Coffee Bean Classifier

At

IC Solutions

Submitted by

Nikhilesh Joshi

1AT17EC069

nikhilesh.joshi.1999@gmail.com

Mohammed Samid

1AT16EC075

mohammed619samid@gmail.com

Instructor's name: Abhishek C

Acknowledgement

Foremost, we would like to express our sincere gratitude to our instructor **Abhishek C.** for the continuous support in this internship, for his patience, motivation, enthusiasm, and immense knowledge. His guidance and encouragement helped us to do our best.

We would also like to thank **IC Solutions** for providing this internship opportunity on Python with Machine Learning at such an affordable price and delivering the content and making us learn new technology from basics.

We would like to thank our parents who guided us and provided us the resources in making this project.

Abstract

Coffee Bean is one among the most widely produced edible crop in the world. Production of the crops depends on the quality of seeds. Classification of seeds plays an important role to check out the quality of the seeds.

Here, we are given the data. In the data different types of coffee beans are given with their respective parameters. We get the better properties of the beans by doing Exploratory Data Analysis (EDA). In EDA, we found relation between the various parameters of the beans and came out with few conclusions.

- Dermason are the smallest seeds Bombay are the largest seeds
- Bombay has the highest perimeter, Dermason with least perimeter. Whereas Barbunya and Cali is almost the same.
- Eccentricity of Barbunya is lesser than Eccentricity of Cali. Observed from roundness parameter.
- Area and Perimeter have almost the same effects on the Class
- In heat map, we could find some null values. We eliminated those null values by filling the values with the median of that columns.

After EDA, we prepared Machine Learning model for the data to find its accuracy. To find its accuracy we approached Decision Tree Regression, Random Forest Regression and Logistic Regression. Also, for the same algorithm we dropped ShapeFactor3 parameter. For all of these algorithms we came up with conclusion that Random Forest Regression had the highest accuracy. In Decision Tree Regression we found out that dropping ShapeFactor3 improved our accuracy. Logistic Regression had the least accuracy

About the Company

ICS is a digital service provider that aims to provide software, designing and marketing solutions to individuals and businesses. At ICS, we believe that service and quality is the key to success.

We provide all kinds of technological and designing solutions from Billing Software to Web Designs or any custom demand that you may have. Experience the service like none other!

Some of our services include:

Development - We develop responsive, functional and super-fast websites. We keep User Experience in mind while creating websites. A website should load quickly and should be accessible even on a small view-port and slow internet connection.

Mobile Application - We offer a wide range of professional android, iOS & Hybrid app development services for our global clients, from a start up to a large enterprise.

Design - We offer professional Graphic design, Brochure design & Logo design. We are experts in crafting visual content to convey the right message to the customers.

Consultancy - We are here to provide you with expert advice on your design and development requirement.

Videos - We create a polished professional video that impresses your audience.

Index

Introduction

The internship is based on Machine Learning with Python, was offered by the collaboration of two companies namely **TakeiteasyEngineers** and **IC Solutions**. We had seven days of training followed by three weeks of project and this completes our internship.

Dry beans also called as *Phaseolus vulgaris*, is a herbaceous annual plant grown worldwide for its edible dry seeds or unripe fruit. There is a verity of coffee beans available. According to Turkish Standards Institution, dry beans are classified as "Barbunya, Bombay, Cali, Dermason, Horoz, Selanik and Seeker" depending on their botanical characteristics. For identification of bean varieties helps farmers to use seeds with basic standards for planting and marketing. This can be time consuming and inefficient if done manually. This is where classification using machine learning comes in place.

Using Python as our programming tool we analyzed the dataset given to us. We understood the problems faced by null values and also tackle them. We also understood how to derive conclusion using different graphs also known as Explanatory Data Analysis.

We trained a machine learning model to classify the seeds based on quantifiable qualities of a seed, for example area, perimeter etc. which make it more reliable than a human classifying it. This makes a human available for a more challenging task. Three models were trained for this task namely Logistic Regression, Decision Tree and Random Forest.

Problem Statement and Objective

Problem statement

The issue with a human classifying the coffee seeds is, we are wasting the person's time and energy in doing something so trivial. Moreover, we cannot guarantee the results to be correct because they are not based on soothing quantitative, rather they are based on the intuition of the person doing the job. This can vary person to person and give us inconsistent results. If we take into account the quantitative features of a seed and let a machine handle this it makes things a lot easier.

Objective

- We analyze the given data that is required to classify seeds.
- Then we need to look for any kind of relations that can be helpful in developing our machine learning model.
- Clean the data by handling the null values.
- Do the preprocessing on the data to make it ML algorithm ready.
- Then we train different models using different scenarios to find the most relevant model for our dataset and pick out the model giving us the best results.

Requirement Specification

Hardware Requirements

A working laptop/desktop.

Software Requirements

Anaconda:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment.

Jupyter Notebook:

This is a part of Anaconda, so to install this Anaconda should installed as a prerequisite.

The modules we will be using are:

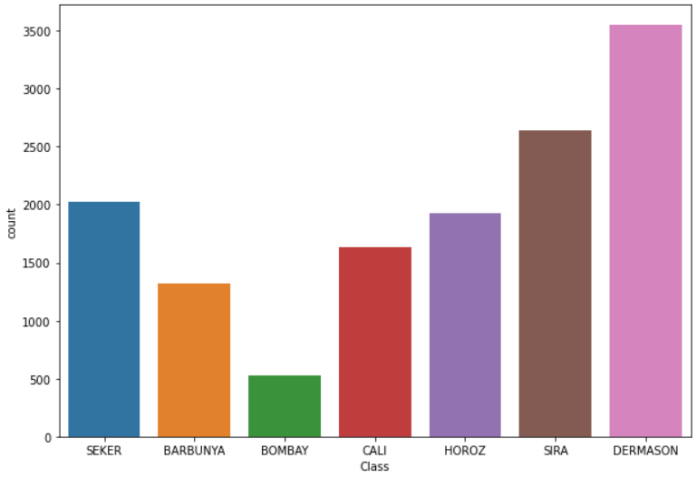
- NumPy
- Pandas
- Matplotlib
- Seaborn
- Sklearn

An alternate to the above mentioned, we can also use Google Colab which is a cloud based machine learning tool available freely.

Explanatory Data Analysis

Count of Classes

```
In [3]: plt.figure(figsize=(10,7))
        sns.countplot(x= 'Class', data= df)
        plt.show()
```

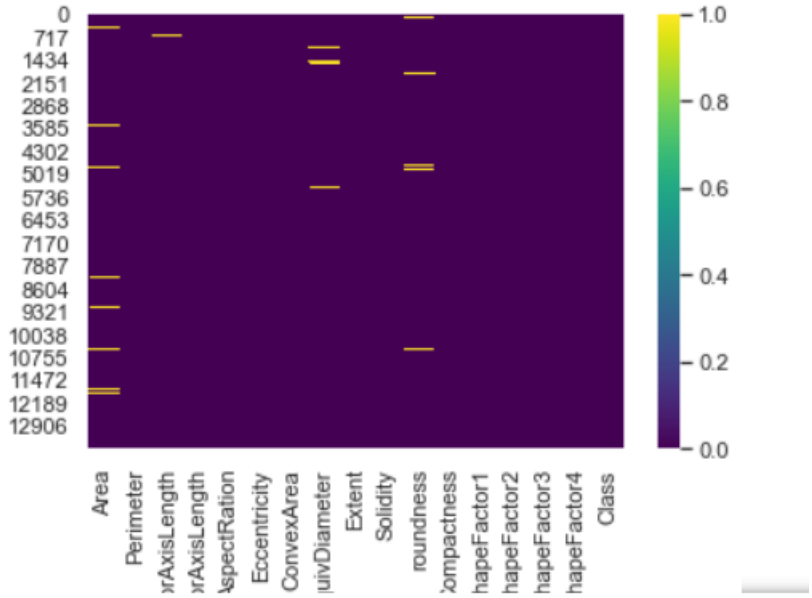


- Dermason count is the highest
- Bombay has the least count
- Dermason has the highest count and Bombay has a very low count , this can make our machine learning model biased and hence we need to take it into account while making a model.

Heat map

```
In [49]: sns.heatmap(df.isnull(),cmap='viridis')
```

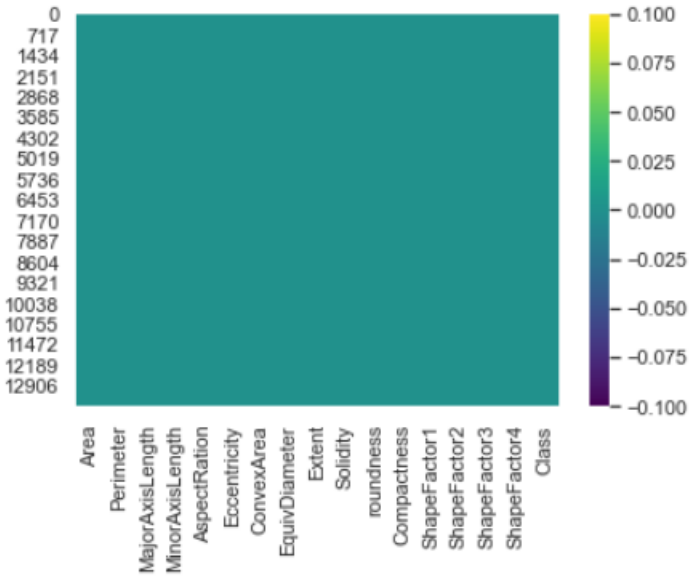
```
Out[49]: <AxesSubplot:>
```



- This plot shows us the columns that contain null values.
- We observe the below columns contain null values:
    - Area
    - EquivDiameter
    - Roundness
    - Perimeter
- We will fill these columns with the median of the respective columns.

Heat map

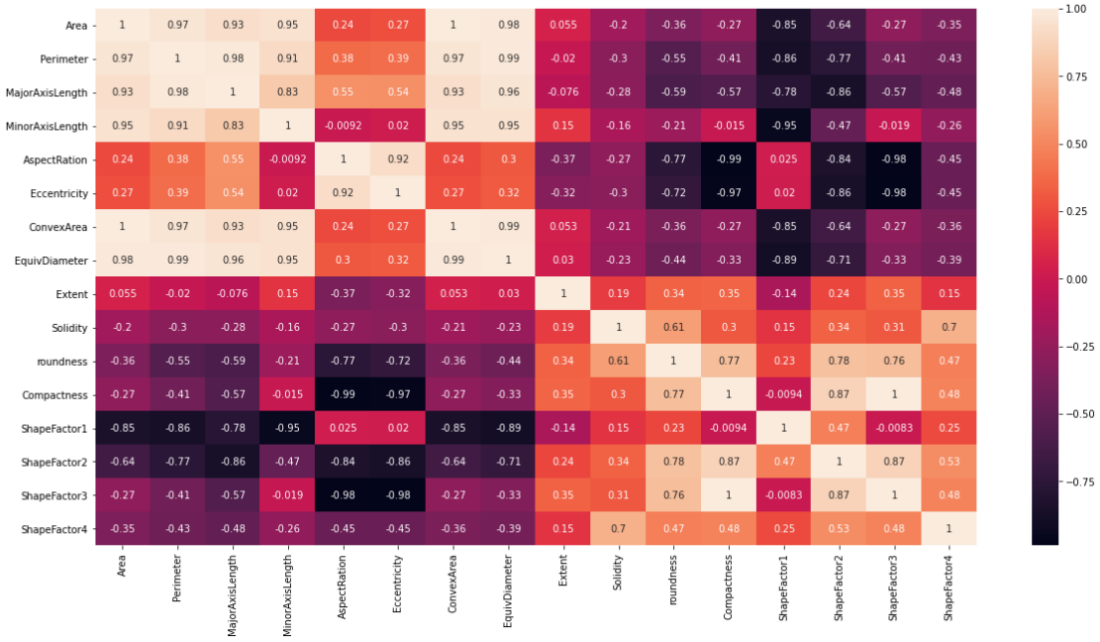
```
In [25]: sns.heatmap(df.isnull(),cmap='viridis')
Out[25]: <AxesSubplot:>
```



Here we replaced all the null values with the median of the respective column. To verify our replacement we use this plot.

Correlation Matrix

```
In [5]: plt.figure(figsize=(20,10))
        sns.heatmap(df.corr(),annot=True)
Out[5]: <AxesSubplot:>
```



- We see that the two pair of attributes:
    - Area and ConvexArea
    - Compactness and ShapeFactor3

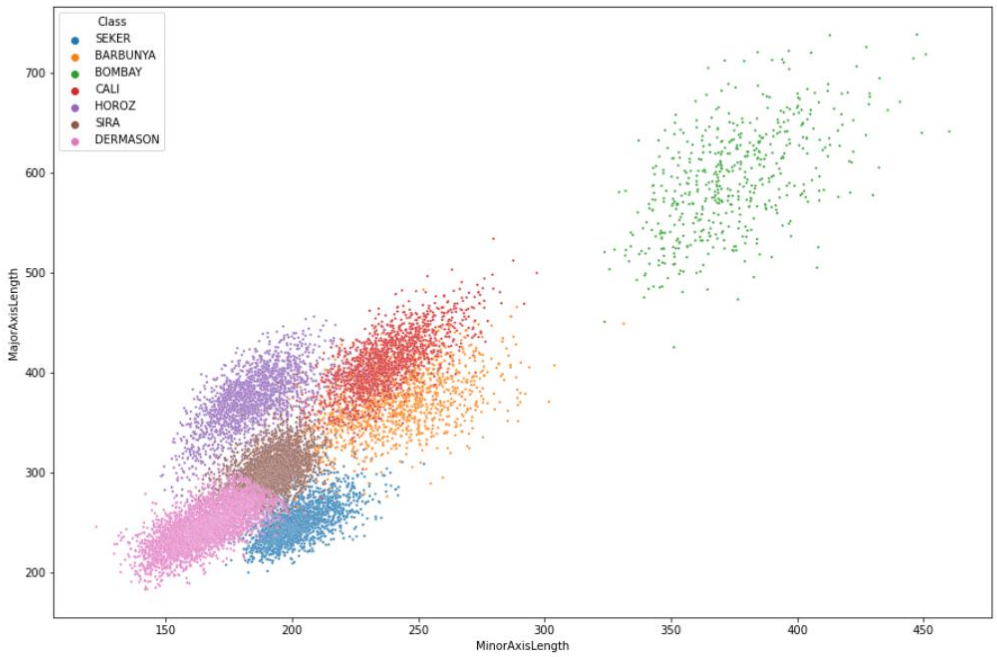
have a correlation of 1. This means that one the factors in a pair would have the same impact on the ML model, so depending on the model we can drop one of the attributes in the pairs.

- Compactness and AspectRatio have the least correlation ratio. It is -0.99 which is very close to -1. This means the impact of them on the classifier is exactly opposite (approximately speaking since it not a perfect -1)
- There are column with almost 0 correlation, they do not have any trend that can relate them with each other ShapeFactor1 and

ShapeFactor3 have the column with the correlation with the one nearest to 0 having a value of -0.0083

Relationship of Major Axis Length and Minor Axis Length

```
In [6]: f, ax = plt.subplots(figsize=(15, 10))
        sns.scatterplot(x="MinorAxisLength", y="MajorAxisLength", s=5, hue="Class", data=df)
Out[6]: <AxesSubplot:xlabel='MinorAxisLength', ylabel='MajorAxisLength'>
```

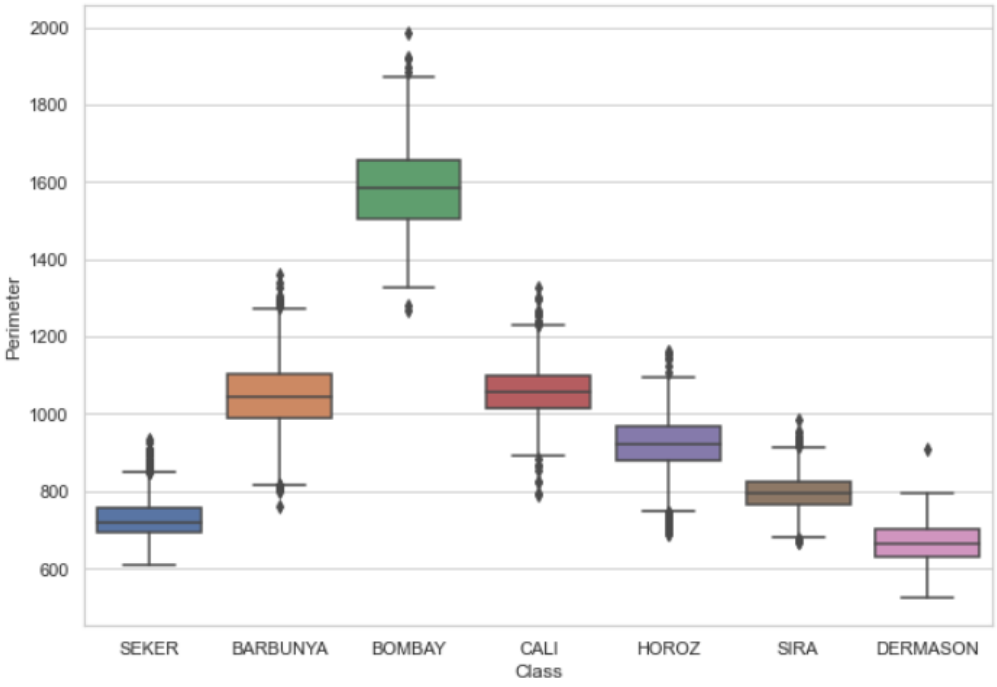


We can see a lot going on in this graph

- We can observe that Dermason seeds are the smallest and Bombay are the biggest.
- If we look closely, Dermason, Sira, Cali and Bombay seeds make a linear trend. This would mean they are more close to a circle.
- If we deviate a little from the linear plot and increase the Major Axis length we get Horoz and by reducing we get Seker, these are more towards the ovular shape.

Class vs Perimeter

```
In [11]: plt.figure(figsize=(10,7))
         sns.set(style='whitegrid')
         sns.boxplot(x="Class",y="Perimeter", data=df)
         plt.show()
```

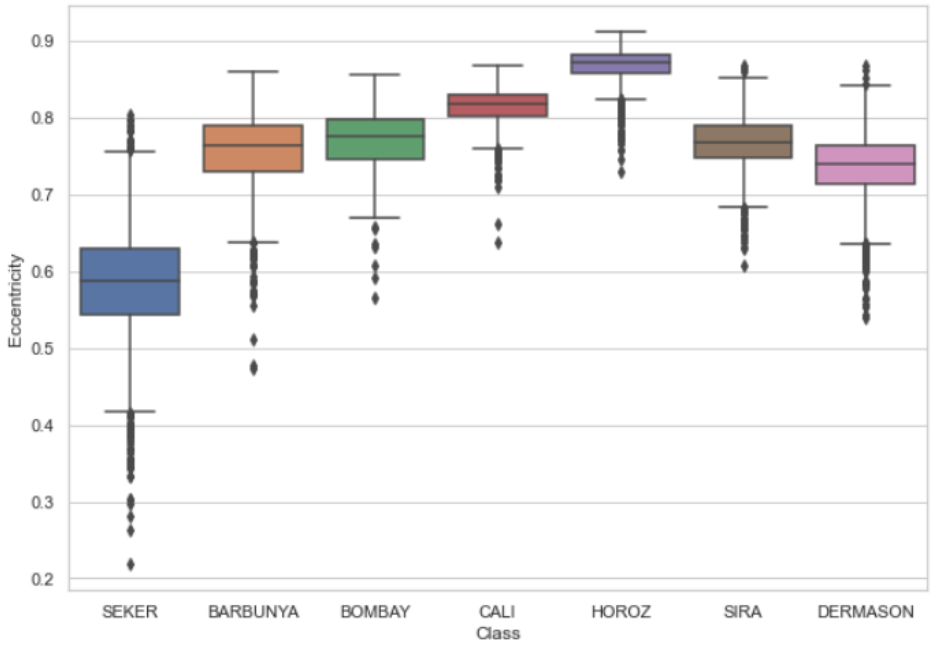


- Here for most of the seeds we see a major difference between the perimeters. We can even make a sorted list for it(ignoring the outliers):
    - Bombay
    - Barbunya almost same as Cali
    - Horoz
    - Sira
    - Seker
    - Dermason

  the list is in non-increasing order

Class vs Eccentricity

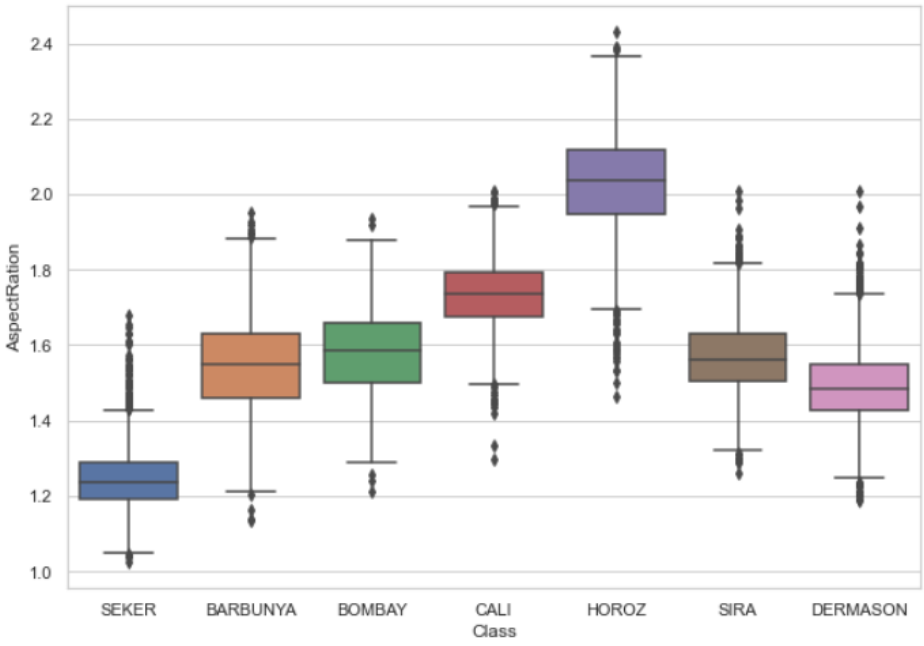
```
In [13]: plt.figure(figsize=(10,7))
         sns.set(style='whitegrid')
         sns.boxplot(x="Class",y="Eccentricity", data=df)
         plt.show()
```



In this plot we see the ambiguity caused by perimeter in the previous graph is cleared here. In the previous graph we say Barbunya and Cali had almost the same perimeter, this is not the case when we look at this graph.

Class vs Aspect ratio

```
In [8]: plt.figure(figsize=(10,7))
        sns.set(style='whitegrid')
        sns.boxplot(x="Class",y="AspectRation", data=df)
        plt.show()
```



Although the seeds might be bigger or smaller (Based on perimeter, area, major axis length etc.), the aspect ratio of Barbunya, Bombay, Sira and Dermason is comparable. This would mean the shape is almost similar to each other. This can be verified with the scatter plot seen in the previous observations

Class vs ShapeFactor1

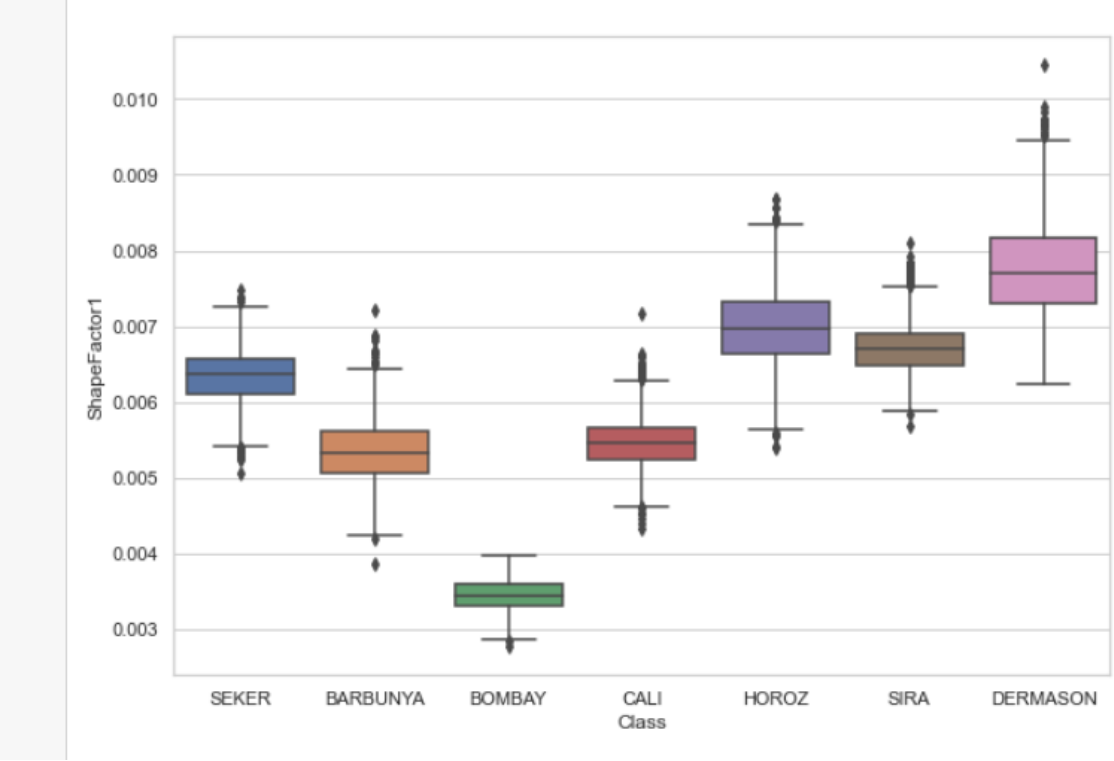
Code:

```
plt.figure(figsize=(10,7))

sns.set(style='whitegrid')

sns.boxplot(x="Class",y= “ShapeFactor1”, data=df)

plt.show()
```



- Dermason has the highest value of ShapeFactor1. Even the outliers are above the second highest.
- Bombay has the least value. Even the outliers are below the second least.
- This attribute makes it a very good candidate for classifying the two.

Class vs ShapeFactor2

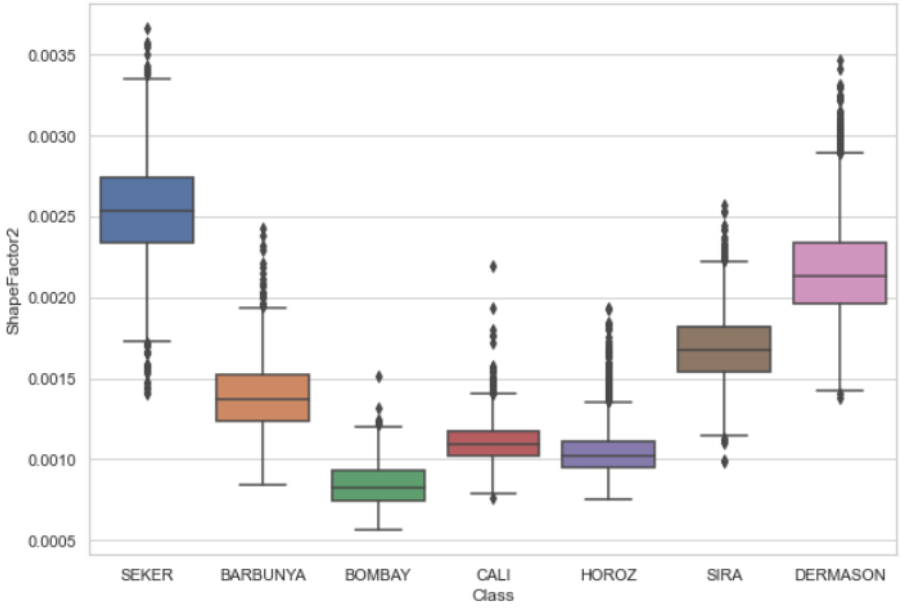
Code:

```
plt.figure(figsize=(10,7))

sns.set(style='whitegrid')

sns.boxplot(x="Class",y= “ShapeFactor2”, data=df)

plt.show()
```



- Seker has the highest value in this plot. This could be helpful to classify a seed as Seker.
- Unlike in most of the attributes Barbunya and Cali have a significant difference here.

Preparing Machine Learning Model

In this Machine Learning Model , We approach Decision Tree Regression, Random Forest Regression and Logistic Regression.

Train Test Split

Considering all attributes

```
In [27]: #Training and testing split
         from sklearn.model_selection import train_test_split
         df = df.sample(frac = 1)
         x = df.drop('Class', axis=1)
         y = df['Class']
         X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=101)
```

- Here, We are first selecting the data frames to be tested and split.
- While training and testing , for X values we consider all the other rows other than ‘Class’ . As you can see in above figure we are using df.drop to drop the Class.
- For Y values we consider only ‘Class’.
- The data is now trained with various algorithms and after testing the model the accuracy of prediction is noted.

Train Test Split (for dropping ShapeFactor3)

In [35]:
```
#Training and testing split
from sklearn.model_selection import train_test_split
df = df.sample(frac = 1)
x = df.drop(['Class', 'ShapeFactor3'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=101)
```

- We consider X values as all the rows other than ‘Class’ and ‘ShapeFactor3’.
- Considering Y values as Class.

DECISION TREE REGRESSION

Training the Model

```
In [28]: #Training model(Decision tree)
         from sklearn.tree import DecisionTreeClassifier
         d_tree = DecisionTreeClassifier()
         d_tree.fit(X_train,y_train)

Out[28]: DecisionTreeClassifier()
```

- We import DecisionTreeClassifier is from sklearn.tree.
- We create an object ‘d_tree’ under the class DecisionTreeClassifier.
- We call the function under the class d_tree and pass the values.

