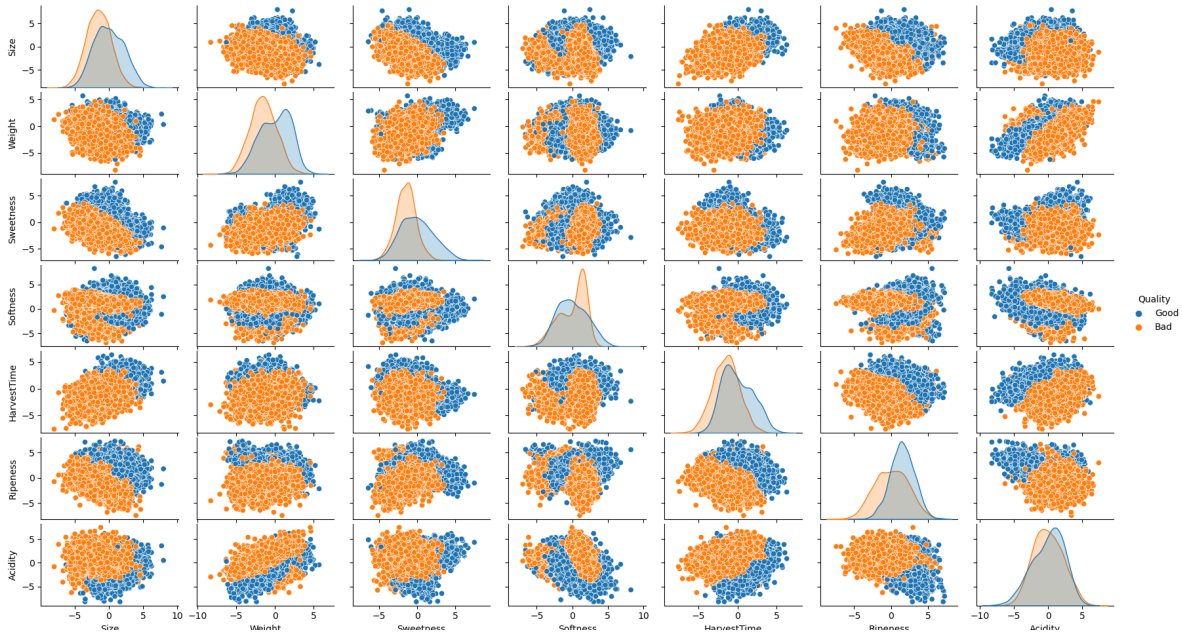
Dataset

The quality of a banana is determined by several factors, including things such as taste, ripeness, and overall health. Depending on how these factors measure together help determine whether or not a piece of fruit is of good quality or not.

The dataset I'll be using for this assignment is the 🍌 | Banana Quality dataset from Kaggle. This dataset includes a variety of qualitative features ranked as quantitative data for determining whether a banana is of good quality or not.

The dataset includes the following features: Size, Weight, Sweetness, Softness, Harvest Time (amount of time passed from harvesting of the fruit), Ripeness, Acidity, and Quality. The dataset includes 8000 records, with a 4006/3994 split between Good and Bad bananas. Here is a set of plots comparing each feature against each other.



From these plots, we can see that there is little correlation between these plots, and that each one has a decent statistical distribution for both good and bad bananas. As such, this report will test all features in our classification methods. We will determine the importance of each feature later.

For the sake of training and testing, data will be split into 80% training, 20% testing.

Classification Methods

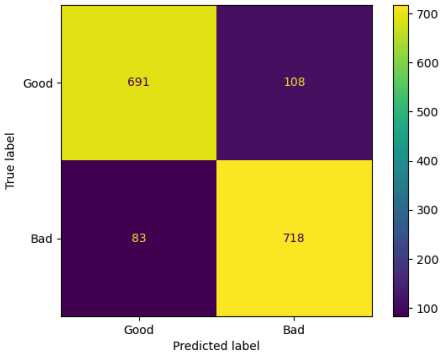
This report will perform a classification analysis using the five following methods:

a) Logistic Regression
b) Nearest Neighbor

c) Naive Bayes
d) Decision Tree
e) Random Forest

1) Logistic Regression

Logistic Regression performs a regression using our features to determine the probability of the record's class. Running the code, we get the following confusion matrix and classification report:




```
Accuracy: 0.880625
              precision    recall  f1-score   support

        Good       0.89      0.86      0.88       799
         Bad       0.87      0.90      0.88       801

    accuracy                           0.88      1600
```

To show an example, we take the first 10 predicted values in our test set (60% accuracy), the probabilities calculated for each (Left is for Bad, Right is for Good), and the actual values. As we can see, the probabilities in the predicted set add up to 1 for each record, and the predicted value matches the higher probability.

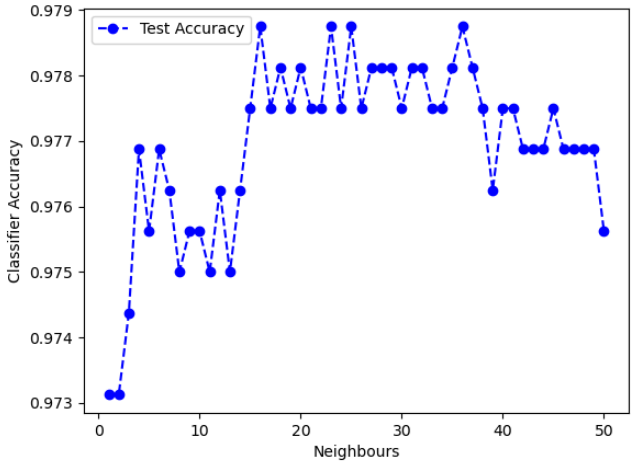
```
Good
Good
 Bad
 Bad
Good
Good
Good
 Bad
 Bad
Good
```

```
[[0.24134575 0.75865425]
 [0.23566649 0.76433351]
 [0.51566125 0.48433875]
 [0.99086444 0.00913556]
 [0.48915085 0.51084915]
 [0.36990141 0.63009859]
 [0.47296062 0.52703938]
 [0.74078953 0.25921047]
 [0.95881802 0.04118198]
 [0.03703098 0.96296902]]
```

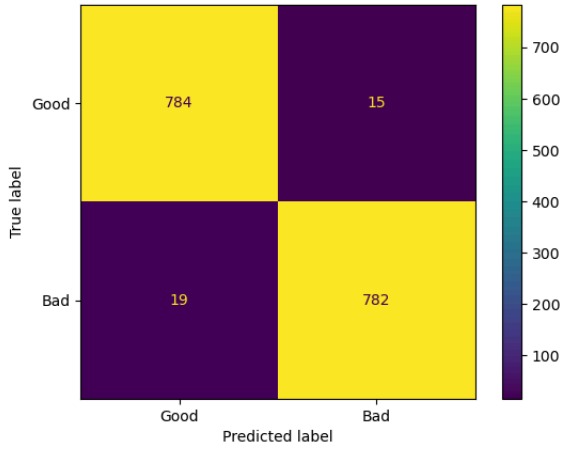
```
2828    Bad
6752    Bad
5683   Good
7706    Bad
614    Good
2711    Bad
1896   Good
2521    Bad
6737    Bad
4250   Good
```

2) Nearest Neighbor

K-Nearest Neighbors determines the class of a record by comparing the euclidean distance of a record to all the records it has been trained on, and picking the most frequent class.
Before predicting, we test the number of neighbors to find the best number that will properly classify any record without being affected by noise or other groups. From testing, it's determined to be 16:



Running our code, we get the following confusion matrix and classification report:



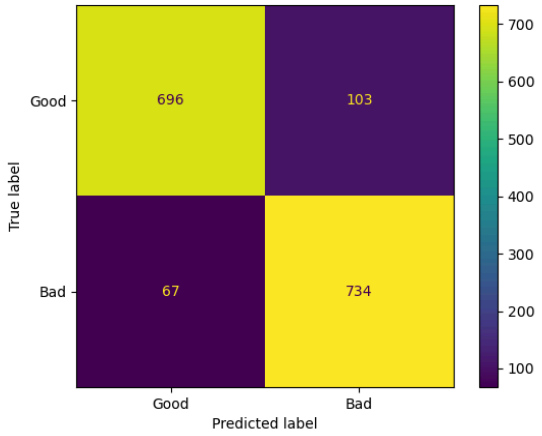

```
Accuracy: 0.97875
              precision    recall  f1-score   support

        Good       0.98      0.98      0.98       799
         Bad       0.98      0.98      0.98       801

    accuracy                           0.98      1600
```

To show an example, we take the first 10 predicted values in our test set (100% accuracy), the probabilities calculated for each (Left is for Bad, Right is for Good), and the actual values. Here, we can see for this set, the closest neighbors are pretty uniform, with maybe just 1 neighbor of a different class.

```
Bad      [[0.9375 0.0625]  2828   Bad
Bad       [1.     0.    ]  6752   Bad
Good      [0.     1.    ]  5683   Good
Bad       [1.     0.    ]  7706   Bad
Good      [0.     1.    ]  614    Good
Bad       [1.     0.    ]  2711   Bad
Good      [0.     1.    ]  1896   Good
Bad       [1.     0.    ]  2521   Bad
Bad       [0.9375 0.0625]  6737   Bad
Good      [0.     1.    ]] 4250   Good
```

3) Naive Bayes

Naive Bayes uses the probability of each class occurring for a specific feature value, and then using those probabilities to calculate the chance for, considering all features, if a record belongs to one class or another.
For this classifier, we use the Gaussian Naive Bayes model as all the features are in a discrete numeric series. Running our code, we get the following confusion matrix and classification report:



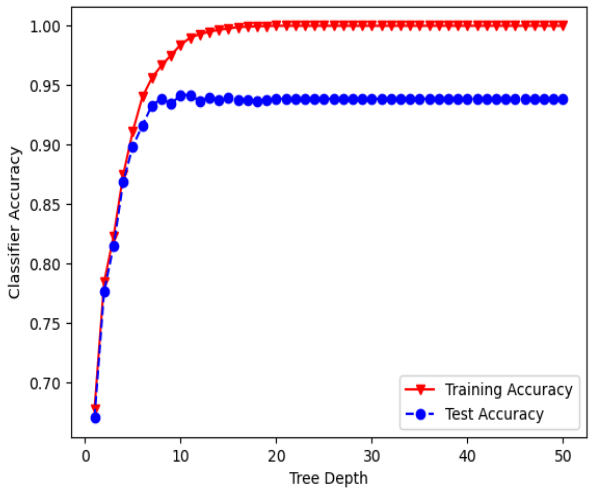

```
Accuracy: 0.89375
              precision    recall  f1-score   support

        Good       0.91      0.87      0.89       799
         Bad       0.88      0.92      0.90       801

    accuracy                           0.89      1600
```

To show an example, we take the first 10 predicted values in our test set (80% accuracy), the probabilities calculated for each (Left is for Bad, Right is for Good), and the actual values.

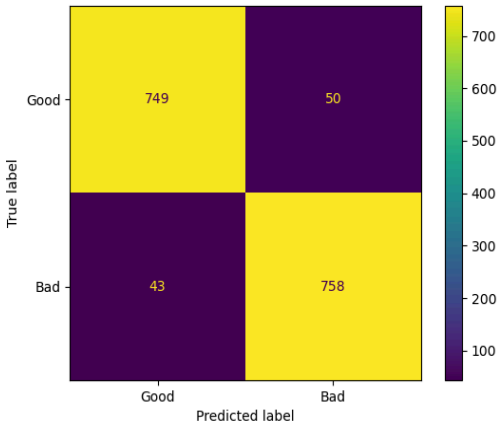
Good	[0.49940477 0.50059523]	2828	Bad
Good	[0.34811862 0.65188138]	6752	Bad
Good	[0.48987326 0.51012674]	5683	Good
Bad	[0.98386709 0.01613291]	7706	Bad
Good	[0.088617 0.911383]	614	Good
Bad	[0.58009007 0.41990993]	2711	Bad
Good	[0.44021086 0.55978914]	1896	Good
Bad	[0.71249617 0.28750383]	2521	Bad
Bad	[0.98410614 0.01589386]	6737	Bad
Good	[0.13785623 0.86214377]]	4250	Good

4) Decision Tree

A decision tree determines the class by creating a traversable tree. Each split is created by testing a specific feature and determining how it can be split to best split the classes and minimize entropy in the prediction.
Before actually predicting, we test the various depths of each tree to find the best depth without overfitting the data. From testing, it's determined to be a depth of 10:

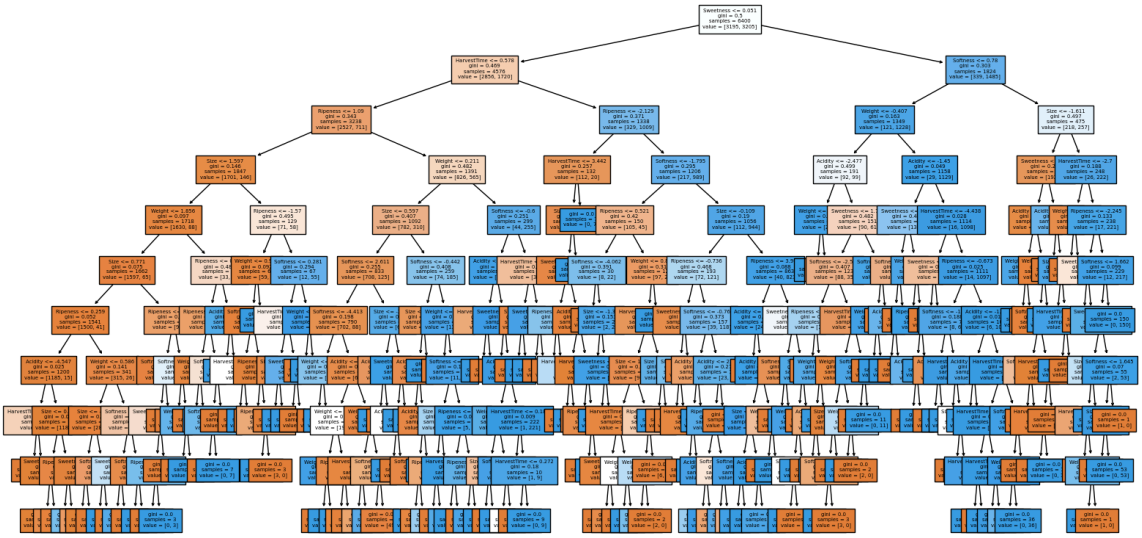


Running our code, we get the following confusion matrix and classification report:



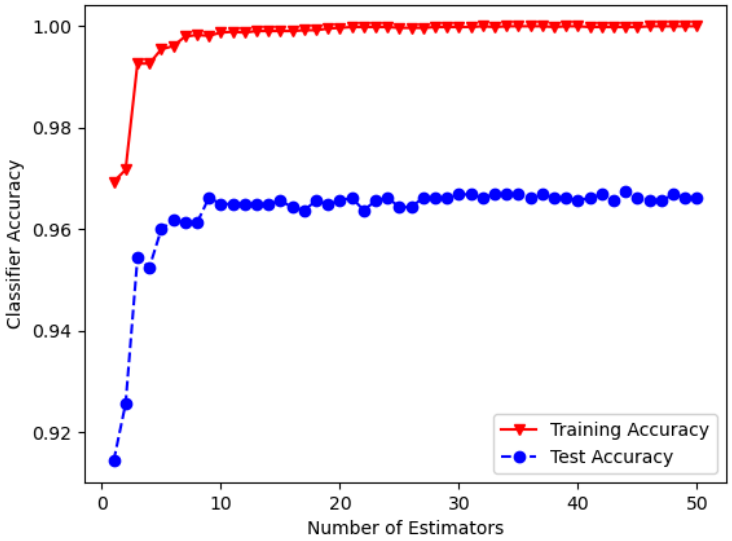

```
Accuracy: 0.941875
              precision    recall  f1-score   support

        Good       0.95      0.94      0.94       799
         Bad       0.94      0.95      0.94       801

    accuracy                           0.94      1600
```

The tree generated (Due to the size of the tree, the lower splits are unreadable. Image is here as an example):

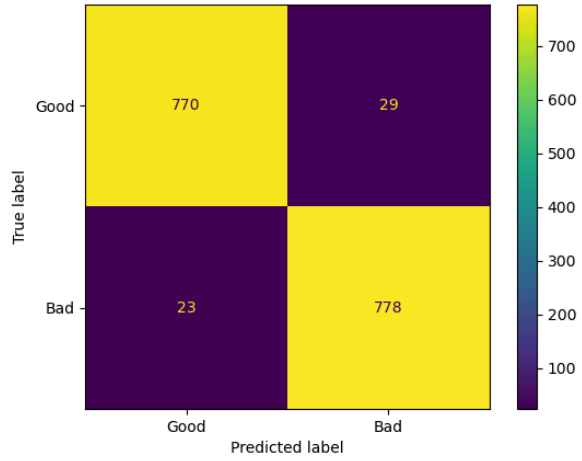


5) Random Forest

Random Forest is similar to Decision Tree, where we construct a traversable tree to determine the class. The difference here is that a Random Forest constructs multiple trees, then votes on the most frequent class predicted between all of them.
Before actually predicting, we test the various number of estimators to find the best accuracy. From testing, it's determined to be 44 estimators:



Running our code, we get the following confusion matrix and classification report:




```
Accuracy: 0.9675
              precision    recall  f1-score   support

        Good       0.97      0.96      0.97       799
         Bad       0.96      0.97      0.97       801

    accuracy                           0.97      1600
```

I won't show the estimators themselves as there are 44 of them, too many to fit in this report. For the example, we take the first 10 predicted values in our test set (100% accuracy), the probabilities calculated for each (Left is for Bad, Right is for Good), and the actual values.

```
Bad
Bad
Good
Bad
Good
Bad
Good
Bad
Good
Good
```

```
[[0.97727273 0.02272727]
 [0.93181818 0.06818182]
 [0.04545455 0.95454545]
 [0.97727273 0.02272727]
 [0.06818182 0.93181818]
 [0.88636364 0.11363636]
 [0.04545455 0.95454545]
 [0.86363636 0.13636364]
 [0.38636364 0.61363636]
 [0.04545455 0.95454545]]
```

```
2828    Bad
6752    Bad
5683   Good
7706    Bad
614    Good
2711    Bad
1896   Good
2521    Bad
6737    Bad
4250   Good
```

Results

With all five classifiers tested and measured, we can compare their accuracies.

Ranking from best Accuracy and F1-Score to worst

Classifier	K-Nearest Neighbors	Random Forest	Decision Tree	Naive Bayes	Logistic Regression
Accuracy	0.97875	0.9675	0.941875	0.89375	0.880625
F1-Score	0.98	0.97	0.94	0.89 (Good)/ 0.90 (Bad)	0.88

With this, we can see that the best model for predicting classes in this dataset is K-Nearest Neighbors, followed very closely by Random Forest at a very high ~98% and ~97% respectively.

Decision Tree is next at ~94%. Naive Bayes and Logistic Regression perform the worst, and despite that they still perform relatively well at ~89% and ~88% respectively. Their F1-scores are relatively similar in value and position (as expected from a dataset with equal classes). From this, we can draw some conclusions based on the accuracy and the examples provided.

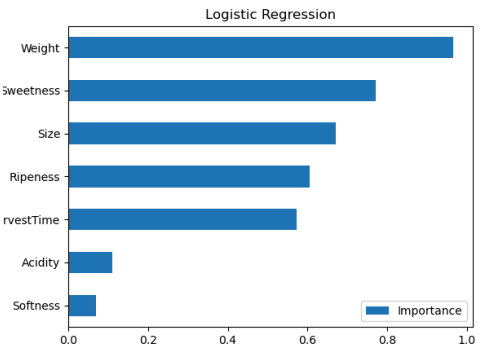
Nearest Neighbor has the highest accuracy of all the classifiers. From the example, we can see that there's very little contention for which class they belong to, suggesting that there's little noise, outliers, and that Good and Bad bananas have very different features to them.

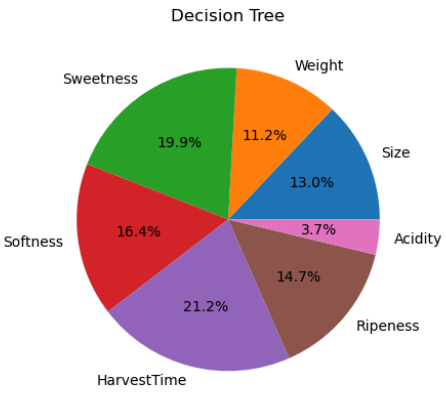
Decision Tree has a high accuracy thanks to the depth and the number of features available to check for. The incorrect predictions may come from either slight overfitting of some of the data or some bananas needing more splits to properly determine the class. Random Forest has access to a large number of estimators that help reduce the amount of overfitting, likely helping it get a higher accuracy score.

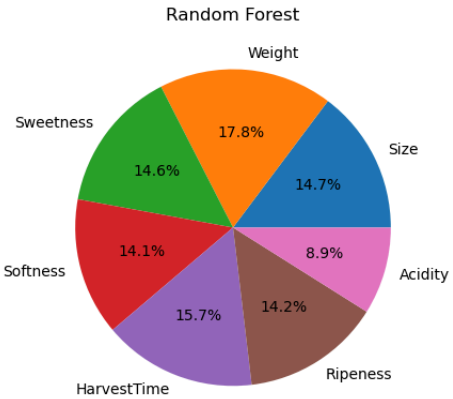
Naive Bayes and Logistic Regression perform the worst at roughly similar values. They still perform extremely well at near 90% accuracy. Their lower accuracy may be in part to the distribution of our features. Checking the plots of the features shows several distributions, mostly related to softness, where Bad bananas are split into multiple clusters. Without accounting and adjusting for these separate clusters, these models may perform worse than the other models. Based on their confusion matrices, they're slightly worse at predicting True Positives and False Negatives than False Positives and True Negatives.

Finally, let's determine the importance of each feature in regards to these models. As Nearest Neighbor uses euclidean distance and Naive Bayes considers the probabilities of every feature, all features are equally important for each.

Below are three graphs (2 pie and 1 bar) that show the importance of each feature to their respective models.







From this, we can see that the models share some similarities and have some differences. For all three plots, Acidity was considered one of the least important between the three. Vice versa,

Sweetness was considered significantly between them all (though not the highest for Random Forest), as well as Ripeness, Harvest Time, and Size. Decision Tree and Random Forest consider softness important, while Logistic Regression considers it the least important.

From this, we can consider in future tests of this dataset to exclude Acidity in these models to help with performance. Besides that, all other features look to be relatively important for determining the class of a banana.

The quality of a banana can vary wildly, and determining whether one is good to eat or not may be tricky at a glance. However, from these tests and our results, we can safely assume that there are a number of variables and features we can use to help us determine whether or not a banana is of good quality or not.

References

l3LIFF. 🍌 | Banana Quality. Retrieved March 4, 2024 from https://www.kaggle.com/datasets/l3llff/banana/data.