Overview

This study applies the Naïve Bayes Classifier (NBC) to classify oranges and grapefruits based on their physical attributes. NBC is a generative classifier among popular supervised machine learning algorithms. In NBC, all the features of the input data are assumed to be independent of each other. NBC calculates posterior probability using the prior probability of each class and the likelihood of the features and applies the same for the classification.

Dataset

The dataset for this study is downloaded from Kaggle's website, which can be found by following this link. It has five features (aka independent variables), which are diameter, weight, and three colors, i.e., red, blue, and green. All features are continuous variables representing real numbers. The dependent variable (aka label) is the type of fruit. Since this is a binary classification problem, there are two classes in the label i.e., grapefruit and orange. There are 10,000 data instances in the dataset, which have been used for training and testing the model.

Analytics

As mentioned, for training and testing the model, a naïve bayes classifier is used, utilizing Scikit-Learn built-in libraries in Python. Other libraries such as Pandas and NumPy, were also used during the analysis. The dataset is divided into training and testing sets that contain 80% and 20% of the data, respectively. To evaluate the model's performance in each set, metrics such as accuracy, sensitivity, specificity, f1 score, and log loss were employed using scikit-learn libraries. In addition, ROC curves are plotted to comprehensively evaluate its performance.

Results

After executing the Python code, the following results were obtained.

Metrics	Training Accuracy	Test Accuracy
Accuracy	92.15%	92.00%
Sensitivity	92.23%	91.01%
Specificity	92.07%	93.02%
F1 Score	92.13%	92.01%
Log Loss	23.23%	22.72%
Area Under the ROC Curve	97.71%	97.85%

Accuracy

The accuracy of the model is calculated as the ratio of correctly classified instances to the total number of instances. Which is also known as classification rate and calculated using the below formula.

$$\frac{True\ Negatives + True\ Positives}{Total\ Number\ of\ Instances}$$

Since both the training and testing classification accuracy are higher (i.e., 92.15% and 92.00%, respectively), there are fewer classification errors.

Sensitivity

Sensitivity measures the ratio of correctly classified positive instances to the total number of actual positive instances. It is also known as recall or true positive rate and is calculated using the below formula.

$$\frac{True\ Positives}{True\ Positives+False\ Negatives}$$

A higher sensitivity implies fewer wrong predictions in the positive class; hence, the model did well in both the training and test data, with scores of 92.23% and 91.01%, respectively.

Specificity

Similarly, just like sensitivity for positive instances, specificity measures the ratio of correctly classified negative instances to the total actual negative instances. It is also known as selectivity or true negative rate and is calculated using the below formula.

$$\frac{True\ Negatives}{True\ Negatives+False\ Positives}$$

Since false positives affect this ratio, lower false positives result in higher specificity. Higher specificity reflects less error in the prediction in the negative class, and our model has 92.07% and 93.02% accuracy for train and test data, which can be considered good performance of the model.

F1 Score

The F1 score is the harmonic mean of precision (calculated by dividing true positive results by all positive results) and recall (calculated by dividing true positive results by actual positives), which is calculated using the below formula.

$$2\frac{Precision*Recall}{Precision+Recall}$$

A higher F1 score indicates a good balance between precision and recall. In our case, we have 92.13% in training data and 92.01% in testing data, which shows the good performance of the model.

Logarithmic Loss

Logarithmic loss (aka log-loss or cross-entropy loss) quantifies the difference between predicted probabilities and actual values to measure the performance of the model. Log-loss is calculated using the following formula.

$$Log\ loss = \frac{1}{N} \sum_{i=1}^{N} [y_i * log(P((y_i|x_i)) + (1 - y_i) * log(1 - p(y_i|x_i))]$$

A Lower log-loss represents better model performance, reflecting less uncertainty. In our case we have 23.23% & 22.72% log loss in training and testing dataset respectively which is relatively low.

ROC Curve

The Receiving Operating Characteristic (ROC) curve assesses the performance of our model across various classification thresholds. The ROC plot visualizes the model's performance at different trade-offs between true positives and false positives for varying thresholds. The ROC curves for both the training and testing datasets were visualized in the below plot.

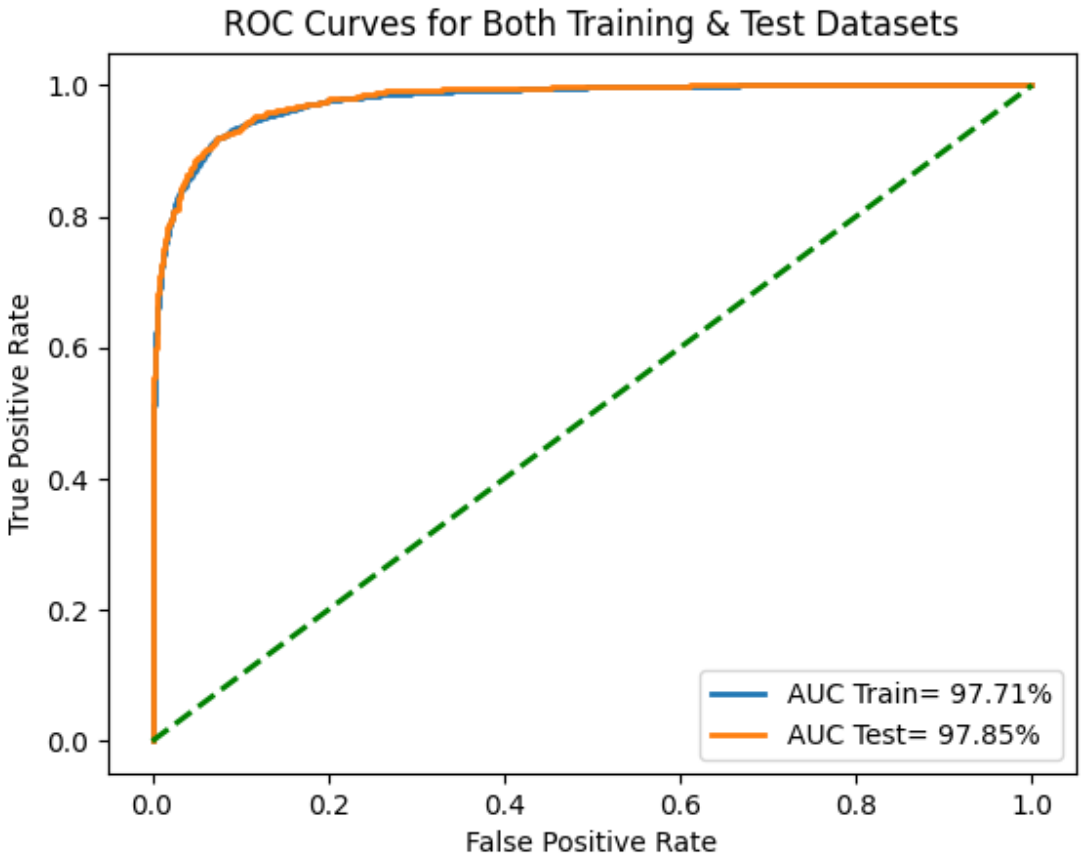

Figure 1. Receiving Operating Characteristic (ROC) Curve

A classifier with a curve closer to the top-left corner of the plot is generally considered better because it achieves higher true positives with lower false positives, indicating better classification. Both train and test data have a similar ROC curve, which is closer to the top-left corner, and more than 97% of the area is under the curve (AUC). This result suggests that the model is performing well.