Testing the Model

```
In [29]: pred_dtree = d_tree.predict(X_test)
         from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
         print(classification_report(y_test, pred_dtree))
```

```
              precision    recall  f1-score   support

         0.0       0.91      0.90      0.90       386
         1.0       1.00      1.00      1.00       156
         2.0       0.91      0.92      0.92       483
         3.0       0.89      0.87      0.88      1070
         4.0       0.89      0.92      0.91       566
         5.0       0.93      0.93      0.93       603
         6.0       0.82      0.83      0.83       820

    accuracy                           0.89      4084
   macro avg       0.91      0.91      0.91      4084
weighted avg       0.89      0.89      0.89      4084
```

- We create an object ‘pred_dtree’ under the class d_tree.predict(X_test).
- We import classification_report, confusion_matrix, accuracy_score from sklearn.metrics.
- Then, classification_report values are printed.

```
In [30]: print(accuracy_score(y_test, pred_dtree))
```

```
0.8915279138099902
```

- We obtain the accuracy_score as 0.8915(89.15%).

DECISION TREE REGRESSION (after dropping ShapeFactor3)

Training the Model

```
In [36]: #Training model(Decision tree)
         from sklearn.tree import DecisionTreeClassifier
         d_tree = DecisionTreeClassifier()
         d_tree.fit(X_train,y_train)

Out[36]: DecisionTreeClassifier()
```

Testing the Model

```
In [37]: pred_dtree = d_tree.predict(X_test)
         from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
         print(classification_report(y_test, pred_dtree))
```

```
              precision    recall  f1-score   support

         0.0       0.89      0.87      0.88       388
         1.0       0.99      0.99      0.99       143
         2.0       0.89      0.91      0.90       477
         3.0       0.89      0.89      0.89      1073
         4.0       0.93      0.91      0.92       605
         5.0       0.94      0.93      0.93       627
         6.0       0.82      0.84      0.83       771

    accuracy                           0.89      4084
   macro avg       0.91      0.91      0.91      4084
weighted avg       0.89      0.89      0.89      4084
```

```
In [38]: print(accuracy_score(y_test, pred_dtree))
```

```
0.8942213516160626
```

- Decision Tree after dropping ShapeFactor3, we obtain accuracy_score as 0.8942(89.42%).
- There is not much difference between t he accuracy_score without dropping any rows and accuracy_score after dropping ShapeFactor3.

RANDOM FOREST REGRESSION

Training the Model

```
In [31]: from sklearn.ensemble import RandomForestClassifier
         rfc = RandomForestClassifier()
         rfc.fit(X_train,y_train)

Out[31]: RandomForestClassifier()
```

- We import RandomForestClassifier from sklearn.ensemble.
- We create an object ‘rfc’ under the class RandomForestClassifier.
- We call the function under rfc and pass the values.

Testing the Model

```
In [32]: pred_rf = rfc.predict(X_test)
         print(classification_report(y_test, pred_rf))
         print(accuracy_score(y_test, pred_rf))
```

```
              precision    recall  f1-score   support

         0.0       0.94      0.90      0.92       386
         1.0       1.00      1.00      1.00       156
         2.0       0.92      0.94      0.93       483
         3.0       0.91      0.93      0.92      1070
         4.0       0.95      0.94      0.95       566
         5.0       0.95      0.96      0.95       603
         6.0       0.89      0.87      0.88       820

    accuracy                           0.92      4084
   macro avg       0.94      0.93      0.94      4084
weighted avg       0.92      0.92      0.92      4084

0.9243388834476004
```

- We Create an object ‘pred_rf’ under the class rfc.predict(X_test).
- Then print the classification_report.
- We obtain the accuracy_score in Random Forest Regression as 0.9244(92.44%).

RANDOM FOREST REGRESSION (after dropping ShapeFactor3)

Training the Model

```
In [39]: from sklearn.ensemble import RandomForestClassifier
         rfc = RandomForestClassifier()
         rfc.fit(X_train,y_train)

Out[39]: RandomForestClassifier()
```

Testing the Model

```
In [40]: pred_rf = rfc.predict(X_test)
         print(classification_report(y_test, pred_rf))
         print(accuracy_score(y_test, pred_rf))
```

	precision	recall	f1-score	support
0.0	0.94	0.88	0.91	388
1.0	0.99	1.00	1.00	143
2.0	0.92	0.94	0.93	477
3.0	0.91	0.93	0.92	1073
4.0	0.96	0.95	0.95	605
5.0	0.95	0.94	0.94	627
6.0	0.87	0.88	0.87	771
accuracy			0.92	4084
macro avg	0.94	0.93	0.93	4084
weighted avg	0.92	0.92	0.92	4084

```
0.9231145935357493
```

- Random Forest Regression after dropping ShapeFactor3, we obtain accuracy_score as 0.9231(92.31%).
- There is not much difference between the accuracy_score without dropping any rows and accuracy_score after dropping ShapeFactor3.

LOGISTIC REGRESSION

Training the Model

```
In [41]: from sklearn.linear_model import LogisticRegression

         logmodel = LogisticRegression()
         logmodel.fit(X_train,y_train)
```

```
/home/nick/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Out[41]: LogisticRegression()
```

- We import LogisticRegression from sklearn,linear_model
- We create an object ‘logmodel’ under the class LogisticRegression
- We call the function under logmodel and pass the values

Testing the Model

```
In [36]: predictions = logmodel.predict(X_test)
         from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
         accuracy_score(y_test, predictions)
         print(classification_report(y_test,predictions))
         print(confusion_matrix(y_test, predictions))
         from sklearn import metrics

         print(metrics.r2_score(y_test, predictions))

         print(metrics.accuracy_score(y_test, predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.82      0.19      0.30       387
         1.0       0.97      0.99      0.98       174
         2.0       0.60      0.80      0.68       494
         3.0       0.69      0.85      0.76      1008
         4.0       0.51      0.50      0.51       588
         5.0       0.45      0.31      0.37       597
         6.0       0.53      0.58      0.55       836

    accuracy                           0.60      4084
   macro avg       0.65      0.60      0.59      4084
weighted avg       0.61      0.60      0.58      4084

[[ 72   0 237   0  66   6   6]
 [  0 173   1   0   0   0   0]
 [ 13   6 394   0   9  61  11]
 [  0   0   0 854  46   5 103]
 [  2   0  25  22 296  30 213]
 [  1   0   0 292  17 186 101]
 [  0   0   2  77 149 122 486]]
0.42608505266737895
0.6025954946131243
```

- We create an object predictions under the class logmodel.predict(X_test).
- Then, we import classification_report, confusion_matrix, accuracy_score from sklearn.metrics.
- We obtain accuracy_score as 0.6025(60.25%).
- After obtaining the accuracy_score we print the classification_report and confusion_matrix.
- Later, we import metrics from sklearn.
- Print the ‘metrics.r2_score’ and ‘metrics.accuracy_score’.

LOGISTIC REGRESSION (after dropping ShapeFactor3)

Training the Model

```
In [40]: from sklearn.linear_model import LogisticRegression

         logmodel = LogisticRegression()
         logmodel.fit(X_train,y_train)

C:\Users\Family\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:940: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

Out[40]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=1, l1_ratio=None, max_iter=100,
                            multi_class='auto', n_jobs=None, penalty='l2',
                            random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                            warm_start=False)
```

Testing the Model

```
In [41]: predictions = logmodel.predict(X_test)
         from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
         accuracy_score(y_test, predictions)
         print(classification_report(y_test,predictions))
         print(confusion_matrix(y_test, predictions))
         from sklearn import metrics

         print(metrics.r2_score(y_test, predictions))

         print(metrics.accuracy_score(y_test, predictions))

                       precision    recall  f1-score   support

                  0.0       0.54      0.06      0.11       410
                  1.0       0.99      1.00      0.99       150
                  2.0       0.53      0.83      0.65       457
                  3.0       0.68      0.86      0.76      1100
                  4.0       0.49      0.46      0.48       577
                  5.0       0.44      0.25      0.32       585
                  6.0       0.52      0.58      0.55       805

             accuracy                           0.58      4084
            macro avg       0.60      0.58      0.55      4084
         weighted avg       0.57      0.58      0.55      4084

         [[ 26   1 307   0  62   2  12]
          [  0 150   0   0   0   0   0]
          [  0   1 380   0   7  53  16]
          [  0   0   0 947  40  27  86]
          [ 17   0  32  22 265  24 217]
          [  0   0   0 318  19 149  99]
          [  5   0   1 112 143  80 464]]
         0.3493407345820294
         0.5830068560235063
```

- Logistic Regression after dropping ShapeFactor3, we obtain accuracy_score as 0.5830(58.30%).
- There is difference between the accuracy_score without dropping any rows and accuracy_score after dropping ShapeFactor3. It is around 0.019 (1.9%)

Machine Learning Model Chart

Serial Number	ML Algorithm Used	Accuracy Score (approx. 4 decimal places)
01	RANDOM FOREST REGRESSION	0.9244
02	RANDOM FOREST REGRESSION (After Dropping ShapeFactor3)	0.9231
03	DECISION TREE REGRESSION (After Dropping ShapeFactor3)	0.8942
04	DECISION TREE REGRESSION	0.8915
05	LOGISTIC REGRESSION	0.6025
06	LOGISTIC REGRESSION (After Dropping ShapeFactor3)	0.5830

Hurdles

The accuracy at first wasn't satisfactory, it was then that we realized that the data given on Dermason is a lot and the data given on Bombay is very less. This was causing the machine learning model to be biased and not give satisfactory results. So, we shuffled the data before feeding it to the model. This made the model better and gave better results.

Conclusion

We conclude that, Random Forest is the most efficient ML model for our dataset. It is the only model that provides an accuracy of more than 90%

We also conclude that, all the columns are not required to make an efficient ML model, if we find satisfactory correlation between two attributes, it may be a good idea to drop a column. This reduces the computation and time required to train a model. Although, this must be done while keeping in mind that the accuracy of the model does not drop.

Shuffling is not required always, but it usually is a good idea to do it. In some cases even the randomness of the test and train spliter in sklearn module is good enough to make the right split, but as in case of our data, it didn't seem good enough.

Bibliography

- https://stackoverflow.com/
- https://towardsdatascience.com/
- https://seaborn.pydata.org/
- https://practice.geeksforgeeks.org/