```
from collections import Counter
from sklearn import preprocessing, svm
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
import sklearn.metrics as metrics
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from imblearn.over_sampling import SMOTE
from collections import Counter
from scipy.stats import randint, loguniform
```

```
df=pd.read_csv("/content/data.csv")
df.head()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity
0	28395	610.291	208.178117	173.888747	1.197191	0.549812
1	28734	638.018	200.524796	182.734419	1.097356	0.411785
2	29380	624.110	212.826130	175.931143	1.209713	0.562727
3	30008	645.884	210.557999	182.516516	1.153638	0.498616
4	30140	620.134	201.847882	190.279279	1.060798	0.333680

```
df.shape
```

```
(13611, 17)
```

```
df.nunique()
```

```
Area               12011
Perimeter          13351
MajorAxisLength    13543
MinorAxisLength    13543
AspectRation       13543
Eccentricity       13543
ConvexArea         12066
EquivDiameter      12011
Extent             13535
Solidity           13522
roundness          13540
Compactness        13543
ShapeFactor1       13521
ShapeFactor2       13506
ShapeFactor3       13543
ShapeFactor4       13532
Class                  7
dtype: int64
```

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

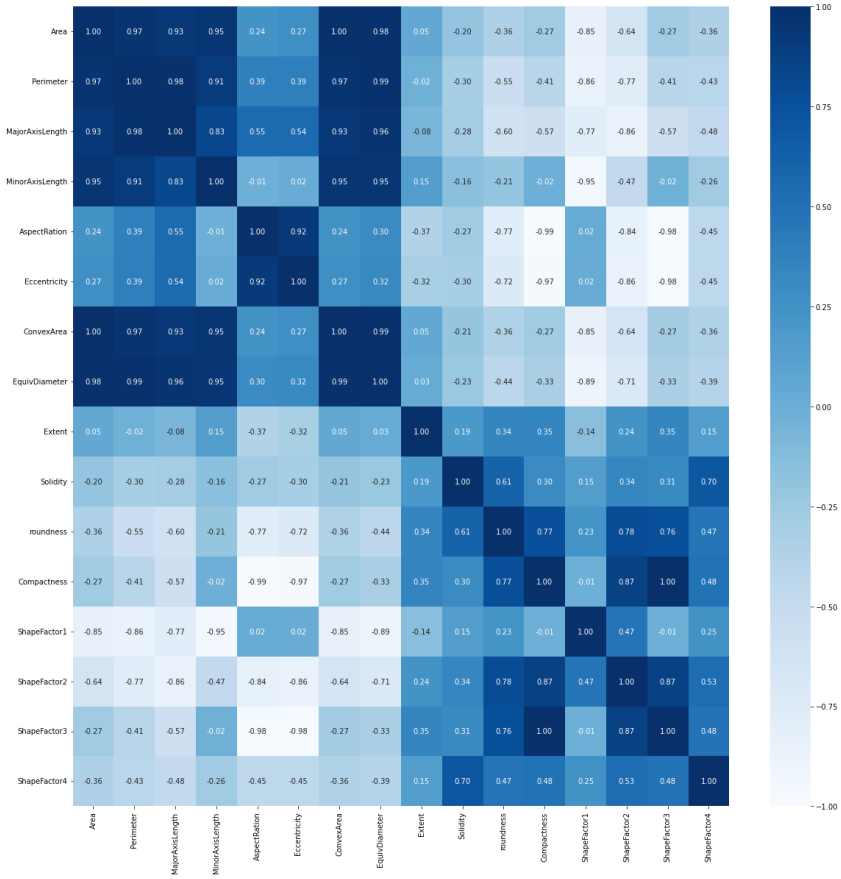
```
DERMASON    3546
SIRA        2636
SEKER       2027
HOROZ       1928
CALI        1630
BARBUNYA    1322
BOMBAY       522
Name: Class, dtype: int64
```

```
#check null value
df.isnull().sum()
```

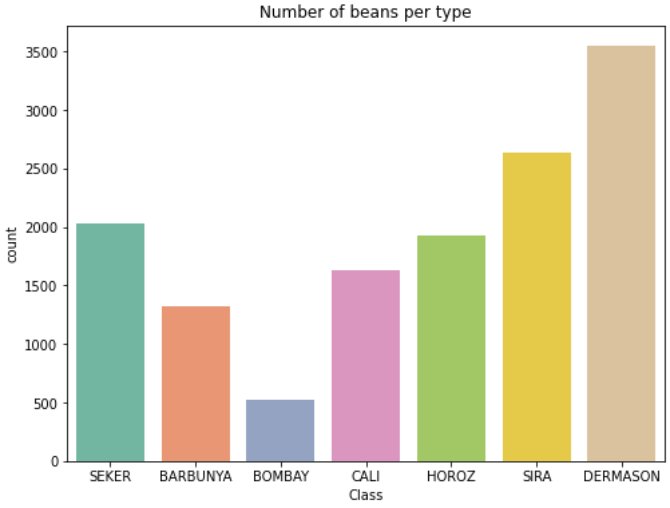
```
Area               0
Perimeter          0
MajorAxisLength    0
MinorAxisLength    0
AspectRation       0
Eccentricity       0
```

```
ConvexArea       0
EquivDiameter    0
Extent           0
Solidity         0
roundness        0
Compactness      0
ShapeFactor1     0
ShapeFactor2     0
ShapeFactor3     0
ShapeFactor4     0
Class            0
dtype: int64
```

```
#check correlation between features
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(),annot=True,cmap="Blues",fmt=".2f",vmin=-1.00,vmax=1.00)
plt.show()
```



```
#display beans per type
plt.figure(figsize=(8,6))
sns.countplot(x=df["Class"],palette="Set2")
plt.title("Number of beans per type")
plt.show()
```

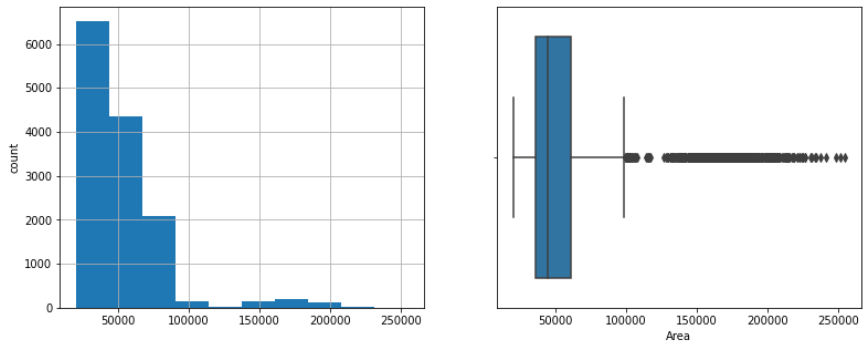



```
#display distibutions and outliet
numerical_features=df.select_dtypes(include=[np.number]).columns
print("numerical features:",numerical_features)
numerical_features=numerical_features.tolist()
```

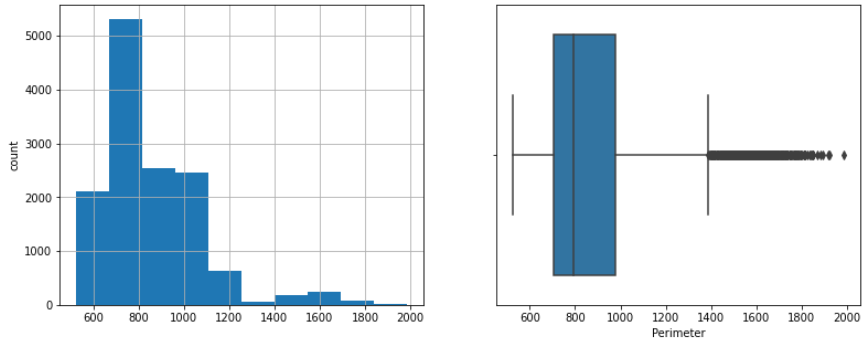
```
numerical features: Index(['Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength',
       'AspectRation', 'Eccentricity', 'ConvexArea', 'EquivDiameter', 'Extent',
       'Solidity', 'roundness', 'Compactness', 'ShapeFactor1', 'ShapeFactor2',
       'ShapeFactor3', 'ShapeFactor4'],
      dtype='object')
```

```
for i in numerical_features:
  print(i)
  print('Skewness : ' , round(df[i].skew(),3))
  plt.figure(figsize = (13,5))
  plt.subplot(1,2,1)
  df[i].hist()
  plt.ylabel('count')
  plt.subplot(1,2,2)
  sns.boxplot(x = df[i])
  plt.show()
```

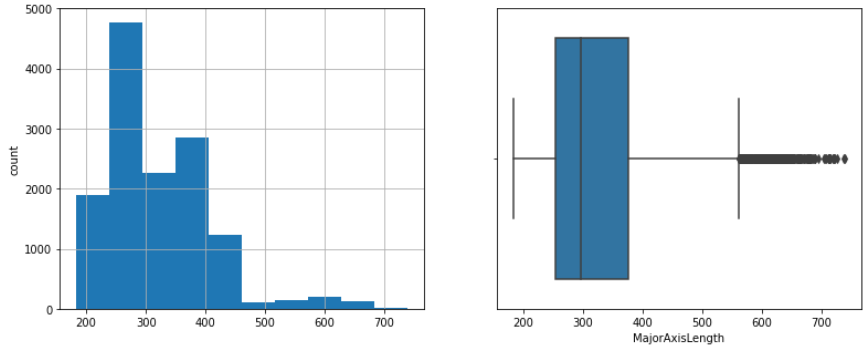
Area
Skewness : 2.953



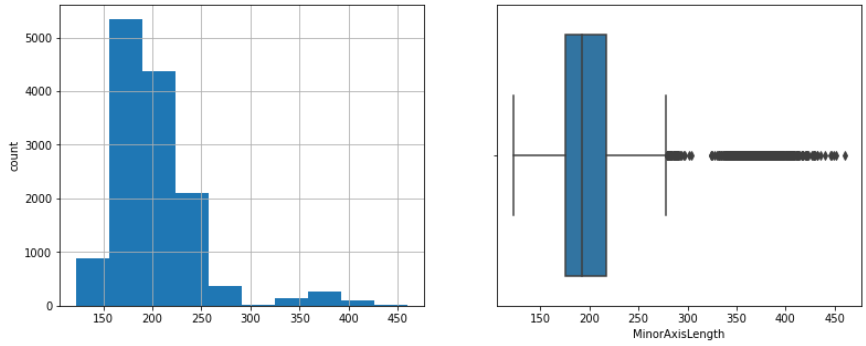
Perimeter
Skewness : 1.626



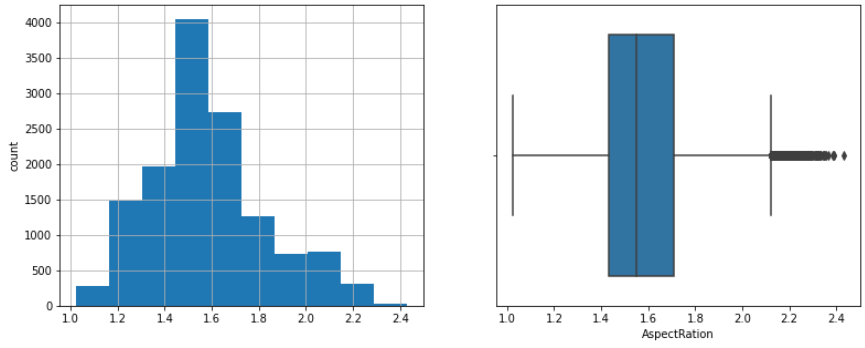
MajorAxisLength
Skewness : 1.358



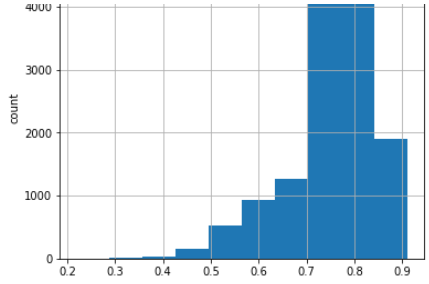
MinorAxisLength
Skewness : 2.238



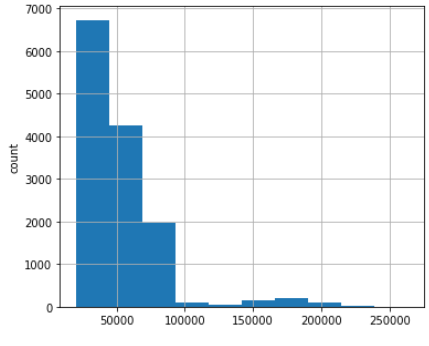
AspectRation
Skewness : 0.583



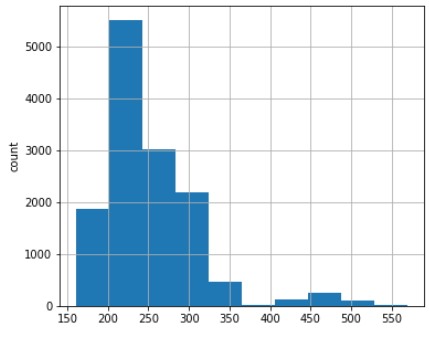
Eccentricity
Skewness : -1.063



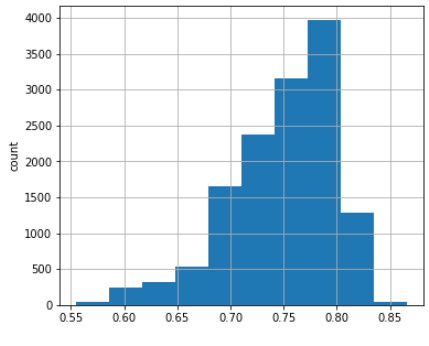
ConvexArea
Skewness : 2.942



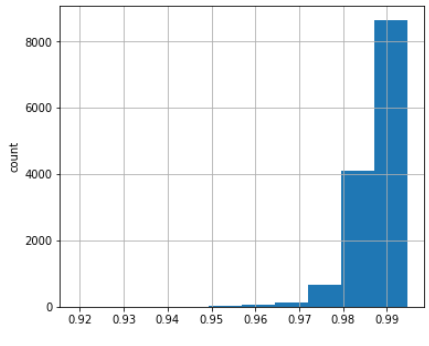
EquivDiameter
Skewness : 1.949



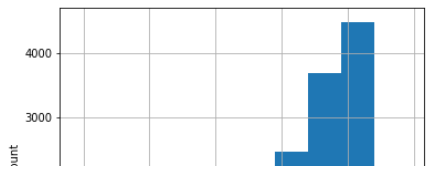
Extent
Skewness : -0.895

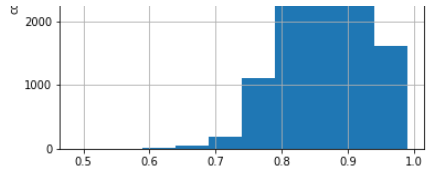
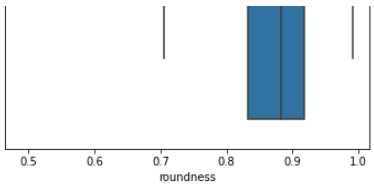


Solidity
Skewness : -2.55

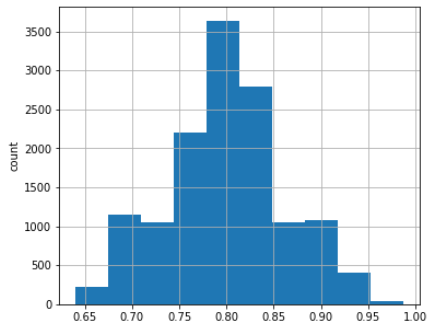


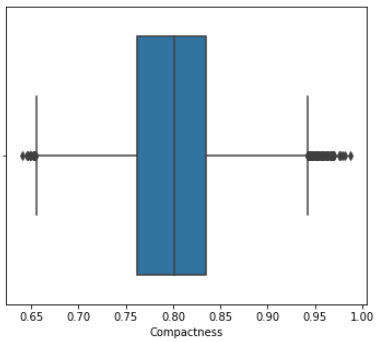
roundness
Skewness : -0.636



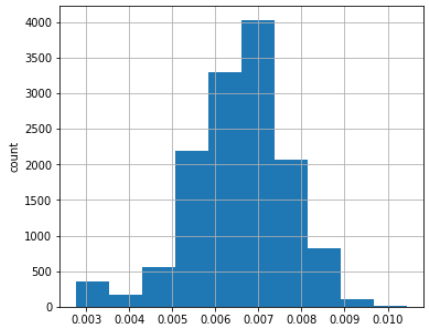



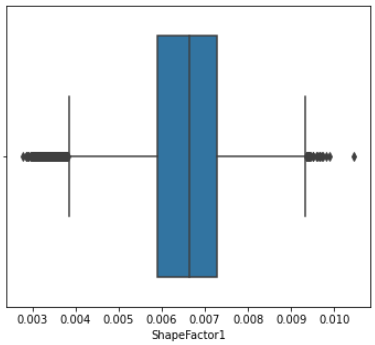
Compactness
Skewness : 0.037



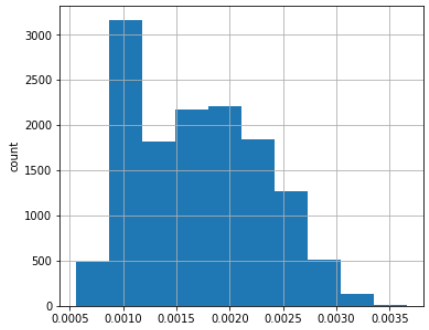


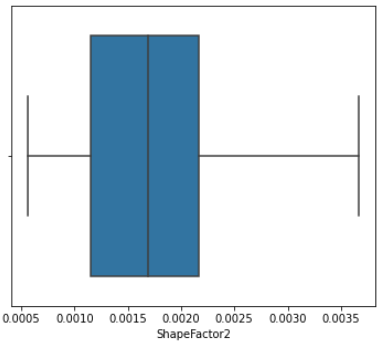
ShapeFactor1
Skewness : -0.534



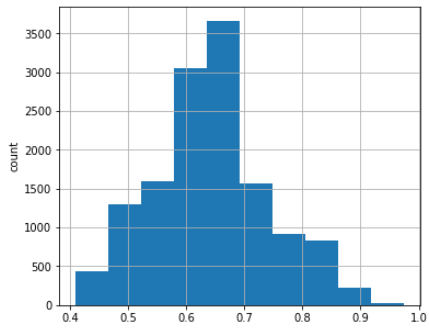


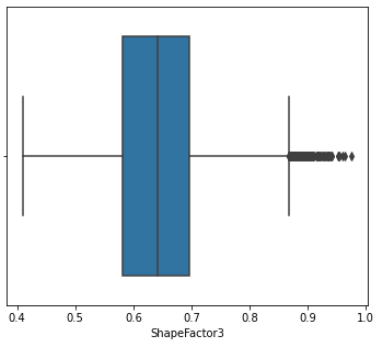
ShapeFactor2
Skewness : 0.301



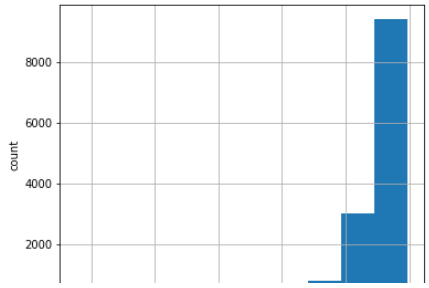


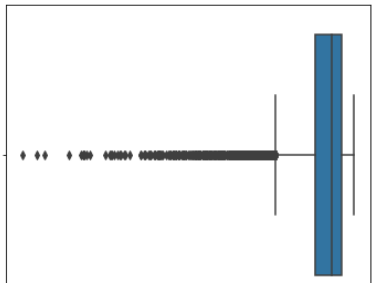
ShapeFactor3
Skewness : 0.242





ShapeFactor4
Skewness : -2.759

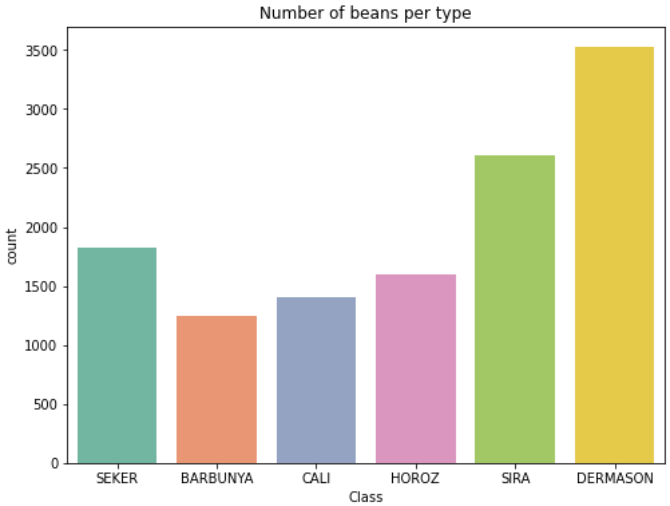





```
# Detect  outliers in the dataset


def detect_outliers(df, features):
    outlier_indices = []

    for c in features:
        Q1 = np.percentile(df[c], 25)
        Q3 = np.percentile(df[c], 75)
        IQR = Q3 - Q1
        outlier_step = IQR * 1.5
        outlier_list_col = df[(df[c] < Q1 - outlier_step) | (df[c] > Q3 + outlier_step)].index
        outlier_indices.extend(outlier_list_col)

    outlier_indices = Counter(outlier_indices)
    multiple_outliers = list(i for i, v in outlier_indices.items() if v > 1)

    return multiple_outliers


data = df.drop(detect_outliers(df,['Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength', 'AspectRation', 'Eccentricity',
                                   'ConvexArea', 'EquivDiameter', 'Extent', 'Solidity', 'roundness', 'Compactness', 'ShapeFactor1',
                                   'ShapeFactor2', 'ShapeFactor3', 'ShapeFactor4']), axis=0).reset_index(drop=True)
print('Number of of samples in the dataset after removing outliers: %d' % len(data))
```

```
Number of of samples in the dataset after removing outliers: 12218
```

```
#Divide data into independent feature and dependent feature
X = data.drop("Class", axis=1)
Y = data['Class']


# Bar Chart to visualize the labels in the output variable
plt.figure(figsize=(8,6))
sns.countplot(x=data["Class"],palette="Set2")
plt.title("Number of beans per type")
plt.show()
```



```
# Convert Class String labels into Integers
from sklearn.preprocessing import LabelEncoder
lab_enc = LabelEncoder()
label_Y = lab_enc.fit_transform(Y)
```

```
# Normalize the input features of the dataset
from sklearn.preprocessing import StandardScaler
normalizer = StandardScaler()
norm_X = normalizer.fit_transform(X)
```

```
# divide data into training and testing set
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=12)
```

```
# Normalize the input features of the dataset
from sklearn.preprocessing import StandardScaler
normalizer = StandardScaler()
scaled_X_train = normalizer.fit_transform(X_train)
scaled__X_test = normalizer.fit_transform(X_test)
```

```
multi_class=LogisticRegression(multi_class="ovr")
```

```
multi_class.fit(scaled_X_train,Y_train)
```

```
▾ LogisticRegression
LogisticRegression(multi_class='ovr')
```

```
y_pred=multi_class.predict(scaled__X_test)
```

```
y_pred
```

```
array(['SIRA', 'SIRA', 'SIRA', ..., 'DERMASON', 'HOROZ', 'DERMASON'],
      dtype=object)
```

```
accuracy = multi_class.score(scaled__X_test,Y_test)
```

```
accuracy
```

```
0.910392798690671
```

```
acc = metrics.accuracy_score(Y_test,y_pred)
```

```
print("Accuracy : ",acc)
```

```
Accuracy :  0.910392798690671
```

✓ 0s completed at 21:07