bnaubvi6a

July 31, 2023

```
[ ]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn import metrics0
```

```
[ ]: df=pd.read_csv("/content/14_Iris.csv")
     df
```

```
[ ]:       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
     0      1            5.1           3.5            1.4           0.2
     1      2            4.9           3.0            1.4           0.2
     2      3            4.7           3.2            1.3           0.2
     3      4            4.6           3.1            1.5           0.2
     4      5            5.0           3.6            1.4           0.2
     ..   …            …           …            …           …
     145  146            6.7           3.0            5.2           2.3
     146  147            6.3           2.5            5.0           1.9
     147  148            6.5           3.0            5.2           2.0
     148  149            6.2           3.4            5.4           2.3
     149  150            5.9           3.0            5.1           1.8

                 Species
     0       Iris-setosa
     1       Iris-setosa
     2       Iris-setosa
     3       Iris-setosa
     4       Iris-setosa
     ..              …
     145  Iris-virginica
     146  Iris-virginica
     147  Iris-virginica
     148  Iris-virginica
     149  Iris-virginica

     [150 rows x 6 columns]
```

```
[ ]: df.head()
```

```
[ ]:    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
```

1 DATA CLEANING AND DATA PREPROCESSING

```
[ ]: 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
```

```
[ ]: df.describe()
```

```
[ ]:                 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
```

```
[ ]: df.columns
```

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

```
[ ]: df1=df.dropna(axis=1)
     df1
```

```
[ ]:      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
     0      1            5.1           3.5            1.4           0.2
     1      2            4.9           3.0            1.4           0.2
     2      3            4.7           3.2            1.3           0.2
     3      4            4.6           3.1            1.5           0.2
     4      5            5.0           3.6            1.4           0.2
     ..   …            …            …             …            …
     145  146            6.7           3.0            5.2           2.3
     146  147            6.3           2.5            5.0           1.9
     147  148            6.5           3.0            5.2           2.0
     148  149            6.2           3.4            5.4           2.3
     149  150            5.9           3.0            5.1           1.8

                 Species
     0       Iris-setosa
     1       Iris-setosa
     2       Iris-setosa
     3       Iris-setosa
     4       Iris-setosa
     ..              …
     145  Iris-virginica
     146  Iris-virginica
     147  Iris-virginica
     148  Iris-virginica
     149  Iris-virginica

     [150 rows x 6 columns]
```

```
[ ]: df1.columns
```

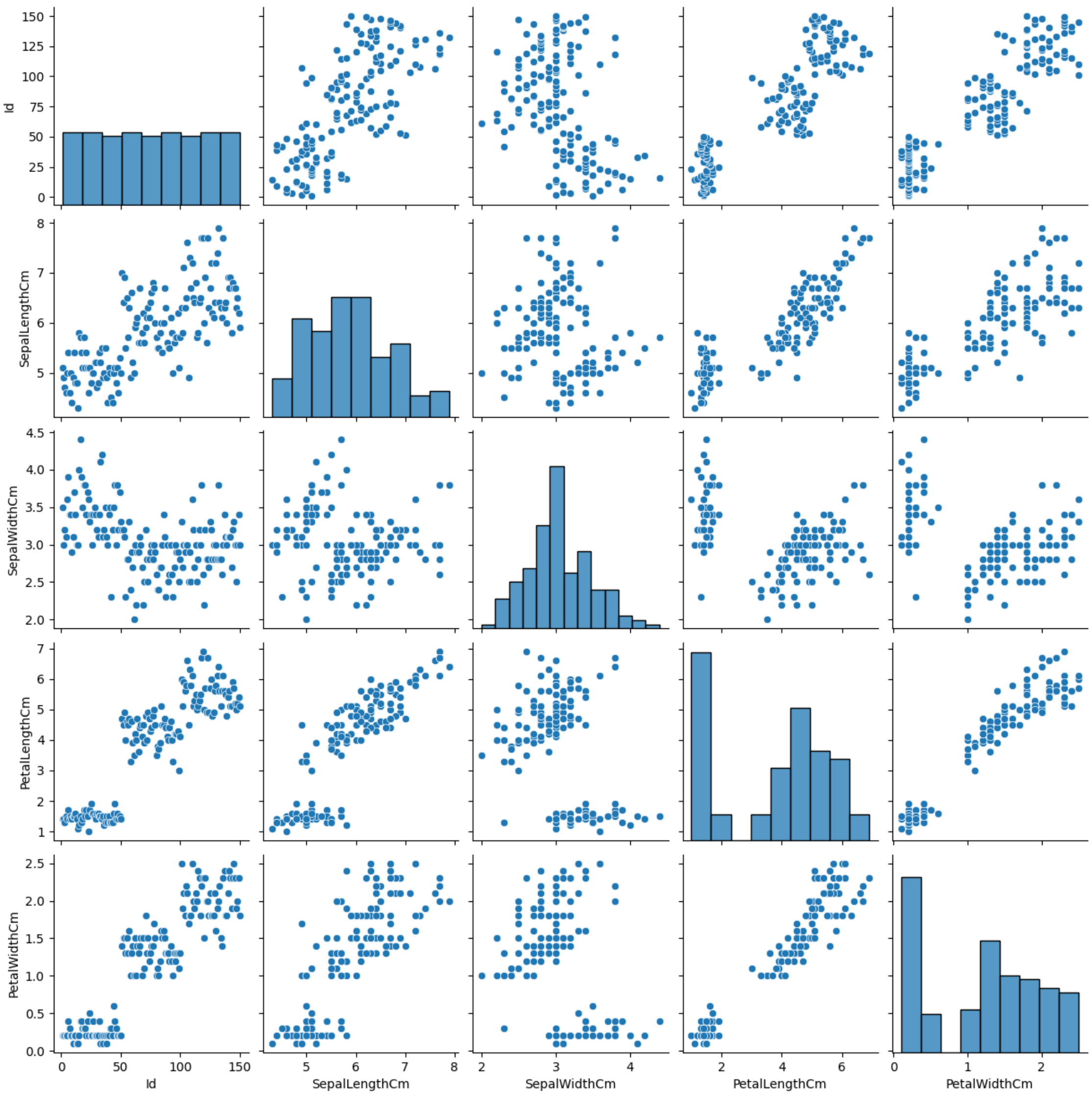
```
[ ]: Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
            'Species'],
           dtype='object')
```

