Indian Institute of Technology Kharagpur
Machine Learning (CS60050)
Spring 2022-23
Project 1: Classification of Rice Varieties using Gaussian Naive Bayes Learning Model (RVNB)

Group 31
Avi Amalanshu (20EC30063)
Anamitra Mukhopadhyay (20CS30064)
Chavle Abhishek Shivanand (22CS60R79)
Guided by Prof. Aritra Hazra and TA Rijoy Mukherjee

February 11, 2023

1 Abstract

The purpose of this report is to present the results of a project that aimed to classify rice into two distinct categories, "Cammeo" and "Osmanick," based on specific attributes using a Gaussian Naive Bayes Classifier. The dataset used for this project contained various characteristics of the rice samples, such as grain length, width, and aspect ratio, as well as color features like hue, saturation, and intensity. A Gaussian Naive Bayes Classifier was trained from scratch on this data. Its performance was evaluated by calculating metrics such as accuracy, precision, recall, and F1-score. The results showed that the classifier was able to achieve an accuracy of 92.7%, demonstrating its effectiveness in classifying rice into the two categories.

2 Classification in Machine Learning

Classification is a fundamental problem in machine learning that involves assigning a label or class to a given input data point, which is a set of attributes.

2.1 Naive Bayes Classifier

Naive Bayes classifiers are a family of probabilistic classification algorithms. The central idea is to model the relationship between the input data and the target class based on Bayes' theorem, which states that the probability of a class given an input feature vector can be estimated based on the prior probability of the class and the likelihood of the features given the class.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Say we have the attributes $X_1, X_2, \ldots, X_n$ and the class Y.

$$P(Y = Y_n|X_1, X_2, \ldots, X_n) = \frac{P(X_1, X_2 \ldots X_n|Y = Y_n)P(Y = Y_n)}{P(X_1, X_2, \ldots, X_n)} \tag{1}$$

Say Y is a binary class (positive examples $Y = 1$ and negative examples $Y = 0$). Dividing, we get

$$\frac{P(Y = 1|X_1, X_2 \ldots, X_n)}{P(Y = 0|X_1, X_2 \ldots, X_n)} = \frac{P(Y = 1)P(X_1, X_2, \ldots, X_n|Y = 1)}{P(Y = 0)P(X_1, X_2 \ldots, X_n|Y = 0)} \tag{2}$$

Naive Bayes classifiers make the "naive" assumption that the features are conditionally independent given the class, which enables them to model complex relationships between the input and the target class with a relatively small number of parameters. So, we have

$$\frac{P(Y = 1|X_1, X_2 \ldots, X_n)}{P(Y = 0|X_1, X_2 \ldots, X_n)} = \frac{P(Y = 1) \cdot \prod_{i=1}^{n} P(X_i|Y = 1)}{P(Y = 0) \cdot \prod_{i=1}^{n} P(X_i|Y = 0)} \tag{3}$$

2.2 Gaussian Naive Bayes Classifier

The Gaussian Naive Bayes classifier is a specific variant of the Naive Bayes classifier for continuous data. It assumes that the features are normally distributed. Using the logarithm of the RHS of equation (3) we have

$$\log\left(\frac{P(Y = 1)}{P(Y = 0)}\right) + \sum_{i=1}^{n} \log\left(\frac{P(X_i|Y = 1)}{P(X_i|Y = 0)}\right) \tag{1}$$

Since we are modelling $P(X_i = x_{ij}|Y = 1) = \frac{1}{\sqrt{2\pi}\sigma_{1i}} \exp\left(\frac{-\frac{1}{2}(x_{ij}-\mu_{1i})}{\sigma_{1i}}\right)^2$

$$\log\left(\frac{P(Y = 1)}{P(Y = 0)}\right) + \sum_{i=1}^{n} \log\left(\frac{\sigma_{0i}}{\sigma_{1i}}\right) + \frac{1}{2}\sum_{i=1}^{n}\left[\frac{(x_{ij} - \mu_{0i})^2}{\sigma_{0i}^2} - \frac{(x_{ij} - \mu_{1i})^2}{\sigma_{1i}^2}\right] \tag{2}$$

By exposing the model to data, we statistically approximate the values of μ_{0i}, μ_{1i}, σ_{0i} and σ_{1i}.

3 Methodology

We wrote Python code that implements a Gaussian Naive Bayes (GNB) binary classifier model in Python. The model has two classes, so it can only be used for binary classification. The model is implemented as a class `GNB_binary` and includes the following methods:

1. `__init__`: Initializes the class and sets the required attributes like the number of attributes, means, variances and constants for each class and the class names.
2. `fit`: The `fit` function in this code implements the Gaussian Naive Bayes (GNB) algorithm for binary classification.
3. `predict_single`: Given an input sample of attributes, it predicts the class based on the mean and variance values.
4. `predict`: Given a test dataframe of samples, it returns the predictions of each sample by calling the `predict_single` method.

