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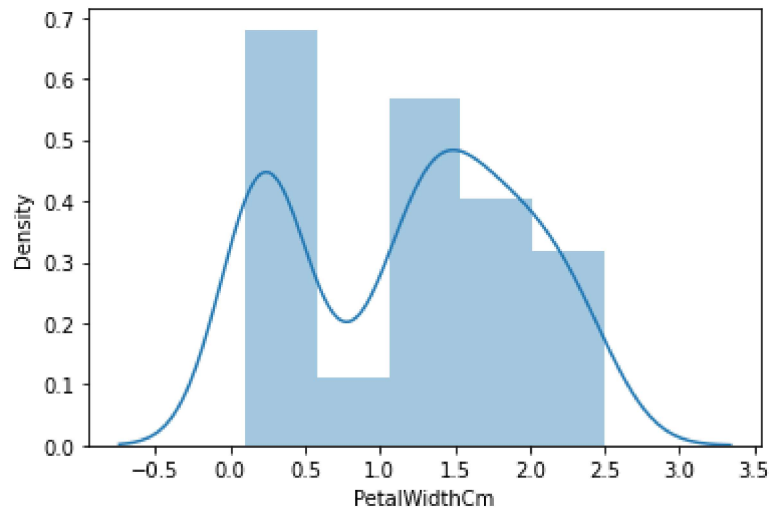
In [7]: `sns.pairplot(df)`

Out[7]: `<seaborn.axisgrid.PairGrid at 0x207e64daaf0>`
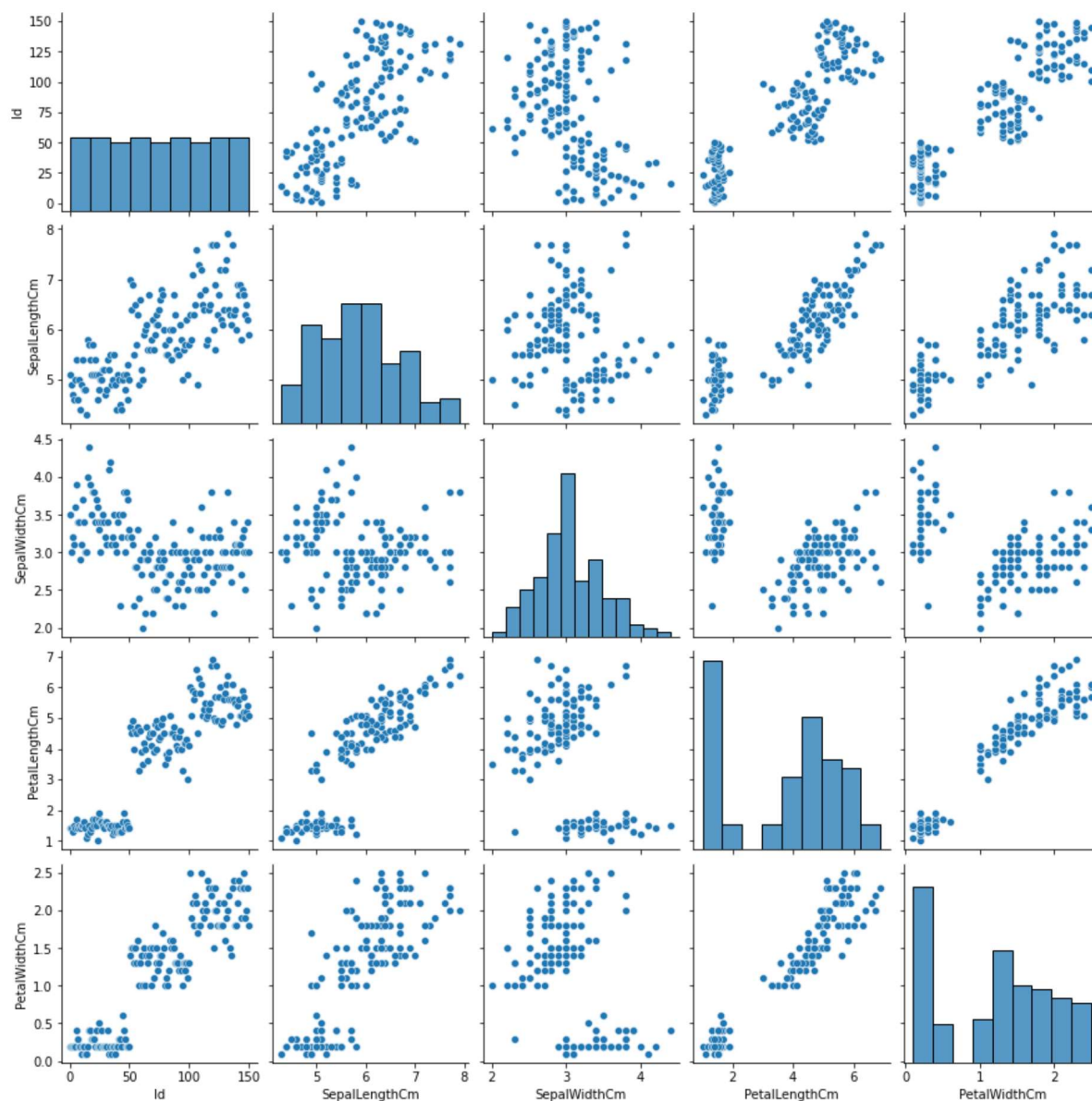
In [9]:
```python
sns.distplot(df['PetalWidthCm'])
```

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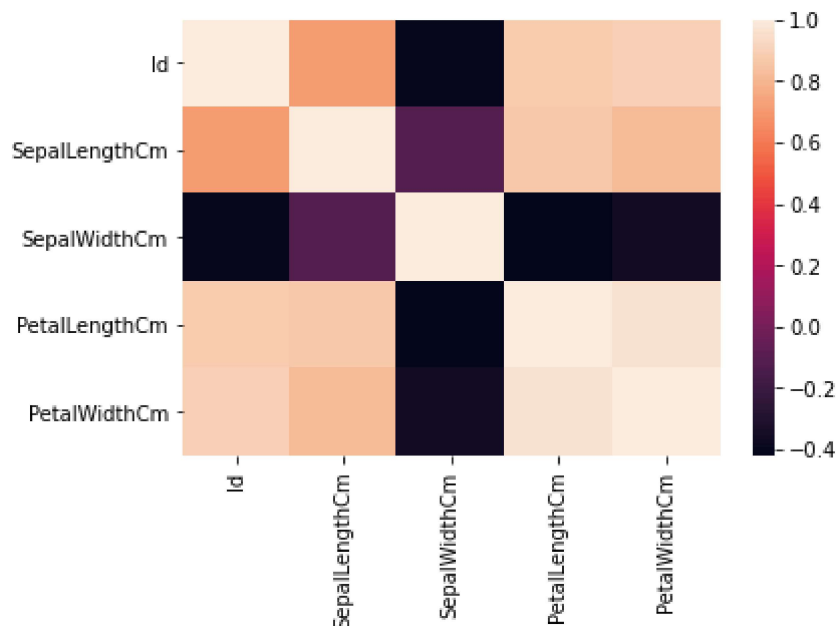
Out[9]: `<AxesSubplot:xlabel='PetalWidthCm', ylabel='Density'>`

In [11]:
```python
df1=df[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm','PetalWidthCm',
        'Species']]
```

In [14]:
```python
sns.heatmap(df1.corr())
```

Out[14]: `<AxesSubplot:>`

In [15]:
```python
x=df1[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]
y=df1['PetalWidthCm']
```

In [16]:
```python
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 [17]:
```python
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]: LinearRegression()

In [18]:
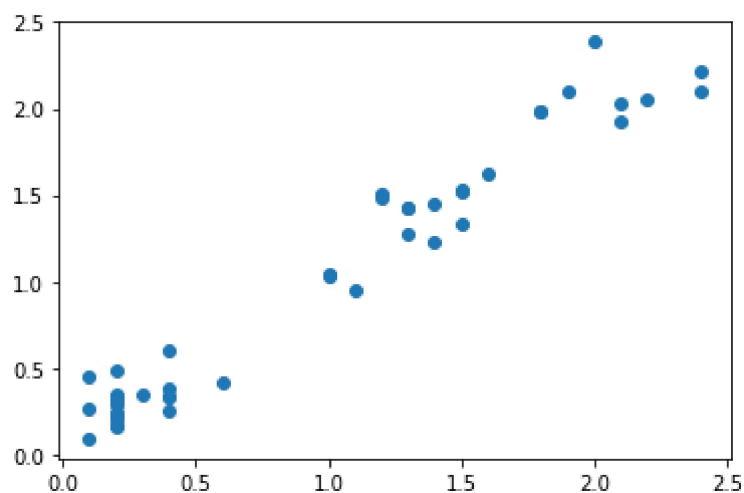```python
print(lr.intercept_)
```

-0.4591663093300591

In [19]:
```python
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[19]:

	Co-efficient
Id	0.003477
SepalLengthCm	-0.155259
SepalWidthCm	0.235201
PetalLengthCm	0.424976

In [20]:
```python
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[20]: <matplotlib.collections.PathCollection at 0x207e8f88dc0>

In [21]:
```python
print(lr.score(x_test,y_test))
```

0.9534293541372023

```python
In [22]: from sklearn.linear_model import Ridge,Lasso
```

```python
In [23]: rr=Ridge(alpha=10)
         rr.fit(x_train,y_train)
```

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

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

```
Out[24]: 0.9480048457961149
```

```python
In [25]: la=Lasso(alpha=10)
         la.fit(x_train,y_train)
```

```
Out[25]: Lasso(alpha=10)
```

```python
In [26]: la.score(x_test,y_test)
```

```
Out[26]: 0.6348819122699374
```

```python
In [ ]:
```