Data collection

Importing libraries

In [1]:

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

Importing dataset

In [2]:

```
data=pd.read_csv(r"C:\Users\user\Downloads\iris.csv")
data
```

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

head

In [3]:

```
# to display first 8 dataset values
da=data.head(8)
da
```

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

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

info

In [4]:

```
# to identify missing values
data.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
```

describe

In [5]:

```
# to display summary of the dataset
data.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

columns

In [6]:
```
# to display headings of the dataset
data.columns
```

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

In [7]:
```
a=data.dropna(axis=1)
a
```

Out[7]:

	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

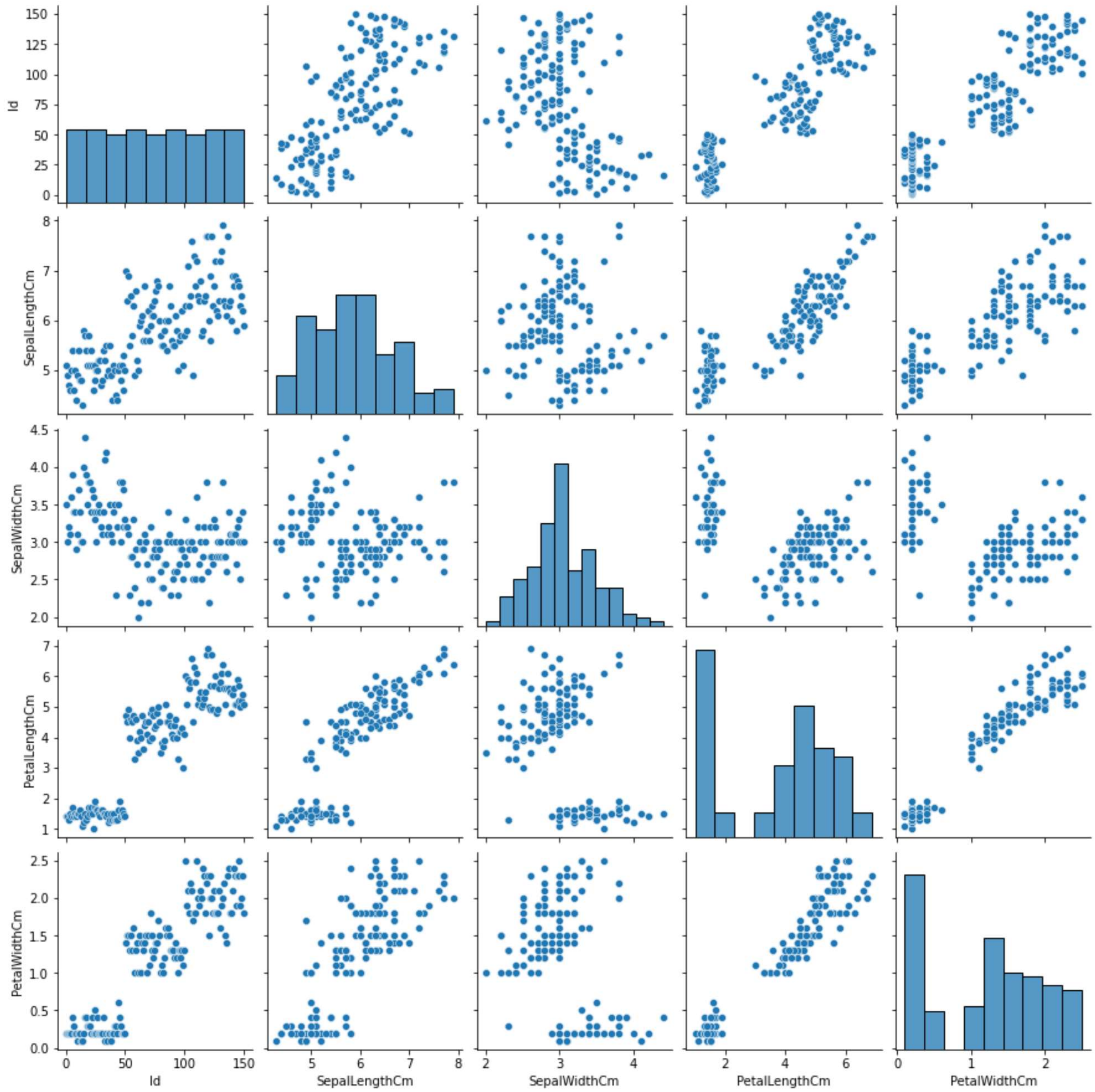
In [8]:
```
a.columns
```

Out[8]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

EDA and Visualization

In [9]:
```
sns.pairplot(a)
```

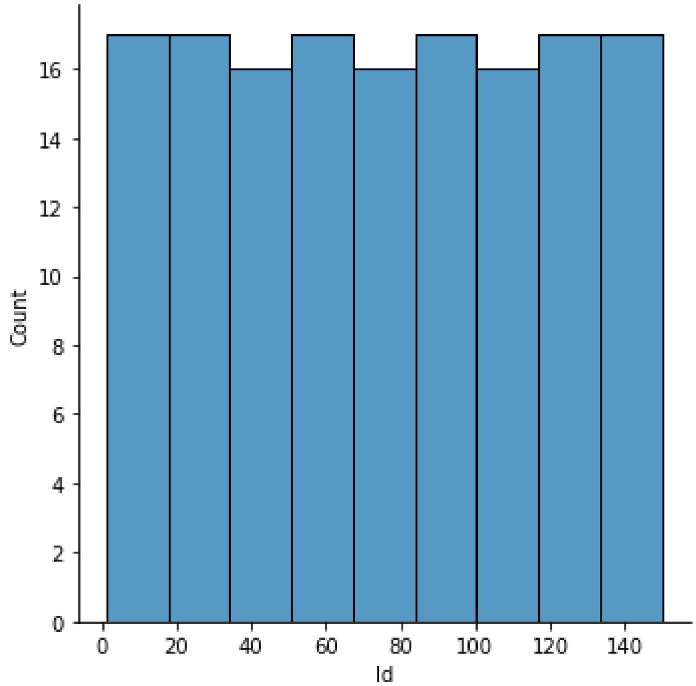
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1f5d904fd60>
```