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

2 EDA AND VISUALIZATION

```
[ ]: sns.pairplot(df1)
```

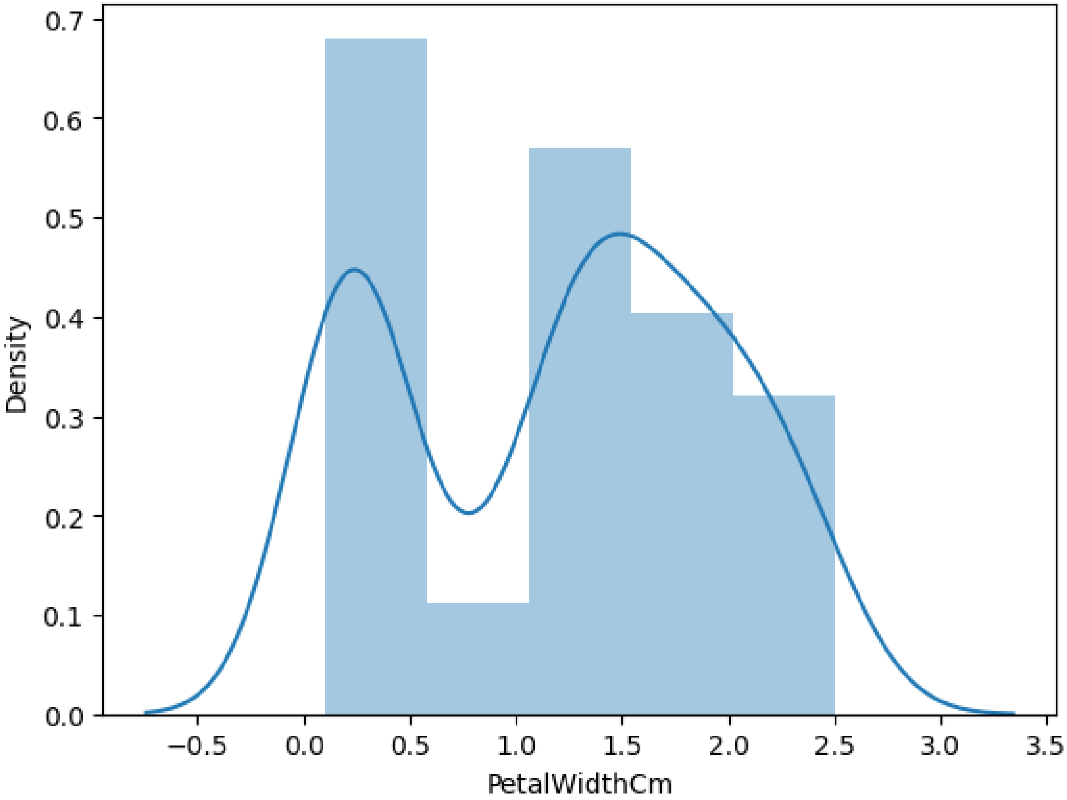
```
[ ]: <seaborn.axisgrid.PairGrid at 0x7900abdc9ff0>
```



```
[ ]: sns.distplot(df1['PetalWidthCm'])
```

```
<ipython-input-11-a51f8e882509>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df1['PetalWidthCm'])
```

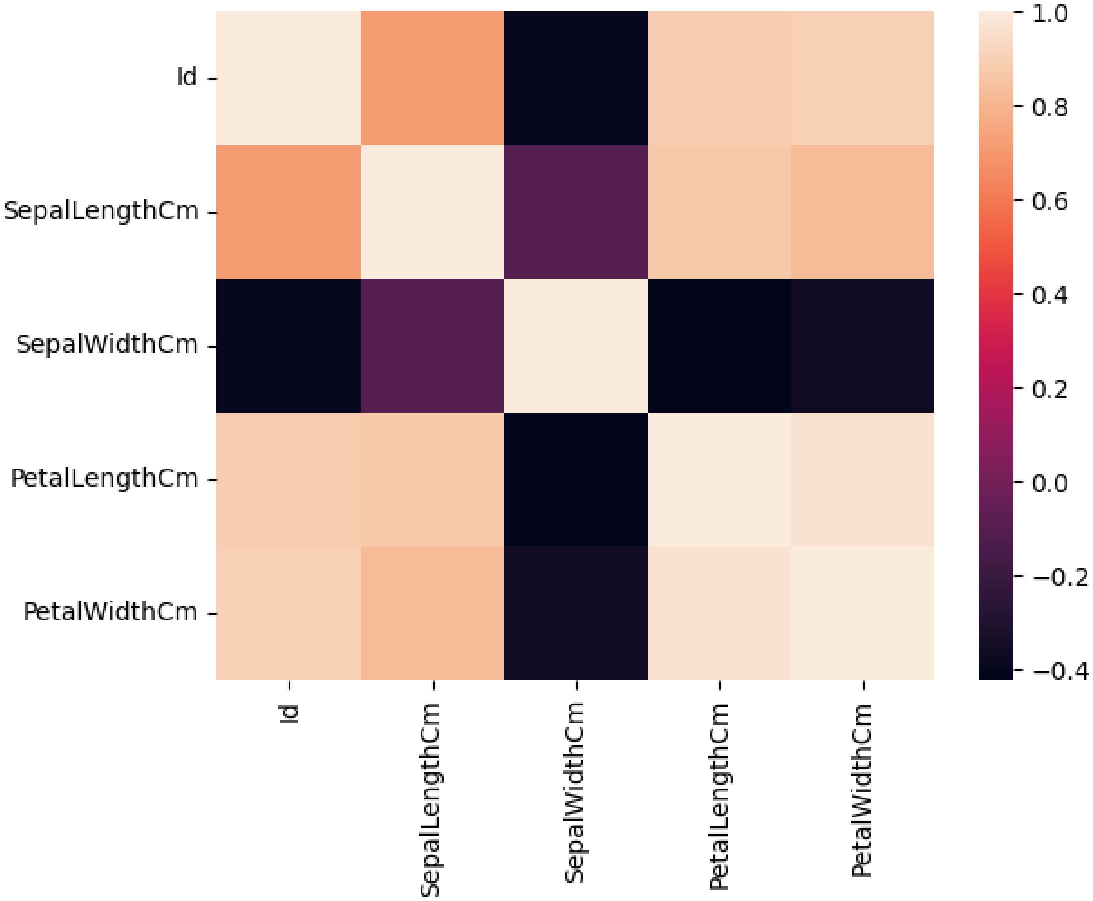
```
[ ]: <Axes: xlabel='PetalWidthCm', ylabel='Density'>
```



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

```
<ipython-input-12-3ed1a1a51dc0>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df1.corr())
```

```
[ ]: <Axes: >
```



3 TO TRAIN THE MODEL AND MODEL BULDING

```
[ ]: x=df[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]
     y=df['PetalWidthCm']
```

```
[ ]: 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)
```

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

```
[ ]: LinearRegression()
```

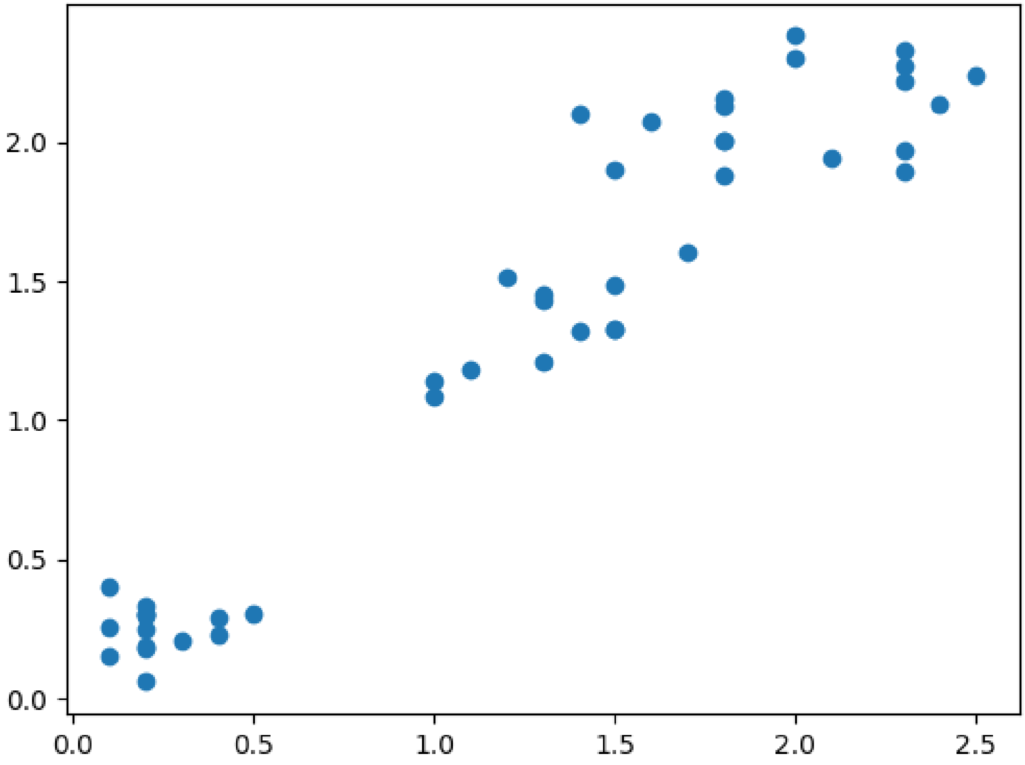
```
[ ]: lr.intercept_
```

```
[ ]: -0.35844946937954214
[ ]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
     coeff
```

```
[ ]:                Co-efficient
     Id                 0.004142
     SepalLengthCm     -0.158139
     SepalWidthCm       0.198941
     PetalLengthCm      0.419605
```

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

```
[ ]: <matplotlib.collections.PathCollection at 0x7900a3c3cc10>
```



4 ACCURACY

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

```
[ ]: 0.9144718273249249
```

```
[ ]: lr.score(x_train,y_train)
[ ]: 0.9598533851111487
```

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

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

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

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

```
[ ]: 0.9075790729420946
```

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

```
[ ]: 0.9521752703861144
```

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

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

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

```
[ ]: 0.7158708326427354
```

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

```
[ ]: 0.7134641451218205
```

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

```
[ ]: ElasticNet()
```

```
[ ]: en.coef_
```

```
[ ]: array([ 0.01578774,  0.        , -0.        ,  0.        ])
```

```
[ ]: en.intercept_
```

```
[ ]: 0.02253359302680069
```

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

```
[ ]: array([0.4014394 , 1.66445877, 0.63825553, 0.5119536 , 0.52774134,
            0.57510457, 1.80654845, 2.201242  , 1.0013736 , 1.50658135,
            2.15387877, 0.08568456, 1.6013078 , 1.25397748, 1.52236909,
            0.05410908, 1.61709554, 1.9012749 , 2.29596845, 1.96442587,
            1.26976522, 0.7014065 , 1.85391167, 0.18041101, 1.72760974,
            1.45921812, 0.54352908, 1.86969942, 2.24860523, 2.13809103,
            1.31712844, 0.49616586, 0.65404328, 2.37490716, 1.06452457,
            2.10651555, 0.36986392, 0.14883553, 1.12767554, 1.15925102,
            2.21702974, 2.07494006, 2.26439297, 0.89085941, 0.13304779])
```

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

```
[ ]: 0.7960049802908848
```

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

```
Mean Absolute Error:  0.28938159306182404
```

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

```
Mean Squared Error:  0.12789329245526135
```

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

```
Root Mean Squared Error:  0.35762171697935424
```