mk 31-07-23

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

In [226]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\14_Iris.csv")
a
```

Out[226]:

	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 [227]:
```
a=a.head(10)
a
```

Out[227]:

	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 [228]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             10 non-null     int64  
 1   SepalLengthCm  10 non-null     float64
 2   SepalWidthCm   10 non-null     float64
 3   PetalLengthCm  10 non-null     float64
 4   PetalWidthCm   10 non-null     float64
 5   Species        10 non-null     object 
dtypes: float64(4), int64(1), object(1)
memory usage: 608.0+ bytes
```

In [229]:
```
a.columns
```

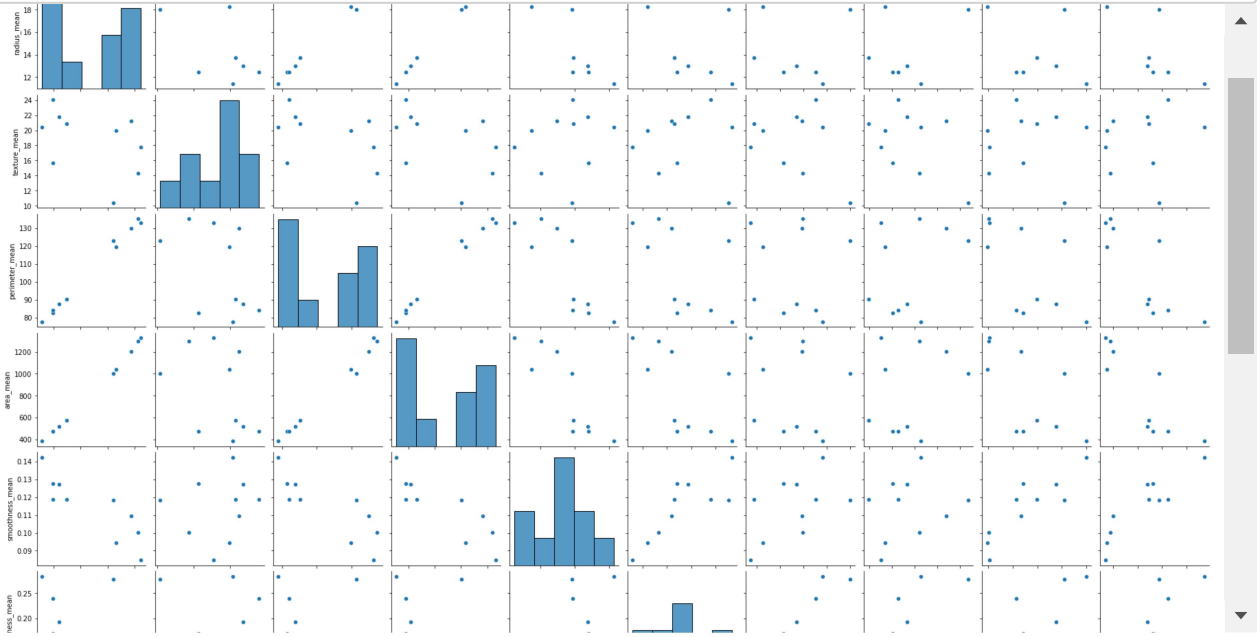
Out[229]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [230]:
```
a.describe()
```

Out[230]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	10.00000	10.000000	10.000000	10.000000	10.000000
mean	5.50000	4.860000	3.310000	1.450000	0.220000
std	3.02765	0.291357	0.307137	0.108012	0.078881
min	1.00000	4.400000	2.900000	1.300000	0.100000
25%	3.25000	4.625000	3.100000	1.400000	0.200000
50%	5.50000	4.900000	3.300000	1.400000	0.200000
75%	7.75000	5.000000	3.475000	1.500000	0.200000
max	10.00000	5.400000	3.900000	1.700000	0.400000

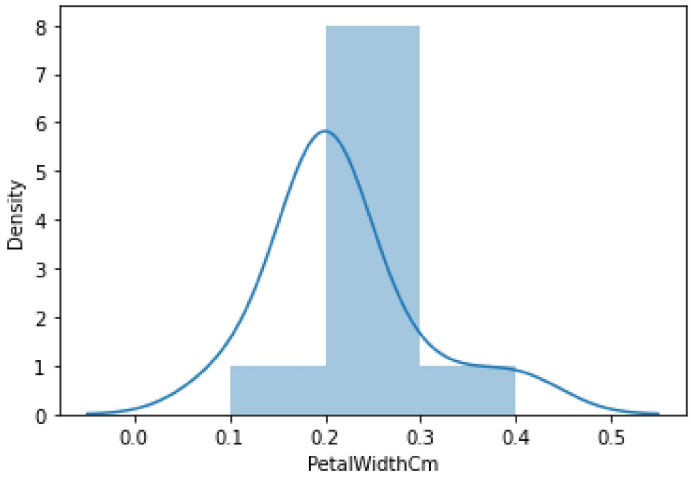
In [231]:
```
sns.pairplot(d)
```



In [232]:
```
sns.distplot(a['PetalWidthCm'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `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 flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

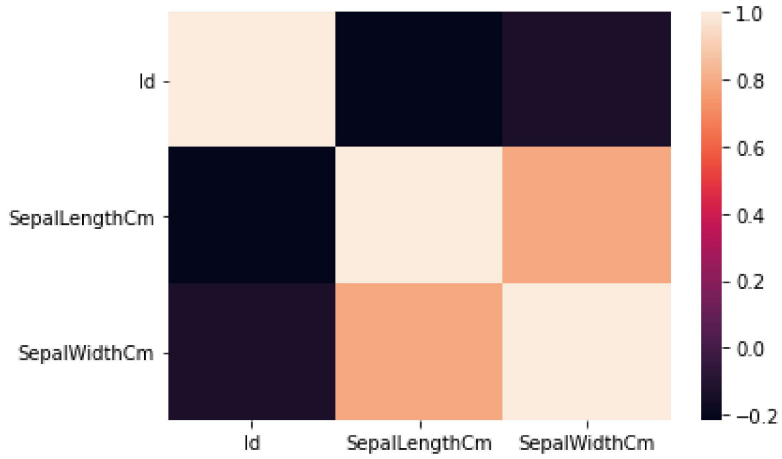
Out[232]:
```
<AxesSubplot:xlabel='PetalWidthCm', ylabel='Density'>
```



In [233]:
```
x1=a[['Id', 'SepalLengthCm', 'SepalWidthCm']]
```

In [234]:
```
sns.heatmap(x1.corr())
```

Out[234]: <AxesSubplot:>



In [235]:
```
x=a[['Id', 'SepalLengthCm', 'SepalWidthCm']]
y=a['PetalWidthCm']
```

In [236]:
```
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 [237]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[237]: LinearRegression()

In [238]:
```
print(lr.intercept_)
```

```
0.5071476736345251
```

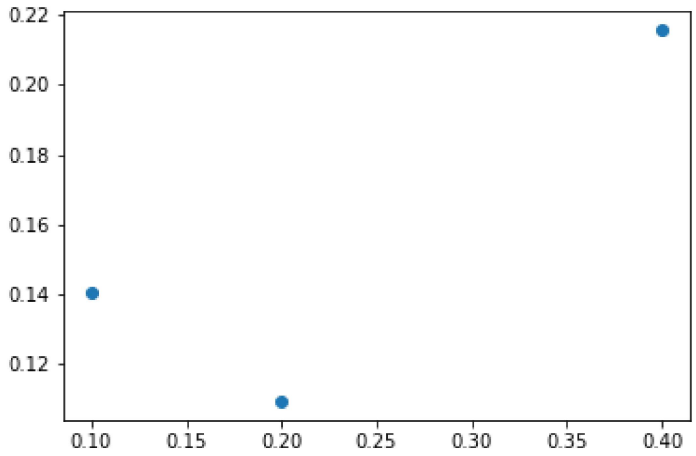
In [239]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[239]:

	Co-efficient
Id	0.000854
SepalLengthCm	-0.229490
SepalWidthCm	0.241702

In [240]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[240]:
```
<matplotlib.collections.PathCollection at 0x190bbc11e20>
```



In [241]:
```
print(lr.score(x_test,y_test))
```

```
0.061142626948208934
```

In [242]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [243]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[243]:
```
Ridge(alpha=10)
```

In [244]:
```
rr.score(x_test,y_test)
```

Out[244]:
```
-0.0699393296593398
```

In [245]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[245]:
```
Lasso(alpha=10)
```

In [246]:
```
la.score(x_test,y_test)
```

Out[246]:
```
-0.023323615160349975
```

In [247]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[247]:
```
ElasticNet()
```

In [248]:
```
print(en.coef_)
```

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

In [249]:
```
print(en.intercept_)
```

```
0.21428571428571425
```

In [250]:
```
print(en.predict(x_test))
```

```
[0.21428571 0.21428571 0.21428571]
```

In [251]:
```
en.score(x_test,y_test)
```

Out[251]:
```
-0.023323615160349975
```

In [252]:
```
from sklearn import metrics
```

In [253]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 0.10511225993356825
```

In [254]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 0.014604448025250086
```

In [255]:
```
print(" Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
 Root Mean Squared Error 0.12084886439371322
```