

In [1]:

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

In [2]:

```
df1=pd.read_csv(r'C:\Users\user\Downloads\14_Iris.csv')
df1
```

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=df1.head(100)
df
```

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
...
95	96	5.7	3.0	4.2	1.2	Iris-versicolor
96	97	5.7	2.9	4.2	1.3	Iris-versicolor
97	98	6.2	2.9	4.3	1.3	Iris-versicolor
98	99	5.1	2.5	3.0	1.1	Iris-versicolor
99	100	5.7	2.8	4.1	1.3	Iris-versicolor

100 rows × 6 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
#   Column              Non-Null Count  Dtype
---  -
0   Id                  100 non-null   int64
1   SepalLengthCm      100 non-null   float64
2   SepalWidthCm       100 non-null   float64
3   PetalLengthCm      100 non-null   float64
4   PetalWidthCm       100 non-null   float64
5   Species            100 non-null   object
dtypes: float64(4), int64(1), object(1)
memory usage: 4.8+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	100.000000	100.000000	100.000000	100.000000	100.000000
mean	50.500000	5.471000	3.094000	2.862000	0.785000
std	29.011492	0.641698	0.476057	1.448565	0.566288
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	25.750000	5.000000	2.800000	1.500000	0.200000
50%	50.500000	5.400000	3.050000	2.450000	0.800000
75%	75.250000	5.900000	3.400000	4.325000	1.300000
max	100.000000	7.000000	4.400000	5.100000	1.800000

In [6]:

```
df.columns
```

Out[6]:

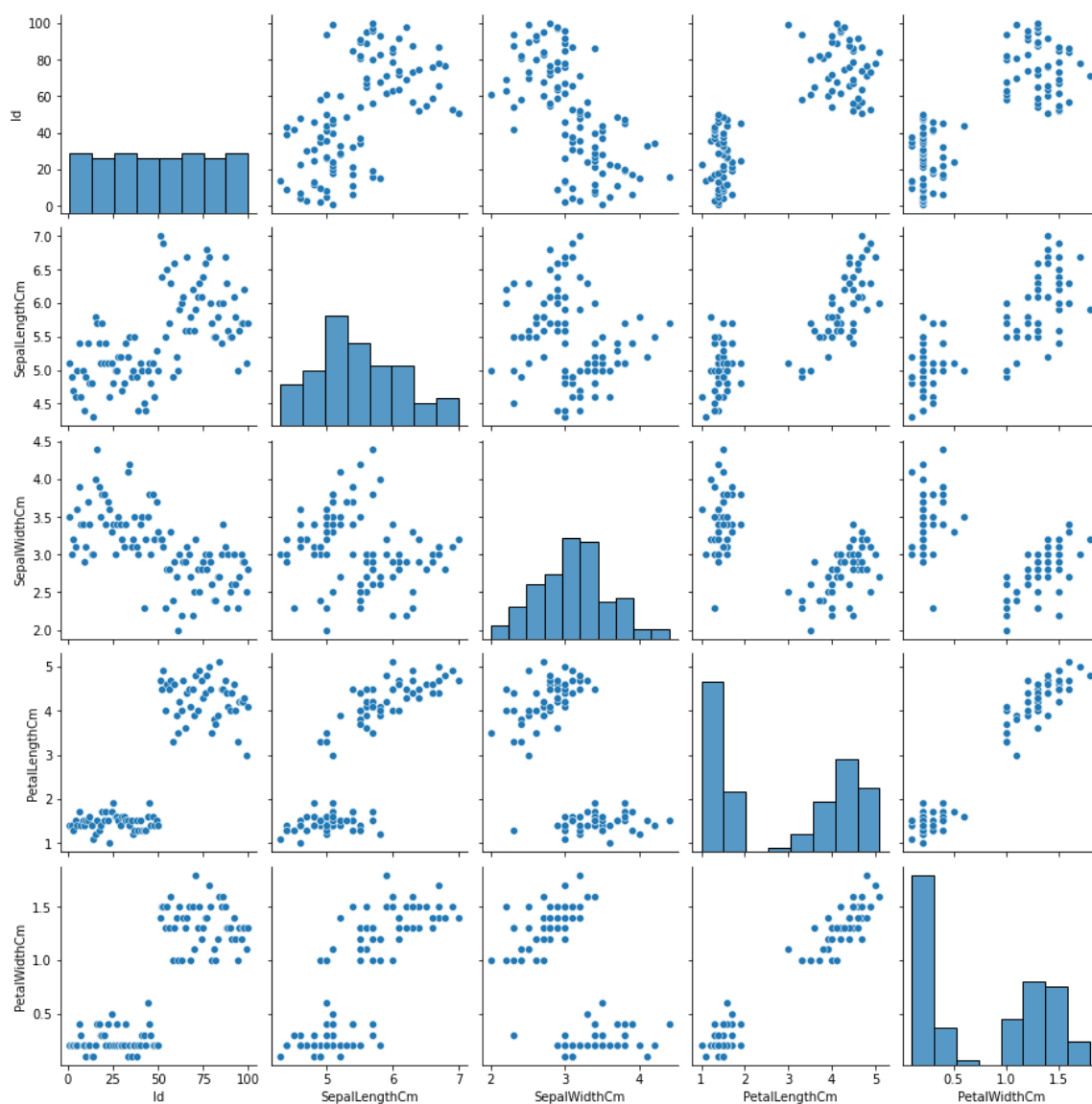
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',  
      'Species'],  
      dtype='object')
```

