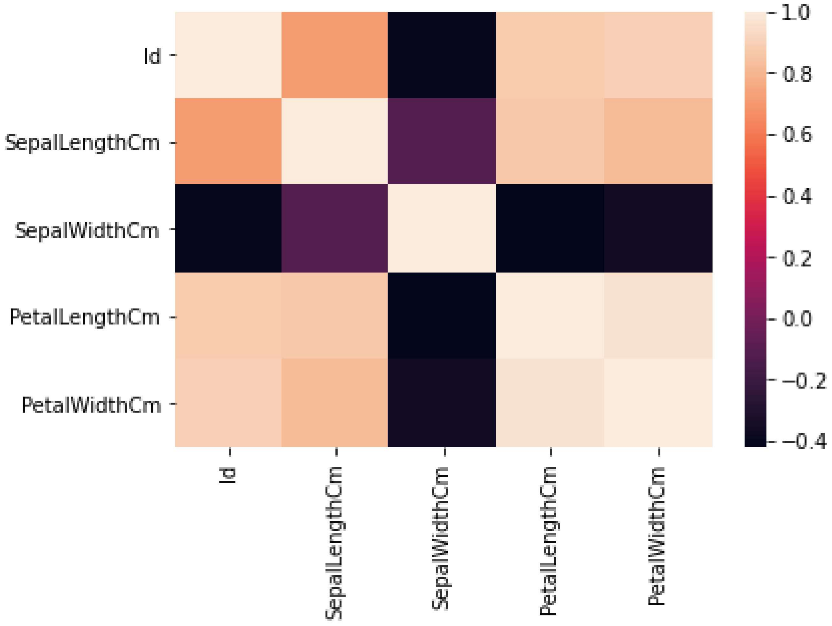
Out[8]:
```
<AxesSubplot:xlabel='PetalWidthCm', ylabel='Density'>
```



In [9]:
```
df1=df[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm','PetalWidthCm',
       'Species']]
```

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:

```
x=df1[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]
y=df1['PetalWidthCm']
```

In [12]:

```
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 [13]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:

```
LinearRegression()
```

In [14]:

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

```
-0.4867855184184695
```

In [15]:

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

Out[15]:

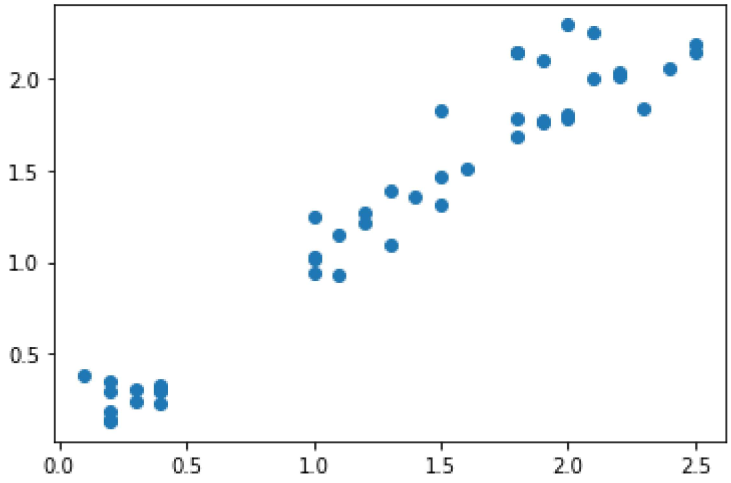
	Co-efficient
Id	0.003040
SepalLengthCm	-0.112235
SepalWidthCm	0.178449
PetalLengthCm	0.414370

In [16]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x238597626d0>
```



In [17]:

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

```
0.9350212507802496
```

```
In [18]: from sklearn.linear_model import Ridge,Lasso
```

```
In [19]: rr=Ridge(alpha=10)
         rr.fit(x_train,y_train)

Out[19]: Ridge(alpha=10)
```

```
In [20]: rr.score(x_test,y_test)

Out[20]: 0.923369768023156
```

```
In [21]: la=Lasso(alpha=10)
         la.fit(x_train,y_train)

Out[21]: Lasso(alpha=10)
```

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

Out[22]: 0.6962235191238506
```

```
In [23]: from sklearn.linear_model import ElasticNet
         en=ElasticNet()
         en.fit(x_train,y_train)

Out[23]: ElasticNet()
```

```
In [24]: print(en.coef_)

         [ 0.01549623  0.          -0.           0.        ]
```

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

         0.029301180477596667
```

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

         [0.38571437 2.0283143  0.80411247 0.43220305 0.75762379 1.09854076
          1.14502943 1.81136714 1.33098414 0.13777476 2.07480298 0.21525589
          0.63365398 1.88884827 0.29273702 1.4239615  2.32274259 2.19877278
          1.12953321 1.50144263 1.73388601 1.9663294  1.95083317 0.9435785
          2.12129166 0.60266153 0.86609737 1.84235959 1.23800679 0.55617286
          1.43945772 2.18327656 1.0055634  0.27724079 1.9043445  1.02105963
          0.81960869 0.24624834 0.4631955  1.11403698 1.19151811 1.37747282
          1.79587092 2.13678788 1.36197659 0.49418795 1.53243508 0.07578986
          0.32372947 0.50968418 0.47869173 2.04381053 1.57892375 2.29175014
          0.61815776 1.06754831 2.33823882 0.69563889 0.85060114 0.35472192
          1.45495395 1.29999169 1.78037469 0.72663134 1.16052566 0.30823324
          1.64090866 0.92808227 1.17602189 0.04479741 2.16778033 2.24526146
          2.15228411 0.19975966 1.85785582 0.18426344 0.23075212 1.82686337
          1.34648037 0.89708982 0.26174457 0.8815936  0.57166908 0.74212756
          0.41670682 1.39296905 1.71838979 0.15327099 2.26075769 0.06029363
          0.68014266 0.77312002 1.68739734 0.95907473 0.71113511 1.22251056
          1.05205208 1.87335204 0.83510492 1.5479313  0.10678231 0.4012106
          1.51693885 1.76487846 2.35373504]
```

In [27]:
```
print(en.score(x_train,y_train))
```
```
0.8132284760020183
```

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

In [29]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```
```
Mean Absolytre Error: 0.15169896989562648
```

In [30]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```
```
Mean Square Error: 0.03601138325894178
```

In [31]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```
```
Root Mean Square Error: 0.38948551949414817
```

In [ ]: