

# 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
<b>count</b>	150.000000	150.000000	150.000000	150.000000	150.000000
<b>mean</b>	75.500000	5.843333	3.054000	3.758667	1.198667
<b>std</b>	43.445368	0.828066	0.433594	1.764420	0.763161
<b>min</b>	1.000000	4.300000	2.000000	1.000000	0.100000
<b>25%</b>	38.250000	5.100000	2.800000	1.600000	0.300000
<b>50%</b>	75.500000	5.800000	3.000000	4.350000	1.300000
<b>75%</b>	112.750000	6.400000	3.300000	5.100000	1.800000
<b>max</b>	150.000000	7.900000	4.400000	6.900000	2.500000

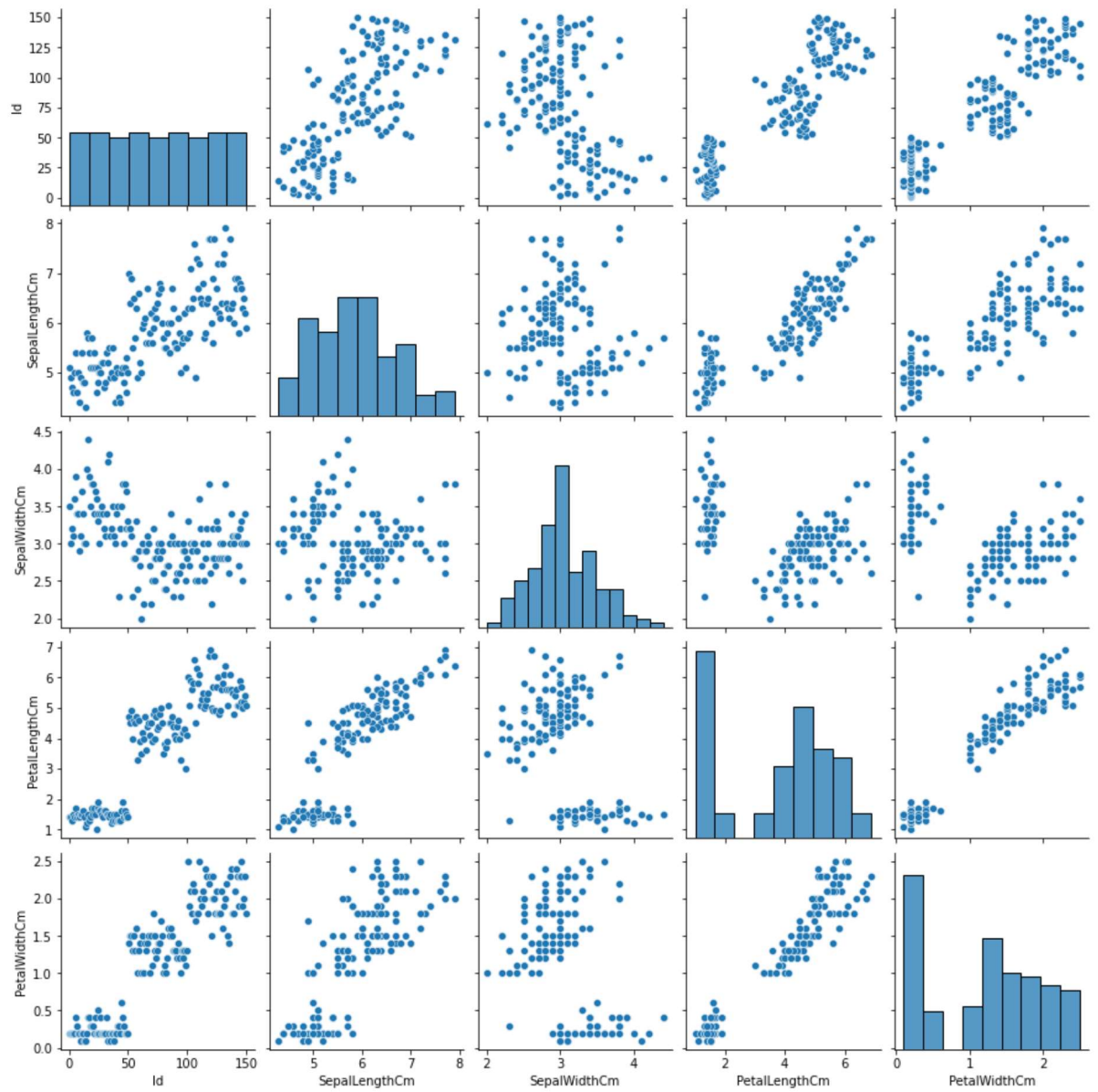
```
In [6]: #to display colums heading
sd.columns
```

