DATA COLLECTION

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

In [2]:
```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\14_Iris.csv")
sd
```

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]:
```
# to display top 10 rows
sd.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

DATA CLEANING AND PRE_PROCESSING

In [4]:
```
sd.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]:
```
# to display summary of statistics
sd.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

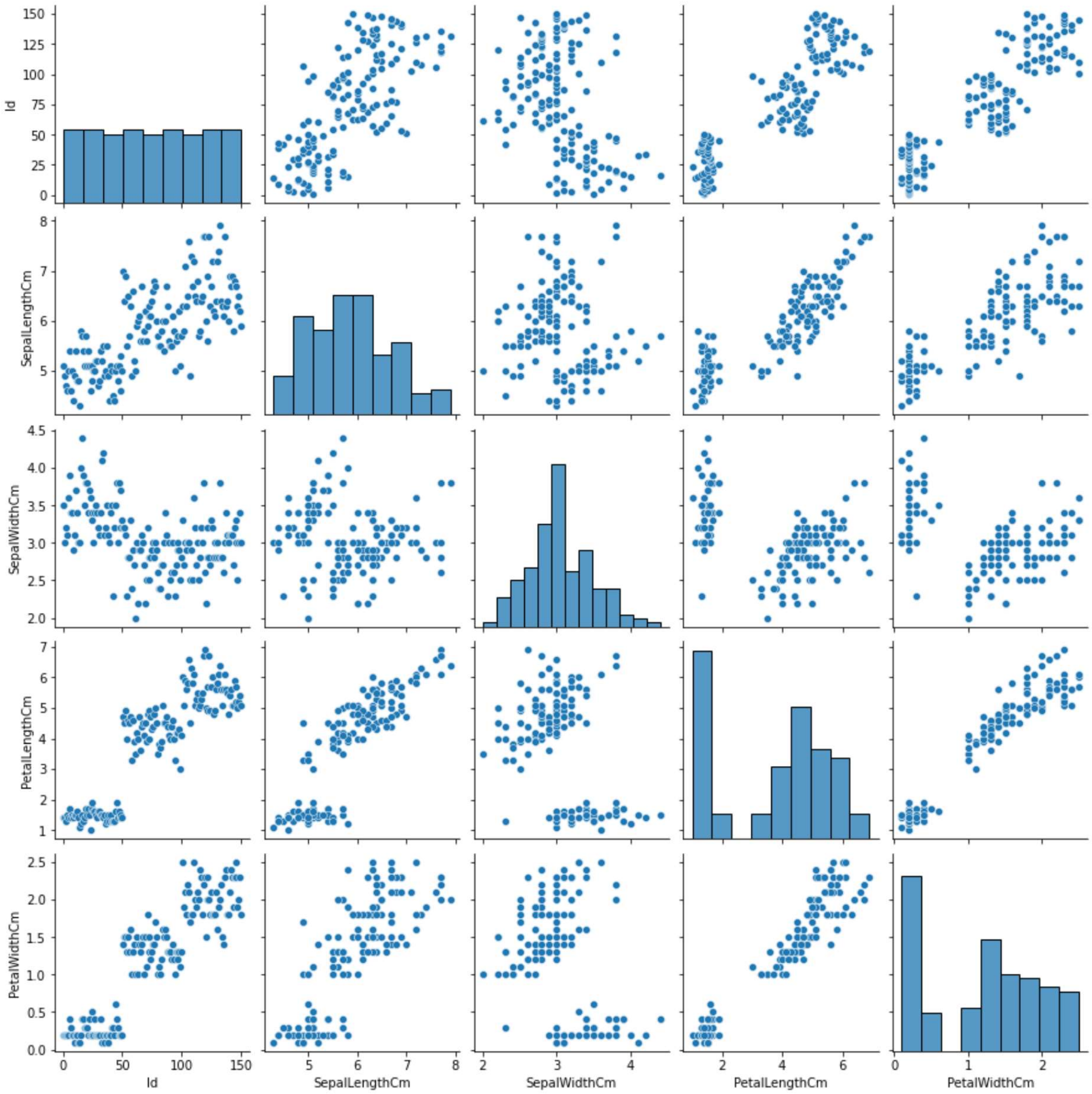
In [6]:
```
#to display colums heading
sd.columns
```

Out[6]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthC
m',
       'Species'],
      dtype='object')
```

EDA and visualization

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

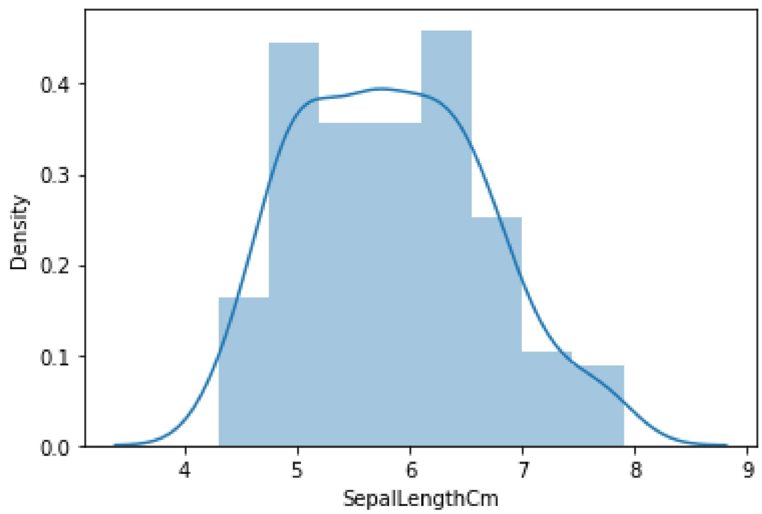
Out[7]: <seaborn.axisgrid.PairGrid at 0x1775c7f7220>



