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\Admin\Downloads\14_Iris - 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.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 [4]:
```
df.describe()
```

Out[4]:

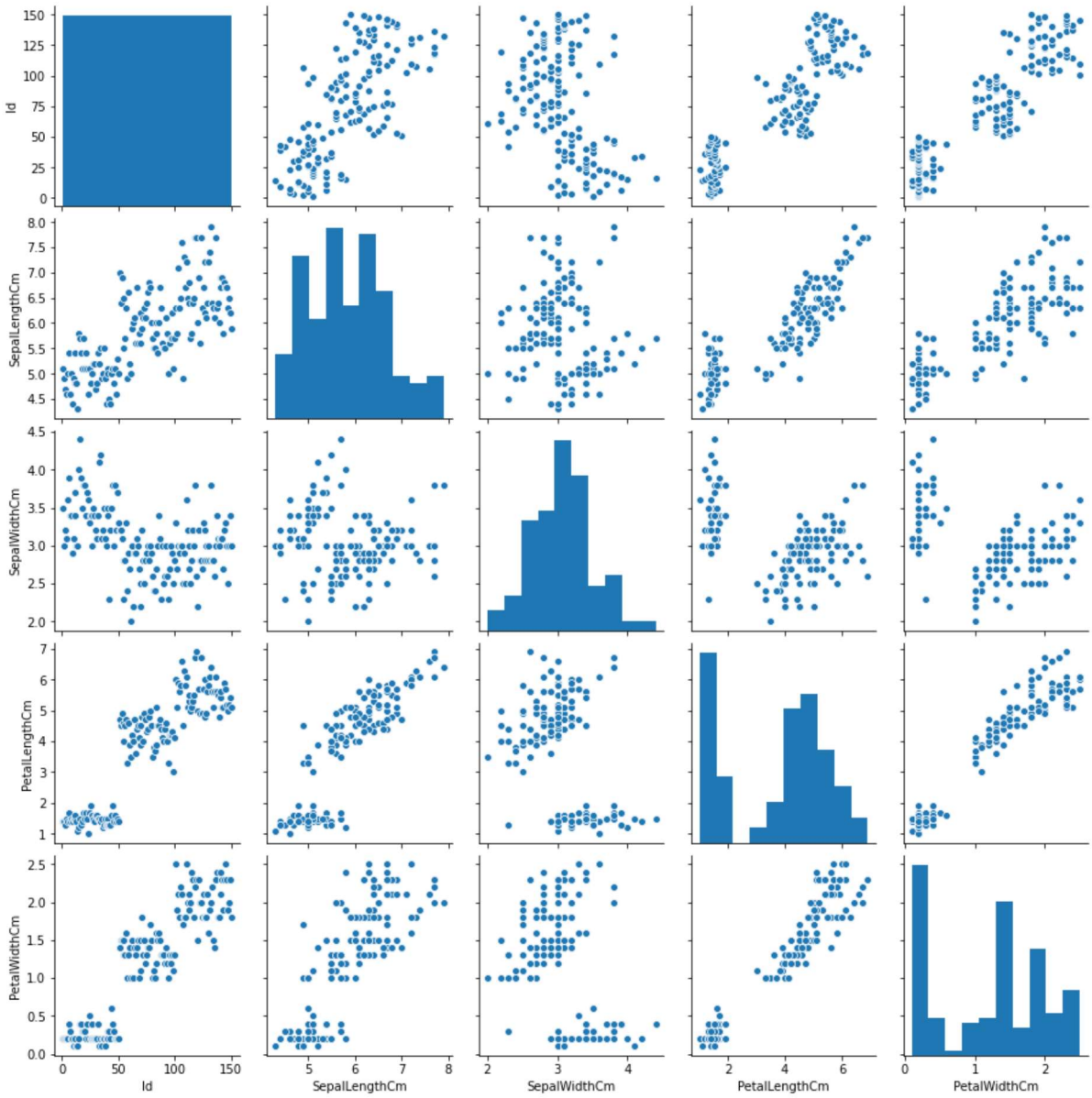
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	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]:
```
df.columns
```