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

## EDA and visualization

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

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x23574046190>
```

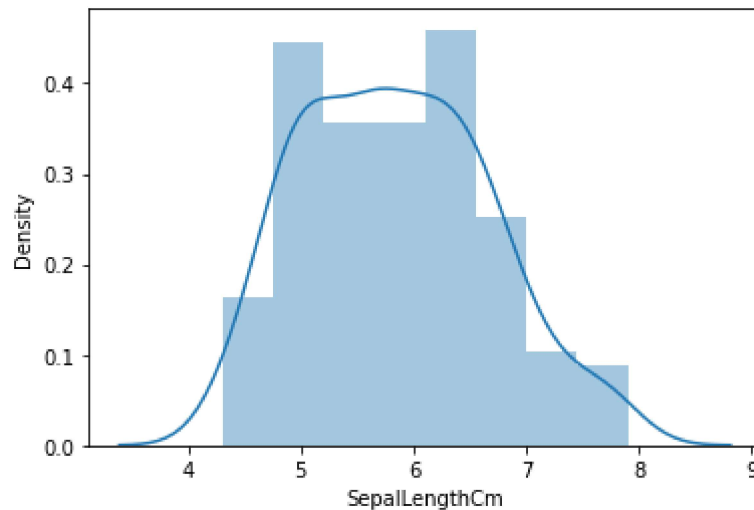


```
In [8]: sns.distplot(sd['SepalLengthCm'])
```

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 function for histograms).

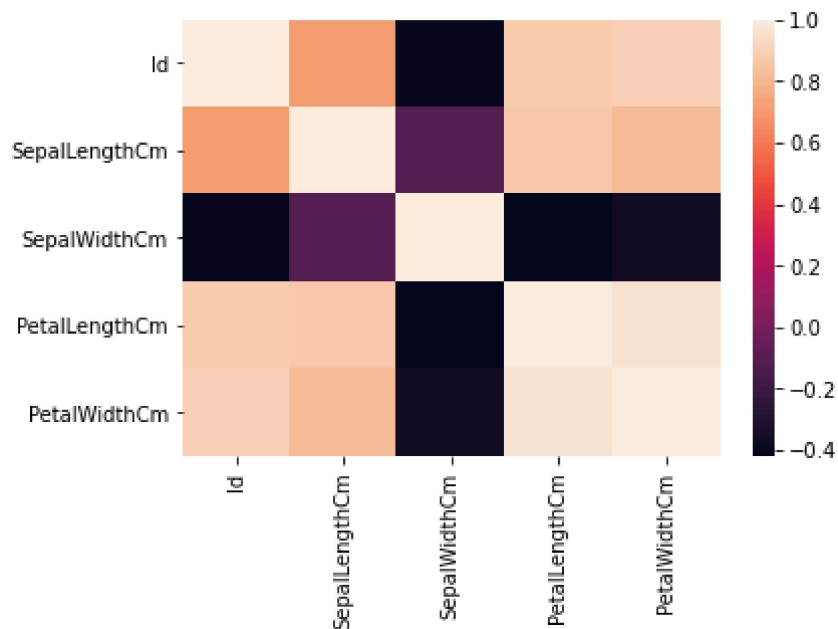
```
warnings.warn(msg, FutureWarning)
```

```
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 going to train Linear Regression model; we need to split out the data into two variables x and y where x is independent on x (output) and y is dependent on x(output) address column as it is not required our model

```
In [13]: x= sd1[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]  
y=sd1['PetalWidthCm']
```

```
In [14]: # 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 [15]: from sklearn.linear_model import LinearRegression  
  
lr=LinearRegression()  
lr.fit(x_train,y_train)
```

```
Out[15]: LinearRegression()
```

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

```
Out[16]: LinearRegression()
```

```
In [17]: print(lr.intercept_)  
  
-0.29052557317060246
```

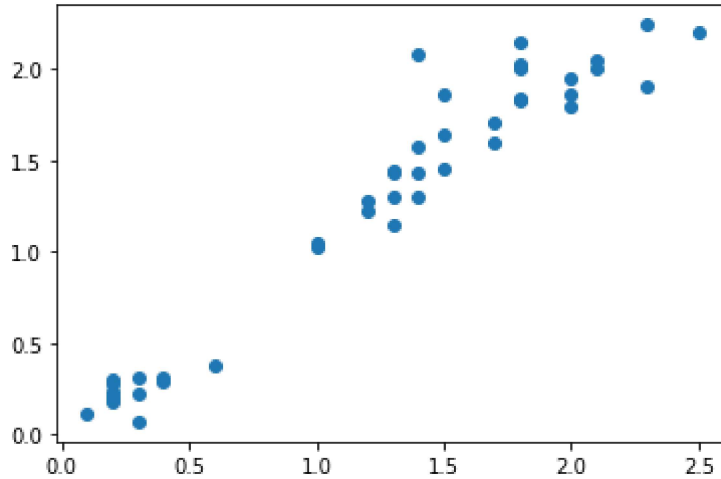
```
In [18]: coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

```
Out[18]:
```

	Co-efficient
Id	0.003244
SepalLengthCm	-0.195046
SepalWidthCm	0.222298
PetalLengthCm	0.454512

```
In [19]: prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[19]: <matplotlib.collections.PathCollection at 0x2357af31ee0>



```
In [20]: print(lr.score(x_test,y_test))
```

0.9400957998439909

```
In [21]: lr.score(x_train,y_train)
```

Out[21]: 0.947950348112873

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

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

Out[23]: Ridge(alpha=10)

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

Out[24]: 0.9422101632413354

```
In [25]: dr.score(x_train,y_train)
```

Out[25]: 0.9390045506639481

```
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]: 0.7845569005183551

In [28]: `la.score(x_train,y_train)`

Out[28]: 0.7119504804388858

In [ ]:

In [ ]: