```
11/12/24, 2:43 PM
                                                                          ass2 I2.ipynb - Colab
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import warnings
    warnings.filterwarnings('ignore')
    df = pd.read_csv('Dry_Bean_Dataset.csv')
    df.head()
    \rightarrow
              Area Perimeter MajorAxisLength MinorAxisLength AspectRation Eccentricity ConvexArea EquivDiameter
          0 28395
                    610291.00
                                    208.178117
                                                     173.888747
                                                                      1.197191
                                                                                   0.549812
                                                                                                  28715
                                                                                                             190.141097 0.763923
                    638018.00
          1 28734
                                    200.524796
                                                     182.734419
                                                                      1.097356
                                                                                   0.411785
                                                                                                  29172
                                                                                                             191.272750 0.783968
          2 29380
                       624.11
                                    212.826130
                                                     175.931143
                                                                     1.209713
                                                                                   0.562727
                                                                                                  29690
                                                                                                             193.410904 0.778113
          3 30008
                    645884.00
                                    210.557999
                                                     182.516516
                                                                      1.153638
                                                                                   0.498616
                                                                                                  30724
                                                                                                             195.467062 0.782681
                    620134.00
                                                                                   0.333680
                                                                                                             195.896503 0.773098
          4 30140
                                    201.847882
                                                     190.279279
                                                                      1.060798
                                                                                                  30417
    df.info()
    <<class 'pandas.core.frame.DataFrame'>
         RangeIndex: 13611 entries, 0 to 13610
         Data columns (total 17 columns):
                               Non-Null Count Dtype
          # Column
                               13611 non-null
              Area
                                               int64
              Perimeter
                               13611 non-null
                                               float64
          1
              MajorAxisLength 13611 non-null float64
              MinorAxisLength 13611 non-null
                                               float64
          4
              AspectRation
                               13611 non-null float64
          5
              Eccentricity
                               13611 non-null float64
          6
              ConvexArea
                               13611 non-null int64
              EquivDiameter
                               13611 non-null float64
          8
              Extent
                               13611 non-null
                                               float64
              Solidity
                               13611 non-null float64
          10
                               13611 non-null
              roundness
                                               float64
          11
              Compactness
                               13611 non-null float64
              ShapeFactor1
                               13611 non-null
                                               float64
          12
              ShapeFactor2
                               13611 non-null float64
          13
              ShapeFactor3
                               13611 non-null float64
          14
                               13611 non-null float64
          15
             ShapeFactor4
          16 Class
                               13611 non-null object
         dtypes: float64(14), int64(2), object(1)
         memory usage: 1.8+ MB
    df.isnull().sum()
    \rightarrow
                            0
        Area
         Perimeter
                            0
         MajorAxisLength
                            0
         MinorAxisLength
                            0
         {\tt AspectRation}
         Eccentricity
                            0
         ConvexArea
                            0
         EquivDiameter
                            0
         Extent
         Solidity
         roundness
         Compactness
                            0
         ShapeFactor1
```

## df.duplicated().sum() <del>→</del> 68

ShapeFactor2

ShapeFactor3

ShapeFactor4

dtype: int64

Class

0

0

0

0

0

df.drop\_duplicates()

Extent Solidity

0.988856

0.984986

0.989559

0.976696

0.990893

0.

0.

0.

0.

 $\overline{\Rightarrow}$ 

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity
0	28395	610291.00	208.178117	173.888747	1.197191	0.549812	28715	190.141097	0.763923	0.988856
1	28734	638018.00	200.524796	182.734419	1.097356	0.411785	29172	191.272750	0.783968	0.984986
2	29380	624.11	212.826130	175.931143	1.209713	0.562727	29690	193.410904	0.778113	0.989559
3	30008	645884.00	210.557999	182.516516	1.153638	0.498616	30724	195.467062	0.782681	0.976696
4	30140	620134.00	201.847882	190.279279	1.060798	0.333680	30417	195.896503	0.773098	0.990893
13606	42097	759696.00	288.721612	185.944705	1.552728	0.765002	42508	231.515799	0.714574	0.990331
13607	42101	757499.00	281.576392	190.713136	1.476439	0.735702	42494	231.526798	0.799943	0.990752
13608	42139	759321.00	281.539928	191.187979	1.472582	0.734065	42569	231.631261	0.729932	0.989899
13609	42147	763779.00	283.382636	190.275731	1.489326	0.741055	42667	231.653248	0.705389	0.987813
13610	42159	772237.00	295.142741	182.204716	1.619841	0.786693	42600	231.686223	0.788962	0.989648
13543 rd	ows × 17	columns								
4										

df.duplicated().sum()

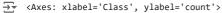
<del>→</del> 68

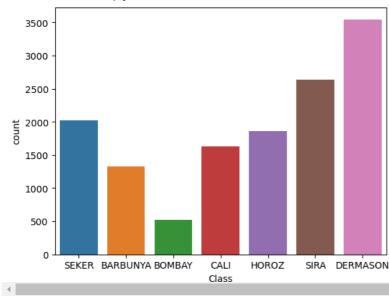
df.drop\_duplicates(inplace=True)

df.duplicated().sum()

**→** 0

sns.countplot(x='Class',data=df)





df.Class.value\_counts()

DERMASON 3546
SIRA 2636
SEKER 2027
HOROZ 1860
CALI 1630
BARBUNYA 1322
BOMBAY 522
Name: Class, dtype: int64

from sklearn.preprocessing import LabelEncoder

```
labelencoder = LabelEncoder()
df["Class"] = labelencoder.fit_transform(df['Class'])
```

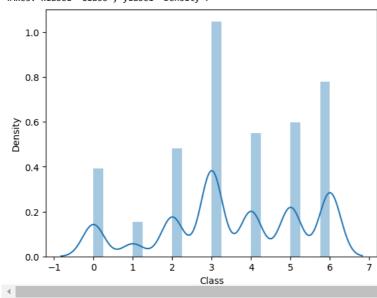
df['Class'].value\_counts()

```
3 3546
6 2636
5 2027
4 1860
2 1630
0 1322
1 522
```

Name: Class, dtype: int64

sns.distplot(df['Class'])

<Axes: xlabel='Class', ylabel='Density'>

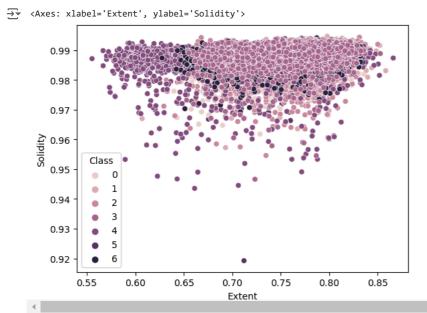


plt.figure(figsize=(12,5),dpi=200)
sns.heatmap(df.corr(),annot=True,linewidth = .5)

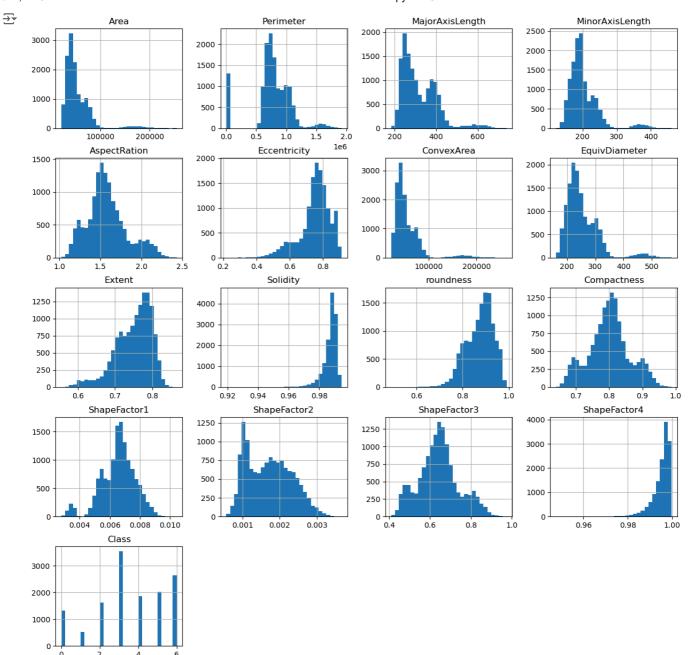
→ <Axes: >

, mes.																		
Area -	1	0.58	0.93	0.95	0.24	0.27	1	0.98	0.055	-0.2	-0.36	-0.27	-0.85	-0.64	-0.27	-0.36	-0.48	- 1.00
Perimeter -	0.58	1	0.58	0.55	0.22	0.22	0.58	0.59	-0.017	-0.18	-0.32	-0.23	-0.52	-0.45	-0.24	-0.26	-0.3	
MajorAxisLength -	0.93	0.58	1	0.83	0.55	0.54	0.93	0.96	-0.077	-0.28	-0.6	-0.57	-0.78	-0.86	-0.57	-0.48	-0.46	- 0.75
MinorAxisLength -	0.95	0.55	0.83	1	0.005 و	0.022	0.95	0.95	0.15	-0.16	-0.21	-0.019	-0.95	-0.48	-0.023	-0.27	-0.46	
AspectRation -	0.24	0.22	0.55	0.0054	1	0.92	0.25	0.31	-0.37	-0.27	-0.76	-0.99	0.021	-0.84	-0.98	-0.45	-0.12	- 0.50
Eccentricity -	0.27	0.22	0.54	0.022	0.92	1	0.27	0.32	-0.32	-0.3	-0.72	-0.97	0.017	-0.86	-0.98	-0.45	-0.2	
ConvexArea -	1	0.58	0.93	0.95	0.25	0.27	1	0.99	0.053	-0.21	-0.36	-0.27	-0.85	-0.64	-0.28	-0.36	-0.48	- 0.25
EquivDiameter -	0.98	0.59	0.96	0.95	0.31	0.32	0.99	1	0.029	-0.23	-0.44	-0.33	-0.89	-0.71	-0.33	-0.39	-0.48	
Extent -	0.055	-0.017	-0.077	0.15	-0.37	-0.32	0.053	0.029	1	0.19	0.34	0.35	-0.14	0.24	0.35	0.15	-0.031	- 0.00
Solidity -	-0.2	-0.18	-0.28	-0.16	-0.27	-0.3	-0.21	-0.23	0.19	1	0.61	0.3	0.15	0.34	0.31	0.7	0.32	
roundness -	-0.36	-0.32	-0.6	-0.21	-0.76	-0.72	-0.36	-0.44	0.34	0.61	1	0.77	0.23	0.78	0.76	0.47	0.39	0.25
Compactness -	-0.27	-0.23	-0.57	-0.019	-0.99	-0.97	-0.27	-0.33	0.35	0.3	0.77	1	-0.006	0.87	1	0.49	0.16	0.23
ShapeFactor1 -								-0.89	-0.14	0.15	0.23	-0.006	1	0.47	-0.005	0.25	0.39	0.50
ShapeFactor2 -											0.78	0.87	0.47	1	0.87	0.53	0.34	0.50
ShapeFactor3 -										0.31	0.76	1	-0.005	0.87	1	0.49	0.17	
ShapeFactor4 -						-0.45	-0.36			0.7	0.47	0.49	0.25	0.53	0.49	1	0.17	0.75
Class -			-0.46						-0.031	0.32	0.39	0.16	0.39	0.34	0.17	0.17	1	
	-		1	- 1			1	1	ı	- 1	- 1	- 1		- 1	-	1	- 1	
	Area	ter	gth	gth	ion	city	rea	ter	Extent	ξ	ess	ess	or1	or2	or3	or4	Class	
	∢	Ĭ.	-en	-en	Rat	ĬŢ	Ϋ́	me	Ext	Solidity	пр	ctu	act	act	act	act	$\Box$	
		Perimeter	xisl	xisl	ect	Eccentricity	ConvexArea	Dia		V)	roundness	ba	эеЕ	эеЕ	эеЕ	эеЕ		
			ŗ.	ΣŶ	AspectRation	E	ပိ	EquivDiameter			2	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4		
			/ajorAxisLength	//InorAxisLength	1			Е				O	S	S	S	S		

sns.scatterplot(x='Extent',y = 'Solidity',hue='Class',data = df)



df.hist(bins=30, figsize=(15,15))
plt.show()



from sklearn.model\_selection import train\_test\_split

```
X = df.drop(columns = 'Class')
Y = df[['Class']]
```

X.head()

3		Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	rou
	0	28395	610291.00	208.178117	173.888747	1.197191	0.549812	28715	190.141097	0.763923	0.988856	0.
	1	28734	638018.00	200.524796	182.734419	1.097356	0.411785	29172	191.272750	0.783968	0.984986	0.
	2	29380	624.11	212.826130	175.931143	1.209713	0.562727	29690	193.410904	0.778113	0.989559	0.
	3	30008	645884.00	210.557999	182.516516	1.153638	0.498616	30724	195.467062	0.782681	0.976696	0.
	4	30140	620134.00	201.847882	190.279279	1.060798	0.333680	30417	195.896503	0.773098	0.990893	0.
	4											<b>&gt;</b>

Y.head()

```
Class
0 5
1 5
2 5
3 5
4 5
```

```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X_scaled,Y,test_size = 0.2,random_state=5)

print('Training data shape : ', x_train.shape)
print('Training labels shape : ', y_train.shape)
print('Testing data shape : ', x_test.shape)
print('Testing labels shape : ', y_test.shape)

Training data shape : (10834, 16)
    Training labels shape : (2709, 16)
    Testing labels shape : (2709, 1)
```

## Training the model

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from \ sklearn.metrics \ import \ accuracy\_score, \ precision\_score, \ recall\_score, \ f1\_score, \ confusion\_matrix, \ classification\_report
models = {"Logistic regression":LogisticRegression(),
         "Decision Tree classifier": DecisionTreeClassifier(),
         "Random Forest Classifier": RandomForestClassifier()}
for i in range(len(list(models))):
    model = list(models.values())[i]
    model.fit(x_train,y_train)
    y_pred_train = model.predict(x_train)
   y_pred_test = model.predict(x_test)
    #training set performance
    model_train_accuracy = accuracy_score(y_train,y_pred_train)
    model_train_f1 = f1_score(y_train,y_pred_train,average='weighted')
    model_train_precision = precision_score(y_train,y_pred_train,average='weighted'
    model_train_recall = recall_score(y_train,y_pred_train,average='weighted')
    train_report = classification_report(y_train,y_pred_train)
    #testing set performance
    model_test_accuracy = accuracy_score(y_test,y_pred_test)
    model_test_f1 = f1_score(y_test,y_pred_test,average='weighted')
    model_test_precision = precision_score(y_test,y_pred_test,average='weighted')
    model_test_recall = recall_score(y_test,y_pred_test,average='weighted')
    test_report = classification_report(y_test,y_pred_test)
```

```
print('Model performance for training set')
   print(list(models.keys())[i])
   print('Accuracy: {:.4f}'.format(model_train_accuracy))
   print('F1:{:.4f}'.format(model_train_f1))
   print('Precision:{:.4f}'.format(model_train_precision))
   print('Recall:{:.4f}'.format(model_train_recall))
   print('
   print('Model performance for test set')
   print('Accuracy: {:.4f}'.format(model_test_accuracy))
   print('F1:{:.4f}'.format(model_test_f1))
   print('Precision:{:.4f}'.format(model_test_precision))
   print('Recall:{:.4f}'.format(model_test_recall))
   print(' ')

→ Model performance for training set
    Logistic regression
    Accuracy: 0.9257
    F1:0.9258
    Precision:0.9260
    Recall:0.9257
    Model performance for test set
    Accuracy: 0.9177
    F1:0.9181
    Precision:0.9190
    Recall:0.9177
    Model performance for training set
    Decision Tree classifier
    Accuracy: 1.0000
    F1:1.0000
    Precision:1.0000
    Recall:1.0000
    Model performance for test set
    Accuracy: 0.8966
    F1:0.8972
    Precision: 0.8983
    Recall:0.8966
    Model performance for training set
    Random Forest Classifier
    Accuracy: 1.0000
    F1:1.0000
    Precision:1.0000
    Recall:1.0000
    Model performance for test set
    Accuracy: 0.9195
    F1:0.9196
    Precision:0.9198
    Recall:0.9195
```

we are getting the best performance on Random Forest Classifier with an accuracy of 91% on training set

Selecting Random Forest Classifier will be the best choice for this dataset

```
scores = []
from sklearn.neighbors import KNeighborsClassifier

for i in range (1,16):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(x_train,y_train)
    y_pred_test = knn.predict(x_test)
    scores.annend(accuracy_score(y_test_y_pred_test))
```