In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df = pd.read_csv("Iris.csv")
# .dropna(axis="columns")
df
```

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.head()
```

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

Data cleaning and pre processing

In [4]:
```
df.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]:
```
df.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

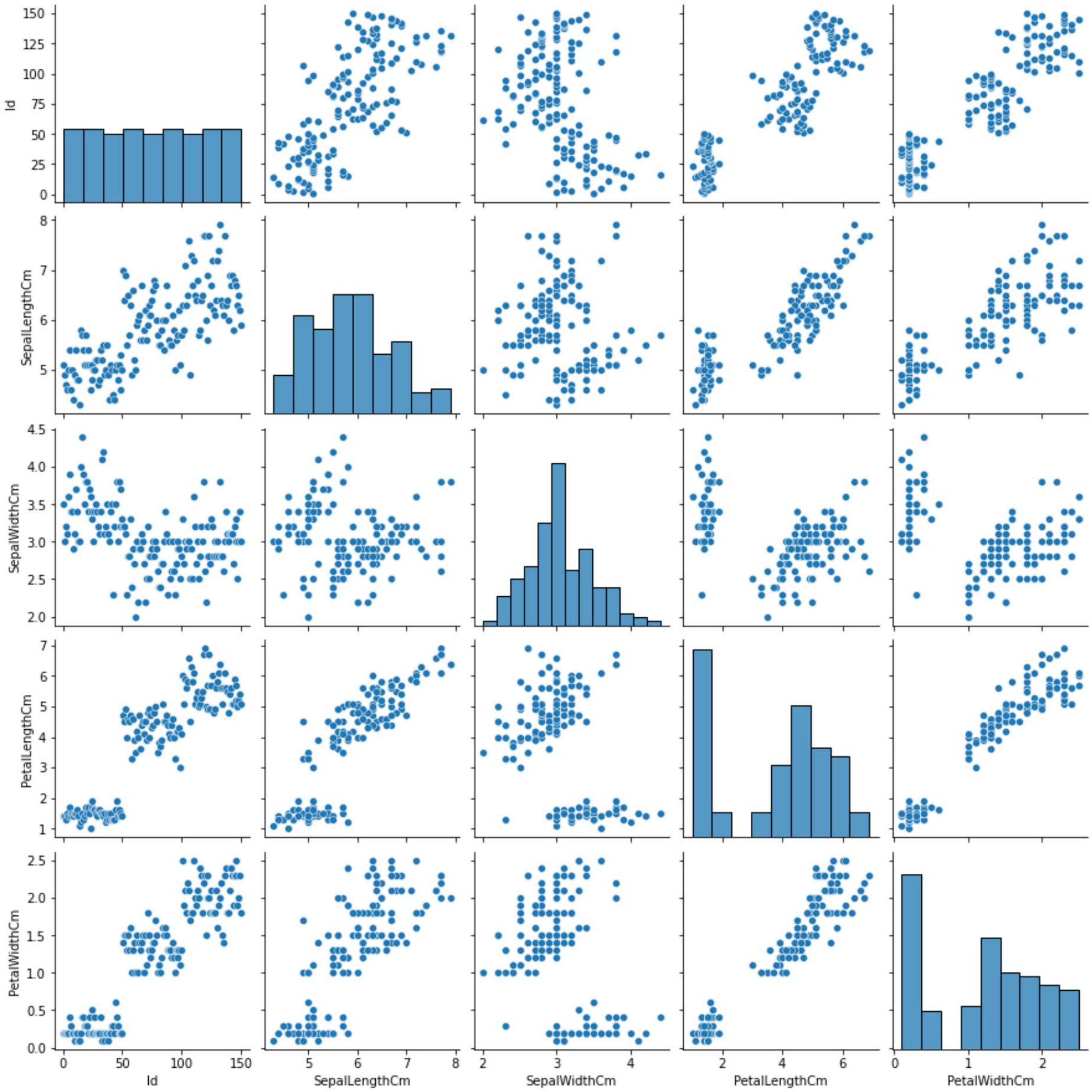
In [6]:
```
df.columns
```

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

EDA and VISUALIZATION

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

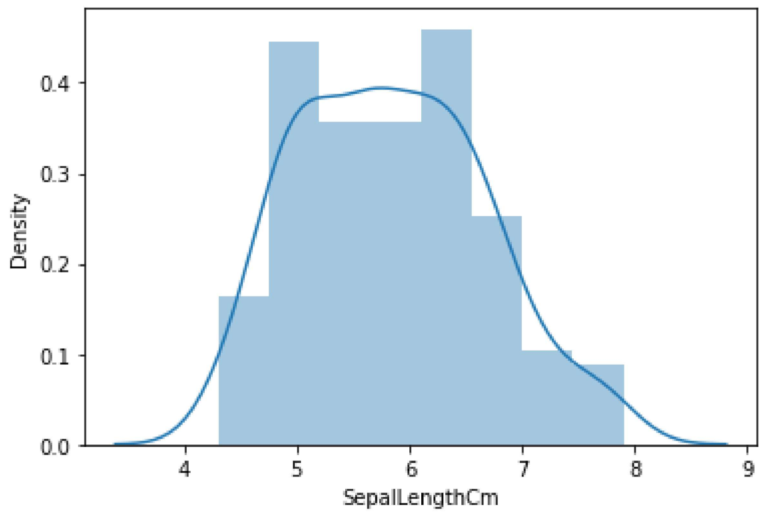
Out[7]: <seaborn.axisgrid.PairGrid at 0x1fc6d446280>