In [8]:
```
sns.distplot(sd['SepalLengthCm'])
```

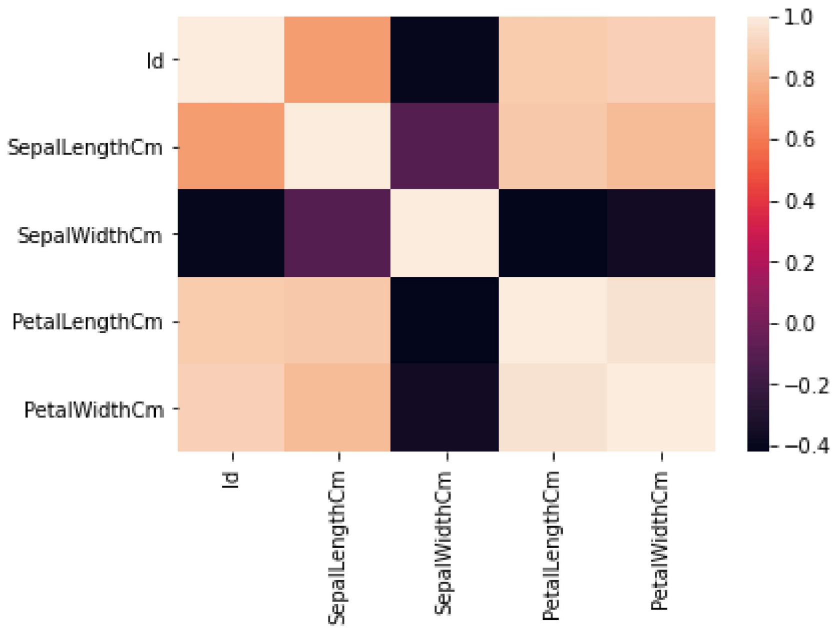
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



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

Out[9]:
```
<AxesSubplot:>
```



In [10]:
```
sd1=sd[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
```

TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [11]:
```
x= sd1[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]
y=sd1['PetalWidthCm']
```

In [12]:
```
# To split my dataset  into training data and test data
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]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:
```
LinearRegression()
```

In [15]:
```
print(lr.intercept_)
```

```
-0.2277368432715572
```

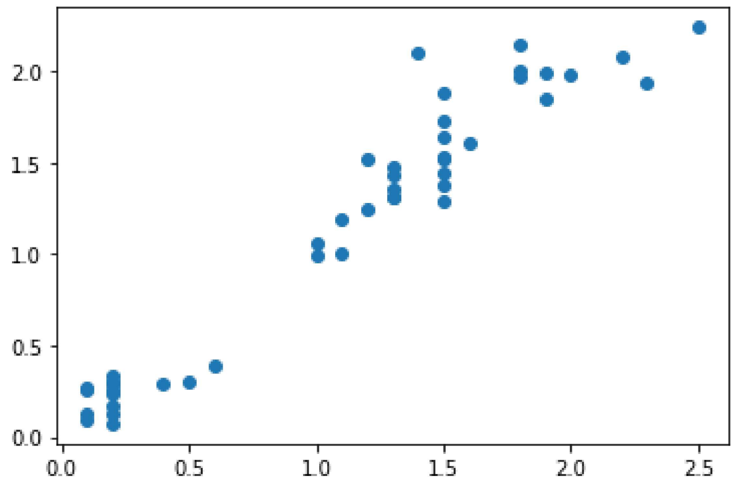
In [16]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Id	0.003760
SepalLengthCm	-0.177068
SepalWidthCm	0.185387
PetalLengthCm	0.432020

In [17]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]: <matplotlib.collections.PathCollection at 0x1775f8514f0>



In [18]:
```
print(lr.score(x_test,y_test))
```

0.9269510477479206

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]: 0.9510098209828819

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

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

Out[21]: Ridge(alpha=10)

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

Out[22]: 0.9151267065690689

In [23]:
```
dr.score(x_train,y_train)
```

Out[23]: 0.9436787566643782

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]: Lasso(alpha=10)

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

Out[25]: 0.702177176757073

In [26]:
```
la.score(x_train,y_train)
```
Out[26]:
```
0.7464150340478268
```

ElasticNet

In [27]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[27]:
```
ElasticNet()
```

In [28]:
```
print(en.coef_)
```
```
[ 0.01600339  0.          -0.           0.        ]
```

In [29]:
```
print(en.intercept_)
```
```
-0.0017865824086864546
```

In [30]:
```
prediction=en.predict(x_test)
```

In [31]:
```
print(en.score(x_test,y_test))
```
```
0.7418381680932282
```

Evaluation metric

In [32]:
```
from sklearn import metrics
```

In [33]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```
```
mean Absolute Error: 0.27703662364470205
```

In [34]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```
```
mean squared Error: 0.12396357475203339
```

In [35]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```
```
Root mean Absolytre Error: 0.3520846130577619
```

In [ ]:
```

```