```
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()
```

	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.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13611 entries, 0 to 13610
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Area             13611 non-null  int64
 1   Perimeter        13611 non-null  float64
 2   MajorAxisLength  13611 non-null  float64
 3   MinorAxisLength  13611 non-null  float64
 4   AspectRation     13611 non-null  float64
 5   Eccentricity     13611 non-null  float64
 6   ConvexArea       13611 non-null  int64
 7   EquivDiameter    13611 non-null  float64
 8   Extent           13611 non-null  float64
 9   Solidity         13611 non-null  float64
 10  roundness        13611 non-null  float64
 11  Compactness      13611 non-null  float64
 12  ShapeFactor1     13611 non-null  float64
 13  ShapeFactor2     13611 non-null  float64
 14  ShapeFactor3     13611 non-null  float64
 15  ShapeFactor4     13611 non-null  float64
 16  Class            13611 non-null  object
dtypes: float64(14), int64(2), object(1)
memory usage: 1.8+ MB
```

```
df.isnull().sum()
```

```
Area               0
Perimeter          0
MajorAxisLength    0
MinorAxisLength    0
AspectRation       0
Eccentricity       0
ConvexArea         0
EquivDiameter      0
Extent             0
Solidity           0
roundness          0
Compactness        0
ShapeFactor1       0
ShapeFactor2       0
ShapeFactor3       0
ShapeFactor4       0
Class              0
dtype: int64
```

```
df.duplicated().sum()
```

```
68
```

```
df.drop_duplicates()
```

	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 rows × 17 columns

```
df.duplicated().sum()
```

```
68
```

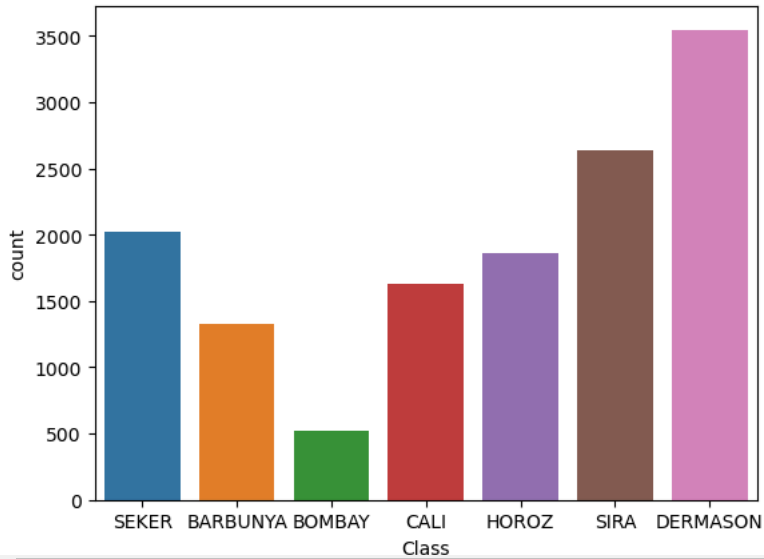
```
df.drop_duplicates(inplace=True)
```

```
df.duplicated().sum()
```

```
0
```

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

```
<Axes: xlabel='Class', ylabel='count'>
```




```
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
```

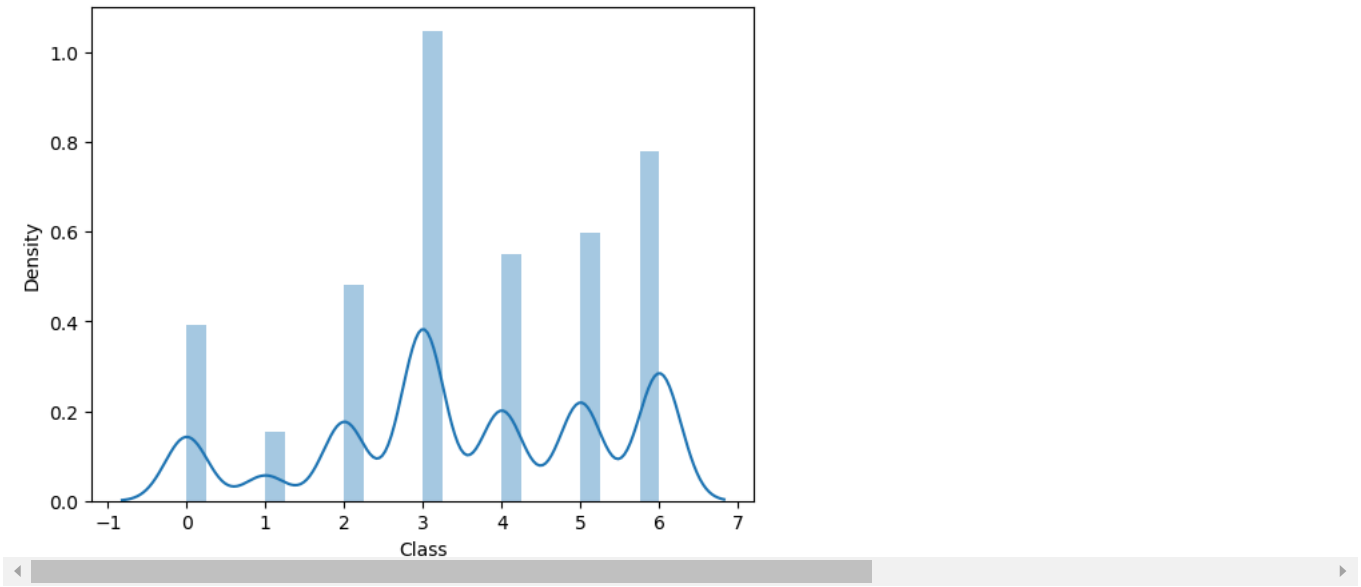
```
labelencoder = LabelEncoder()
df["Class"] = labelencoder.fit_transform(df['Class'])
```

```
df['Class'].value_counts()
```

```
3    3546
6    2636
5    2027
4    1860
2    1630
0    1322
1     522
Name: Class, dtype: int64
```

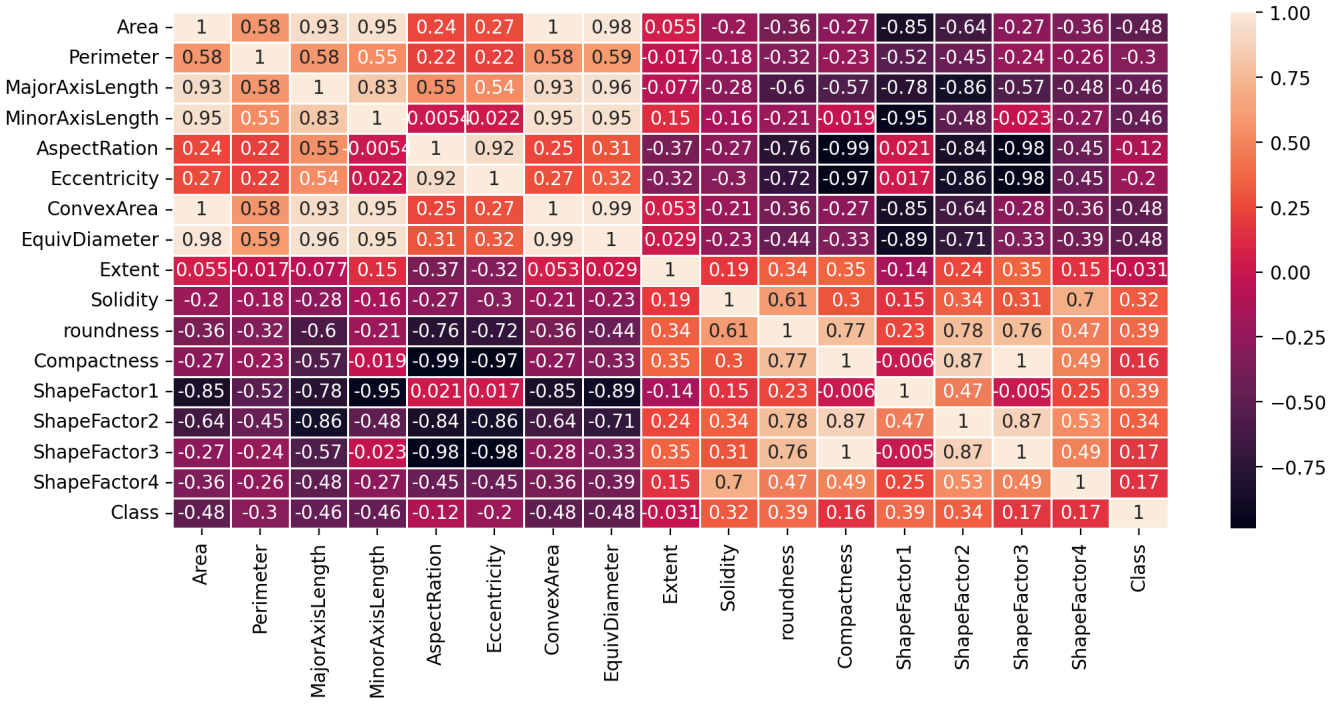
```
sns.distplot(df['Class'])
```

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



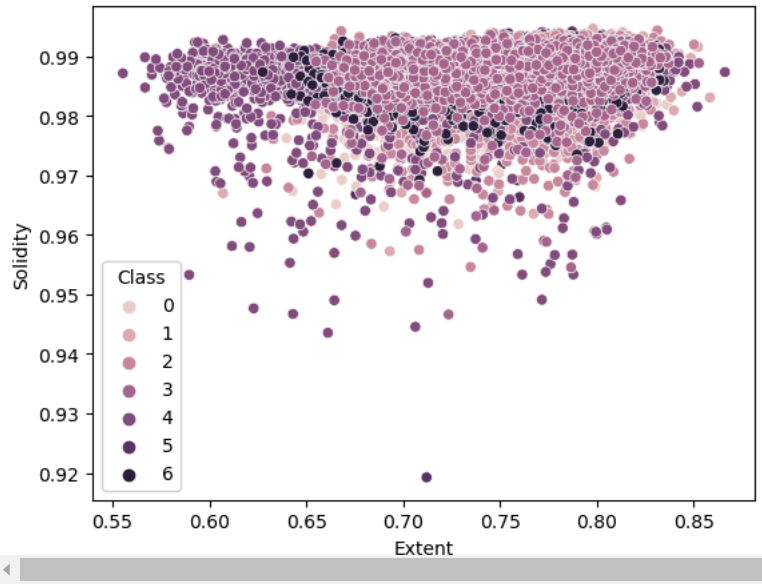

```
plt.figure(figsize=(12,5),dpi=200)
sns.heatmap(df.corr(),annot=True,linewidth = .5)
```

```
<Axes: >
```

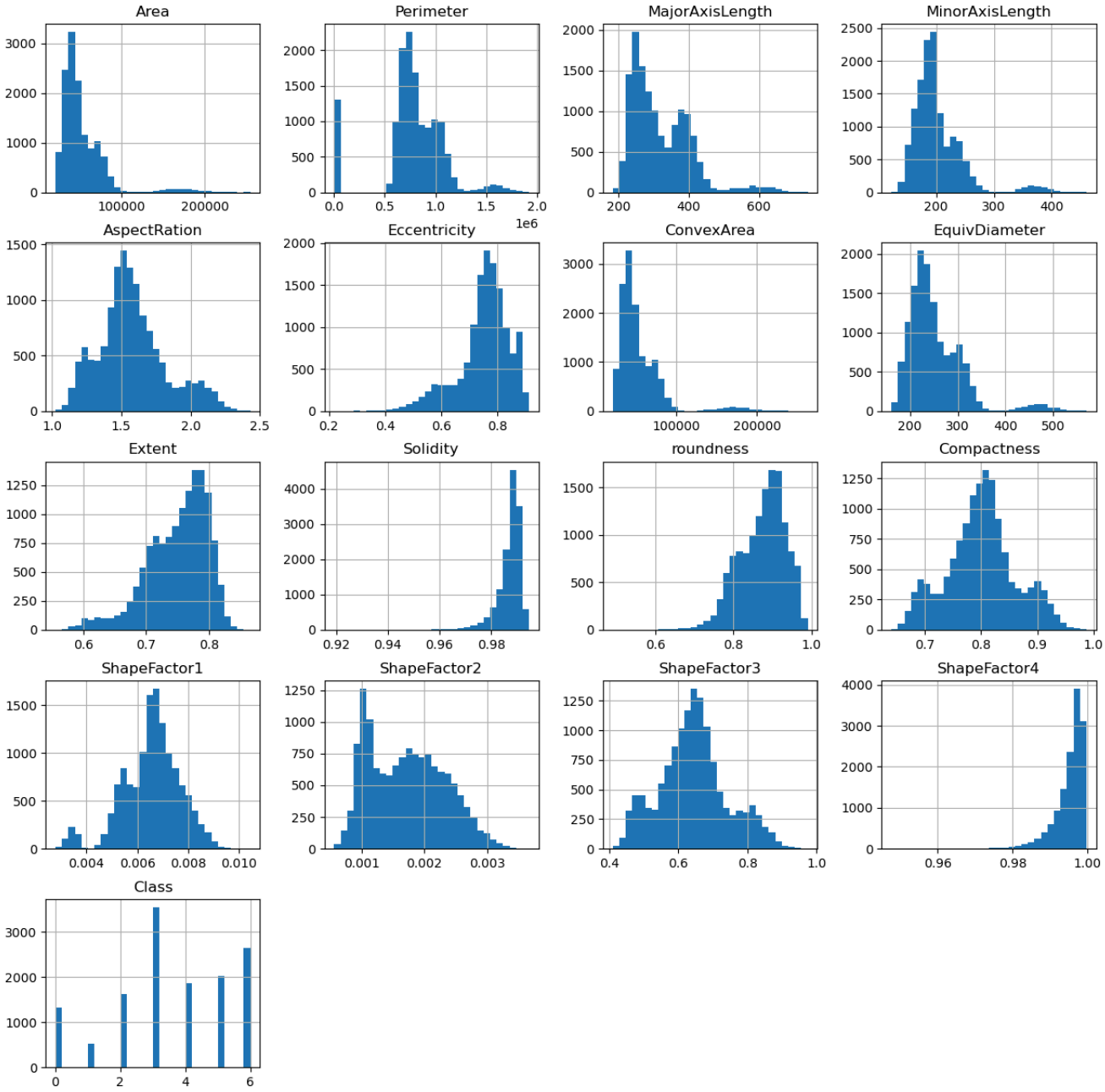


```
sns.scatterplot(x='Extent',y = 'Solidity',hue='Class',data = df)
```

```
<Axes: xlabel='Extent', ylabel='Solidity'>
```




```
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()
```

	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.

```
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 :  (10834, 1)
Testing data 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.append(accuracy_score(y_test,y_pred_test))
```