Classification of Beans

Jason Hendrix

5/4/2022

# Introduction

In this analysis, we will be using methods from [this book](https://rafalab.github.io/dsbook/) to consider over 10,000 picture observations of 7 different bean types. Our goal is to produce machine learning models that use characteristics of beans to predict their type. We will then evaluate the performance of each model to determine which one maximizes accuracy when evaluated on a separate test set. The data set can be found [on GitHub](https://github.com/jasonhendrix9/IS4300-Final-Project), and it was originally pulled from the University of California, Irvine, which may be found [here](https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset).

We begin by examining the 16 variables included in the data, which are defined as follows:

1. *Area*: the number of pixels inside the boundaries of the bean
2. *Perimeter*: length of the border of the bean
3. *Major Axis Length*: length of the longest line whose ends touch the boundary of the bean
4. *Minor Axis Length*: length of the longest line whose ends touch the boundary of the bean when the bean is placed perpendicular to the Major Axis
5. *Aspect Ratio*:
6. *Eccentricity*: how near the ellipse that fits the around bean is to a circle (a value of 1 indicates a perfect circle)
7. *Convex Area*: the area of the smallest convex polygon that completely contains the bean
8. *Equivalent Diameter*: diameter of the circle that has the same area as the bean
9. *Extent*: the number of pixels in the image divided by the Area of the bean
10. *Solidity*:
11. *Roundness*:
12. *Compactness*:
13. *Shape Factor 1*:
14. *Shape Factor 2*:
15. *Shape Factor 3*:
16. *Shape Factor 4*:

We also have a variable for the bean type:

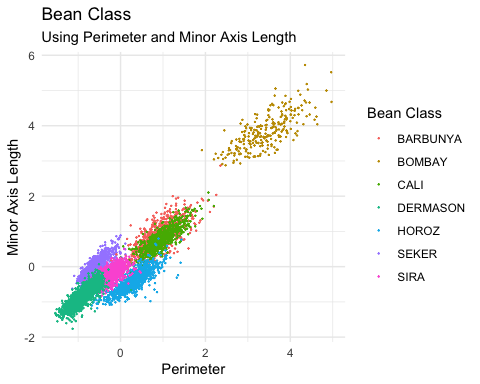
1. *Class*: Bean type - Barbunya, Bombay, Cali, Dermason, Horoz, Seker, or Sira

# Visualizations

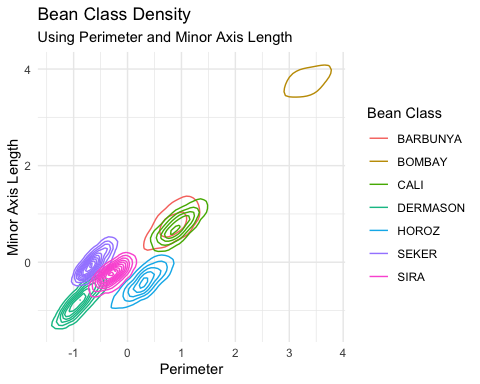
While we will begin with training our models using variables 1-16 to predict variable 17, we will select a few of these variables to create a few plots of Class. We will see the significance of the variables that we visualize here later when we train the models, in the Analysis section.

## Perimeter vs Minor Axis Length

We will first consider the Perimeter and Minor Axis Length variables.



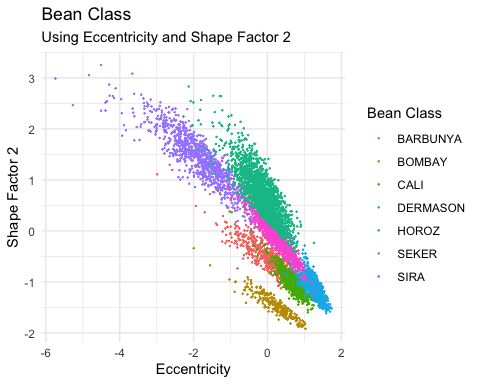
If we switch our view from considering each bean and instead consider the density of the scatter plot, we can better visualize the clustering in our data.

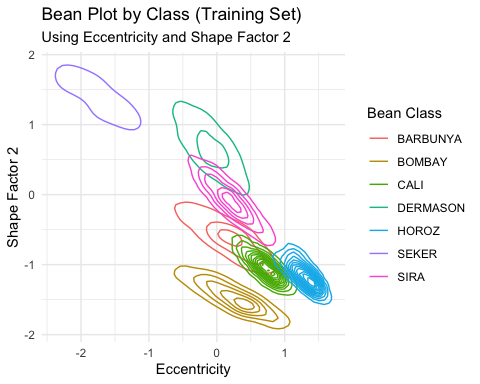


While there is some overlap (especially among Barbunya and Cali beans), most of the bean types have semi-distinct representations as a combination of these two variables.

## Eccentricity vs Shape Factor 2

Now, we will generate the same plot types of the bean class using another couple of variables - Eccentricity and Shape Factor 2.





We again find that most of the bean types have nearly-distinct clusters, yet there is still much overlap between Barbunya and Cali beans.

# Analysis

We will now turn our attention to training models to classify the bean types. We begin with a decision tree, which will find and partition all relevant variables in a waterfall-type format.

## Decision Tree

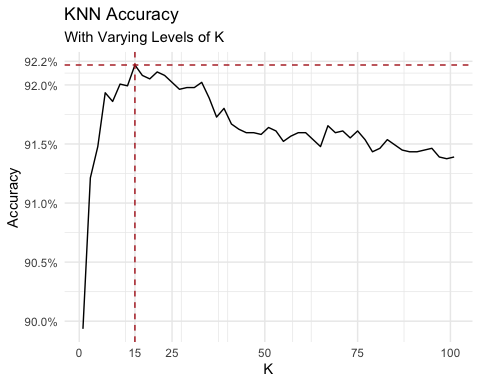
Diagram

Description automatically generated

Using the decision tree above, we attain an 87.6% accuracy in correctly predicting Class. The most significant variables to this model are Perimeter and Minor Axis Length, which we have already visualized in a previous section.

We transition now to using the K Nearest Neighbor approach, which will evaluate the Class of the nearest K points to each new point to determine the most likely Class for the new point. First, we must find an optimal value for K.

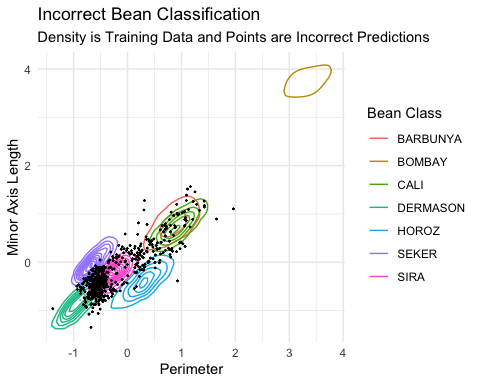
## Optimize K for Accuracy



Our maximum accuracy is 92.2% at K = 15, which is considerably better than the accuracy of the decision tree.

## KNN Performance

Since there were a few beans that were incorrectly classified by the KNN model, it might be worthwhile to consider where those beans fall on the two most important variables to the decision tree.

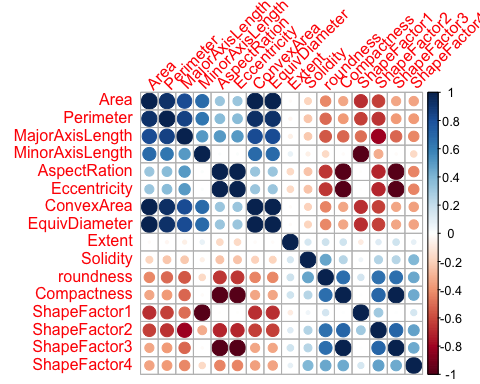


Most of these errors on the plot above are in the overlap of Dermason and Sira in Perimeter and Minor Axis Length.

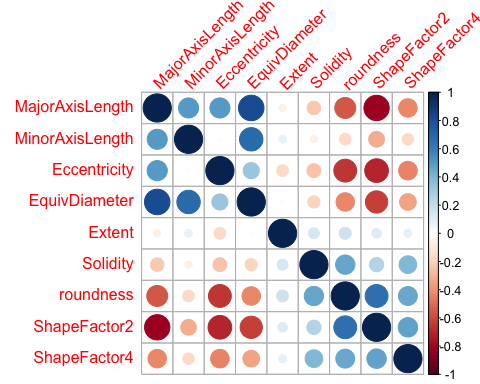
## Correlation of Variables

While we have created a very accurate model using all of the variables, we should seek to reduce the number of dimensions, if possible. A large reduction in dimensionality is often worth a trade-off with a slight reduction in accuracy.

If two variables are highly correlated, they won’t be independent. Since two dependent variables convey about the same information to the model, we will remove highly correlated variables and retrain the models. In the following chart, the size of a dot indicates the magnitude of the correlation between the variables that it represents, and a red dot indicates negative correlation and a blue dot indicates positive correlation.



Now, let’s remove all of the variables that have at least 90% correlation with another variable.



This reduction makes the relationships appear much cleaner.

Now, we can retrain our models and evaluate impact to the accuracy for each.

# Decision Tree with Correlated Variables Removed

Diagram

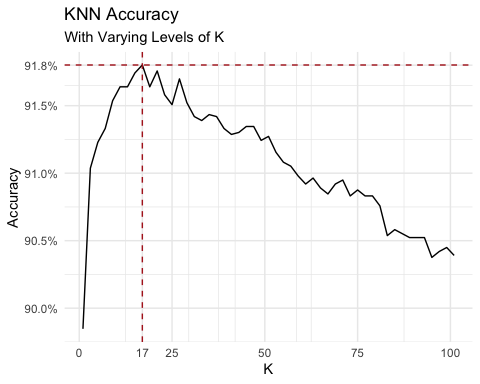
Description automatically generated

After retraining the decision tree, we have a new accuracy of 86.8% - we traded a 7-dimension reduction for less than a 1% penalty to accuracy than before. The most important variables have changed to Eccentricity and Shape Factor 2, which has been visualized in the Visualization section.

Now, we retrain the KNN model.

## KNN with Correlated Variables Removed

Let’s re-evaluate the optimal value for K.



The best K is 17, which yields an accuracy of 91.8%, which again is a less than 1% reduction in accuracy than before.

# Conclusion

We have constructed and compared several machine learning models to classify beans. We found that for this data set, K Nearest Neighbors was the optimal algorithm - it classified the bean type with an accuracy of 92%. We have also shown that for an opportunity cost of less than 1% accuracy for both modelling methods, we could reduce the number of variables in each model by almost 50%, greatly reducing model complexity.