In [7]:

```
sns.pairplot(df)
```

Out[7]:

<seaborn.axisgrid.PairGrid at 0x25988c3ac10>



In [8]:

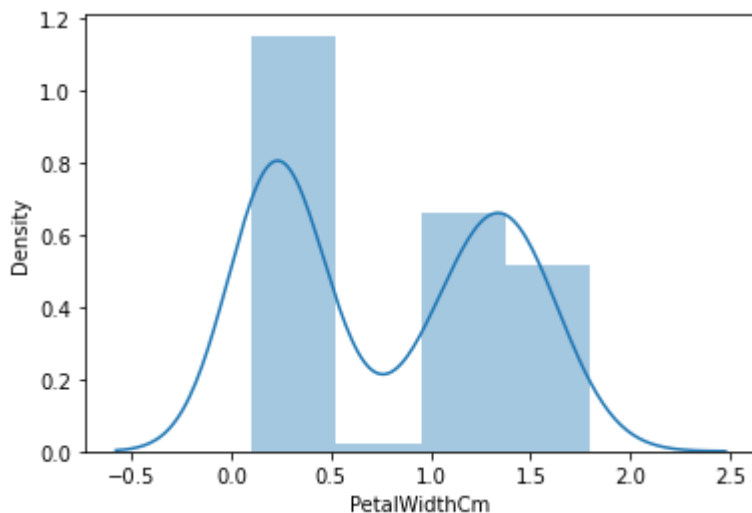
```
sns.distplot(df['PetalWidthCm'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).

```
warnings.warn(msg, FutureWarning)
```

Out[8]:

<AxesSubplot:xlabel='PetalWidthCm', ylabel='Density'>

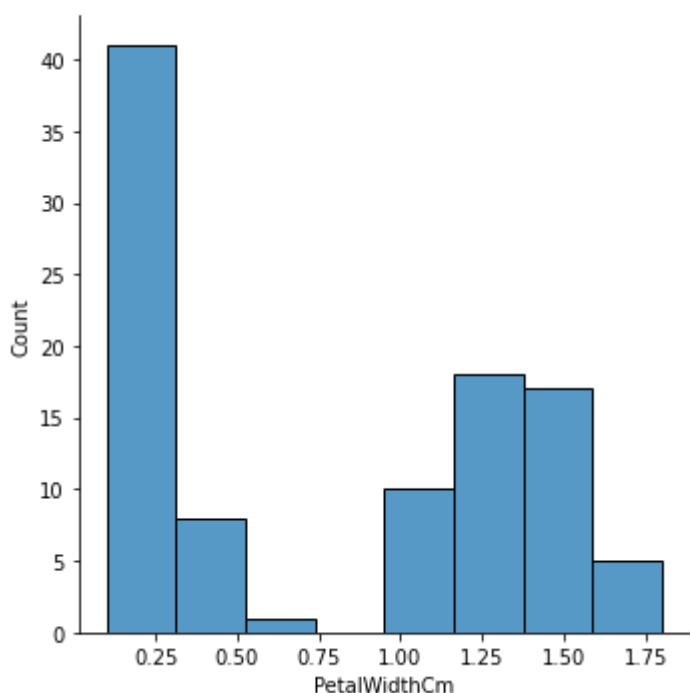


In [9]:

```
sns.displot(df["PetalWidthCm"])
```

Out[9]:

<seaborn.axisgrid.FacetGrid at 0x25988c3a5b0>



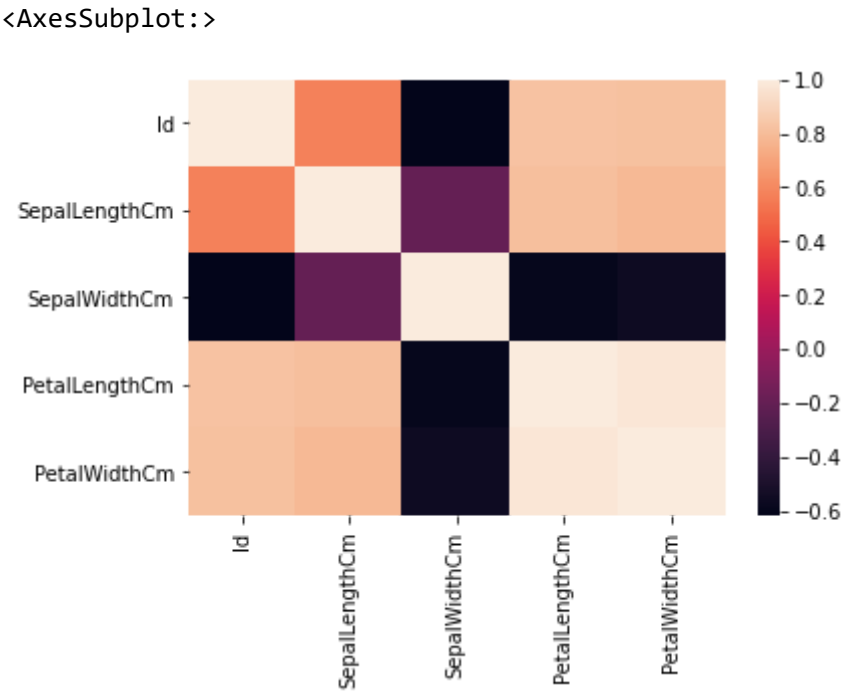
In [10]:

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

In [11]:

```
sns.heatmap(df1.corr())
```

Out[11]:



In [12]:

```
df2=df.dropna(axis=1)  
df2
```

Out[12]:

	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
...
95	96	5.7	3.0	4.2	1.2	Iris-versicolor
96	97	5.7	2.9	4.2	1.3	Iris-versicolor
97	98	6.2	2.9	4.3	1.3	Iris-versicolor
98	99	5.1	2.5	3.0	1.1	Iris-versicolor
99	100	5.7	2.8	4.1	1.3	Iris-versicolor

100 rows × 6 columns

In [13]:

```
x=df2[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]
y=df2[['PetalWidthCm']]
```

In [14]:

```
from sklearn.model_selection import train_test_split
```

In [15]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [16]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[16]:

```
LinearRegression()
```

In [17]:

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

```
[-0.34585664]
```

In [18]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[18]:

	0	1	2	3
0	0.001319	-0.094225	0.12072	0.425191

In [19]:

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

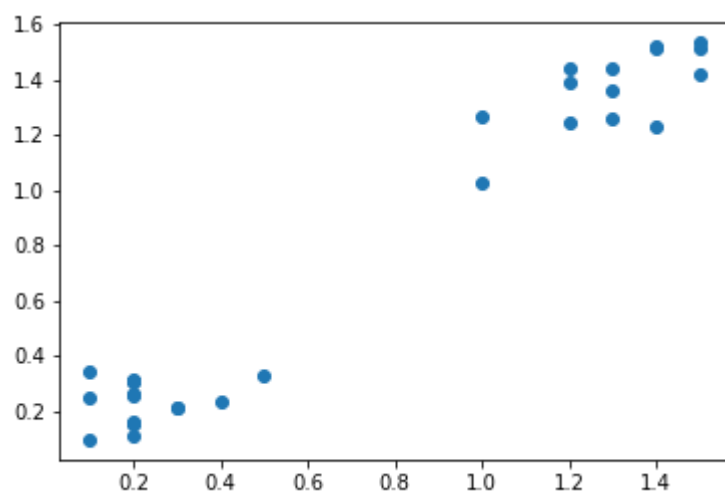
```
0.9487120195837342
```

In [20]:

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

Out[20]:

<matplotlib.collections.PathCollection at 0x2598bceaa60>



In [21]:

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

Out[21]:

0.9487120195837342

In [22]:

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

Out[22]:

0.9637225562865754

In [23]:

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

In [24]:

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

Out[24]:

Ridge(alpha=10)

In [25]:

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

Out[25]:

0.9447487439672552

In [26]:

```
la=Lasso(alpha=10)  
la.fit(x_train,y_train)
```

Out[26]:

Lasso(alpha=10)

In [27]:

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

Out[27]:

0.2823793935726967