distribution plot

In [10]:
```
sns.displot(a["Id"])
```

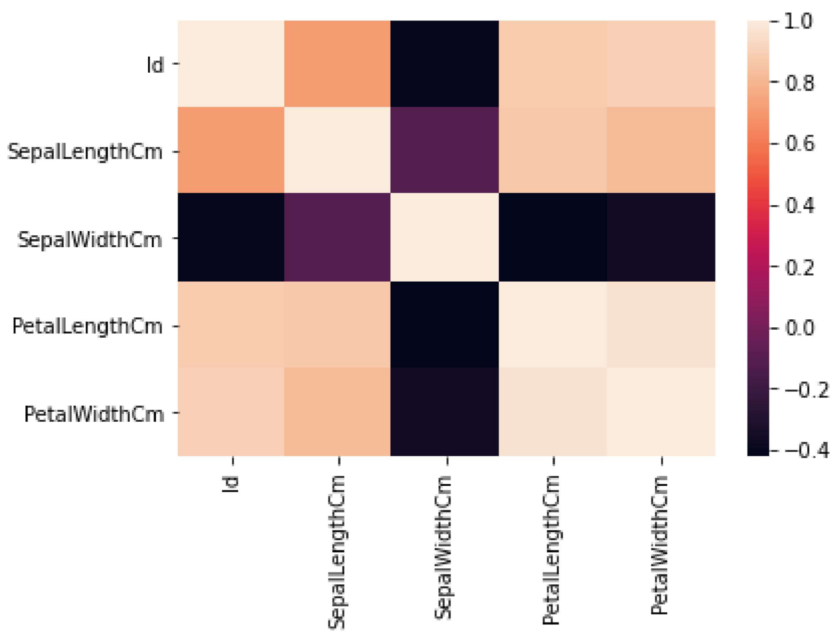
Out[10]: <seaborn.axisgrid.FacetGrid at 0x1f5db0e5190>



correlation

In [11]:
```
dat=data[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
        'Species']]
sns.heatmap(dat.corr())
```

Out[11]: <AxesSubplot:>



To train the model-Model Building

In [12]:
```
x=a[['Id']]
y=a['Id']
```

In [13]:
```
# to split my dataset into training 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 [14]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

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

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

```
-5.684341886080802e-14
```

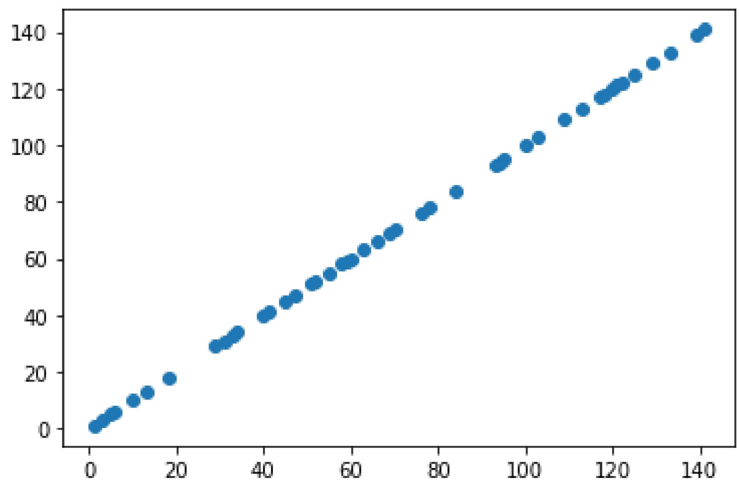
In [16]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Id	1.0

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

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1f5dc06a730>
```



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

```
1.0
```

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

Out[19]:
```
1.0
```

Ridge regression

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

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[21]: 0.9999999974838389

In [22]:
```
rr.score(x_train,y_train)
```

Out[22]: 0.9999999975587197

Lasso regression

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_train,y_train)
```

Out[23]: 0.9999730822246666

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

Out[24]: 0.999972256583328

In [25]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[25]: ElasticNet()

In [26]:
```
print(en.coef_)
```

```
[0.99948131]
```

In [27]:
```
print(en.intercept_)
```

```
0.04028977606824924
```

In [28]:
```
predict=en.predict(x_test)
```

In [29]:
```
print(en.score(x_test,y_test))
```

```
0.9999997227097166
```

In [30]:

```
from sklearn import metrics
```

In [31]:

```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute error: 0.018934833536820985
```

In [32]:

```
print("Mean Squared error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Squared error: 0.00047570058718622837
```

In [33]:

```
print("Root squared error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root squared error: 0.02181056136797557
```