Out[5]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthC
m',
       'Species'],
      dtype='object')
```

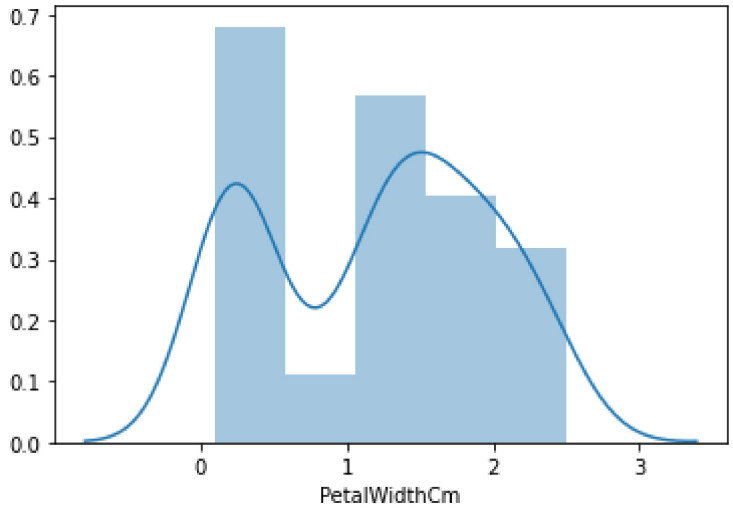
In [6]:
```
sns.pairplot(df)
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x1c037d2f490>
```



In [7]:
```
sns.distplot(df['PetalWidthCm'])
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c038d24ac0>
```



In [8]:
```
df1=df[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
df1
```

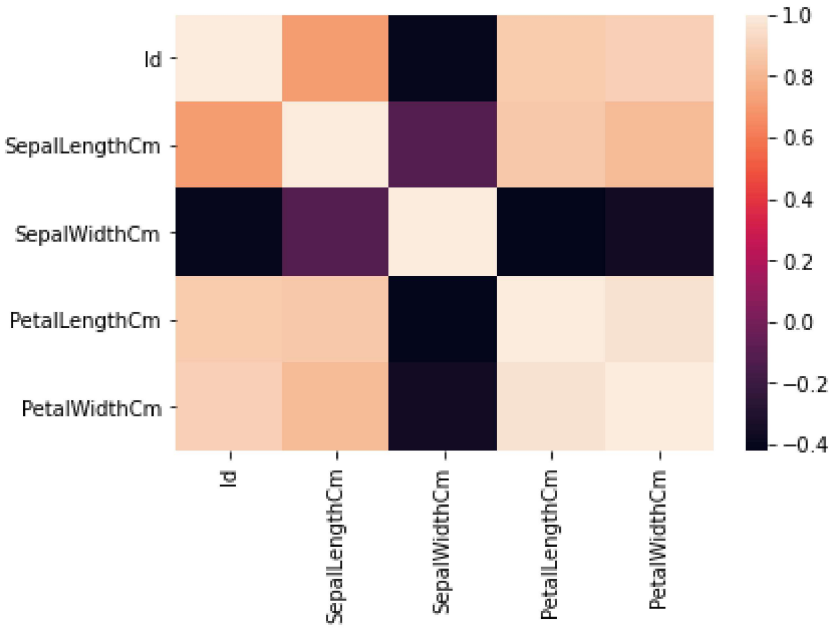
Out[8]:

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

150 rows × 5 columns

In [9]:
```
sns.heatmap(df1.corr())
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c038d9f340>
```



In [10]:
```
x=df1[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]
y=df1['PetalWidthCm']
```

In [11]:
```
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 [12]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

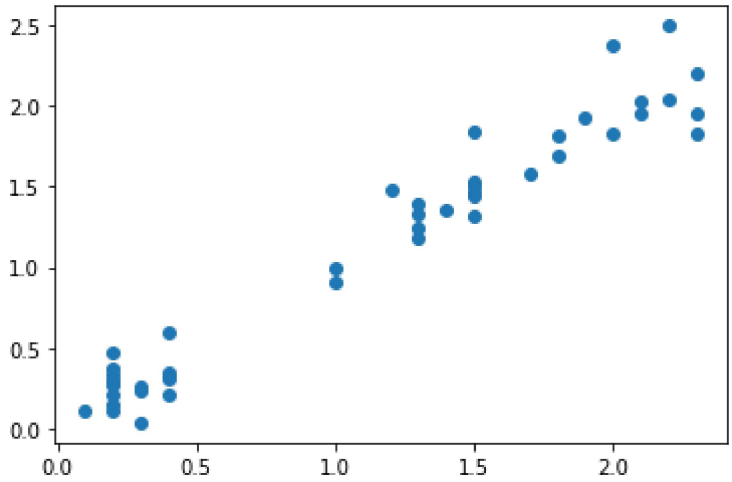
Out[12]:
```
LinearRegression()
```

In [13]:
```
print(lr.intercept_)
```

```
-0.4402916433474664
```

In [14]:
```
prediction= lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[14]: <matplotlib.collections.PathCollection at 0x1c0398832e0>



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

```
0.951417615821395
```

In [16]:
```
print(lr.score(x_train,y_train))
```

```
0.9428469851391287
```

In [17]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [18]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[18]: Ridge(alpha=10)

In [19]:
```
rr.score(x_test,y_test)
```

Out[19]: 0.953495417094863

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

Out[20]: Lasso(alpha=10)

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

Out[21]: 0.7015880975986795

In [22]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[22]:
```
ElasticNet()
```

In [23]:
```
print(en.intercept_)
```

```
0.0517356142336185
```

In [24]:
```
print(en.predict(x_test))
```

```
[2.04187999 1.39392601 2.25786466 0.71511707 2.11901737 1.00823911
 0.99281164 0.74597203 0.39114008 0.14430047 0.6996896  0.86939183
 0.15972794 2.02645252 1.87217776 0.60712474 0.83853688 0.96195669
 0.1905829  1.25507873 0.8231094  1.45563591 2.21158223 0.45284999
 2.24243718 1.28593368 2.08816242 0.2522928  0.77682698 0.80768193
 0.54541484 1.5944832  1.08537649 1.84132281 1.2705062  1.16251387
 0.68426212 2.35042951 1.40935349 2.1035899  1.96474262 1.76418543
 0.52998736 0.51455989 0.46827746]
```

In [25]:
```
print(en.score(x_test,y_test))
```

```
0.8046237828472149
```

Evaluation

In [26]:
```
from sklearn import metrics
```

In [27]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 0.12771879965444105
```

In [28]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 0.028313573586282736
```

In [29]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prec
```

```
Root Mean Squared Error: 0.16826637687393978
```

In [ ]:
```

```