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

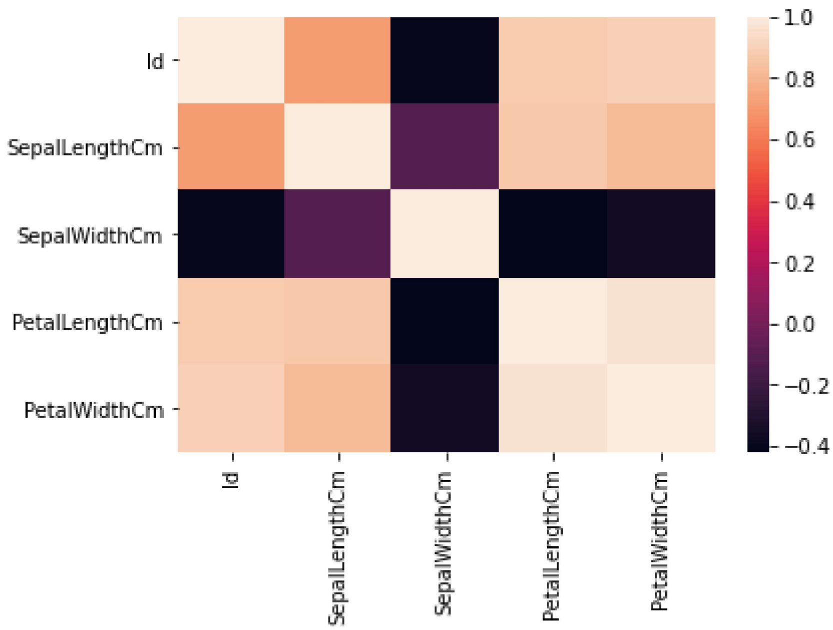
Out[8]:
```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='Density'>
```



In [9]:
```
df1 = df[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm
       'Species']]
```

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:
```
x = df1[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalWidthCm']]
y = df1[ 'PetalLengthCm']
```

split the data into training and test data

In [12]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

In [13]:
```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[13]: LinearRegression()

In [14]:
```
lr.intercept_
```

Out[14]: -0.2752468379329023

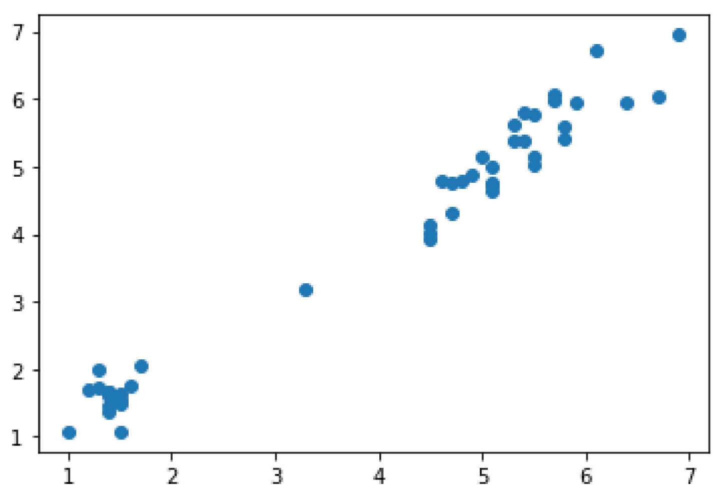
In [15]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
Id	0.001377
SepalLengthCm	0.733720
SepalWidthCm	-0.645673
PetalWidthCm	1.344928

In [16]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[16]: <matplotlib.collections.PathCollection at 0x1fc6ec6a9d0>



In [17]:
```
lr.score(x_test,y_test)
```

Out[17]: 0.9702082337641564

ACURACY

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[19]:
```
0.9469166829095399
```

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

Out[20]:
```
0.9526805633458773
```

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

Out[21]:
```
Lasso(alpha=10)
```

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

Out[22]:
```
0.7692575517416311
```

In [23]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

In [24]:
```
print(en.coef_)
```

```
[ 0.03639067  0.         -0.          0.        ]
```

In [25]:
```
print(en.intercept_)
```

```
1.0031375554410276
```

In [26]:
```
print(en.predict(x_test))
```

```
[6.02504962 3.11379625 5.73392428 6.09783095 6.42534695 2.20402957
 4.05995359 2.53154557 2.85906158 6.20700295 5.22445494 2.93184291
 3.00462425 3.33214025 4.89693894 5.29723627 1.54899756 1.9492949
 1.8401229  1.6217789  4.96972027 1.33065356 1.58538823 1.87651357
 2.0584669  5.33362694 5.40640827 5.87948695 5.80670561 6.06144028
 6.27978429 5.15167361 6.24339362 1.07591889 6.46173762 3.04101491
 3.44131225 1.47621623 1.29426289 5.26084561 2.82267091 5.07889227
 5.95226828 2.78628024 5.11528294]
```

In [27]:
```
print(en.score(x_test,y_test))
```

```
0.7784645511260997
```

In [28]:
```
# Evaluation Metrics
from sklearn import metrics
```

In [29]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.2719506202609821
```

In [30]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.11051406483015257
```

In [31]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Root Mean Squared Error: 0.3324365576018266
```