4 Evaluation Metrics

4.1 Accuracy

Accuracy is defined as the number of correct predictions divided by the total number of predictions. It is used as a measure of how well a classifier performs in general. The formula for accuracy is:

$$\text{Accuracy} = \frac{(\text{True Positives} + \text{True Negatives})}{\text{Total Observations}}$$

4.2 Precision

Precision is a measure of the exactness or quality of the predictions made by a classifier. It is defined as the number of true positive predictions divided by the sum of true positive and false positive predictions. The formula for precision is:

$$\text{Precision} = \frac{\text{True Positives}}{(\text{True Positives} + \text{False Positives})}$$

4.3 Recall

Recall is a measure of the completeness of the predictions made by a classifier. It is defined as the number of true positive predictions divided by the sum of true positive and false negative predictions. The formula for recall is:

$$\text{Recall} = \frac{\text{True Positives}}{(\text{True Positives} + \text{False Negatives})}$$

4.4 F1-Score

The F1-Score is the harmonic mean of precision and recall and is used as a single metric for evaluating the performance of a classifier. The formula for F1-Score is:

$$\text{F1-Score} = 2 \times \text{Precision} \times \frac{\text{Recall}}{(\text{Precision} + \text{Recall})}$$

4.5 Support

Support is the number of observations of each class in the test set. It is used to give a better understanding of the distribution of the target class in the test set.

5 Results

5.1 5-fold Cross Validation

	Cammeo	**Osmancik**	**Macro Avg**	**Weighted Avg**
precision	0.920398	0.912913	0.916655	0.915913
recall	0.864486	0.950000	0.907243	0.915730
f1-score	0.891566	0.931087	0.911327	0.915249
support	214.000000	320.000000	534.000000	534.000000

Table 1: Classification Report for Fold #1. **Accuracy**: 0.9157303370786517

	Cammeo	**Osmancik**	**Macro Avg**	**Weighted Avg**
precision	0.919283	0.929260	0.924271	0.925019
recall	0.903084	0.941368	0.922226	0.925094
f1-score	0.911111	0.935275	0.923193	0.925003
support	227.000000	307.000000	534.000000	534.000000

Table 2: Classification Report for Fold #2. **Accuracy**: 0.9250936329588015

	Cammeo	**Osmancik**	**Macro Avg**	**Weighted Avg**
precision	0.892241	0.933555	0.912898	0.915960
recall	0.911894	0.918301	0.915097	0.915572
f1-score	0.901961	0.925865	0.913913	0.915684
support	227.000000	306.000000	533.000000	533.000000

Table 3: Classification Report for Fold #3. **Accuracy**: 0.9155722326454033

	Cammeo	**Osmancik**	**Macro Avg**	**Weighted Avg**
precision	0.899563	0.914474	0.907019	0.907984
recall	0.887931	0.923588	0.905760	0.908068
f1-score	0.893709	0.919008	0.906359	0.907996
support	232.000000	301.000000	533.000000	533.000000

Table 4: Classification Report for Fold #4. **Accuracy**: 0.9080675422138836

	Cammeo	**Osmancik**	**Macro Avg**	**Weighted Avg**
precision	0.890995	0.925466	0.908231	0.911755
recall	0.886792	0.928349	0.907571	0.911820
f1-score	0.888889	0.926905	0.907897	0.911784
support	212.000000	321.000000	533.000000	533.000000

Table 5: Classification Report for Fold #5. **Accuracy**: 0.9118198874296435

	Cammeo	**Osmancik**	**Macro Avg**	**Weighted Avg**
precision	0.904496 (**0.920160**)	**0.923134** (0.911215)	0.913815 (**0.915687**)	**0.915326** (0.915269)
recall	**0.890837** (0.889961)	0.932321 (**0.936000**)	0.911579 (**0.912981**)	0.915257 (**0.915136**)
f1-score	0.897447 (**0.904809**)	**0.927628** (0.923441)	0.912538 (**0.914125**)	**0.915143** (0.914997)
support	222.400000 (518.000000)	311.000000 (625.000000)	533.400000 (1143.000000)	533.400000 (1143.000000)

Table 6: average of the 5-fold classification reports. Values in brackets are those produced by `sklearn`'s inbuilt classifier. **Accuracy**: **0.9152567264652767** (0.9151356080489939)

	Cammeo	**Osmancik**	**Macro Avg**	**Weighted Avg**
precision	**0.932406** (0.920160)	**0.923438** (0.911215)	**0.927922** (0.915687)	**0.927502** (0.915269)
recall	**0.905405** (0.889961)	**0.932321** (0.945600)	**0.925503** (0.912981)	**0.927384** (0.915136)
f1-score	**0.918707** (0.904809)	**0.934387** (0.923441)	**0.926547** (0.914125)	**0.927281** (0.914997)
support	518.000000 (518.000000)	625.000000 (625.000000)	1143.000000 (1143.000000)	1143.000000 (1143.000000)

Table 7: Results when trained on the full 70% train set and tested on the 30% train set. Values in brackets are those produced by `sklearn`'s inbuilt classifier. **Accuracy**: **0.9273840769903762** (0.9151356080489939)

6 Conclusion

The project aimed to train a Gaussian Naive Bayes classifier from scratch and compare its performance with the inbuilt Gaussian Naive Bayes classifier provided by the scikit-learn library. The model was implemented using the `GNB_binary` class, which has functions for training the model on a given dataset, predicting the class of a single instance, and making predictions for a whole test set. The performance of the model was evaluated using the accuracy, precision, recall, F1-score, and support metrics. These metrics were calculated using the inbuilt classification_report function from scikit-learn.

The model was trained and tested using 5-fold cross-validation, which is a technique that divides the data into 5 folds, uses 4 folds for training and 1 fold for testing, repeats the process 5 times with each fold serving as the test set once, and calculates the average performance of the model over all 5 runs. The results of the model showed that the accuracy, precision, recall, F1-score, and support values calculated from the model were consistent with the values obtained from the inbuilt scikit-learn Gaussian Naive Bayes classifier.

We found the cross-validation experiment to show similar performance to the sklearn model. In fact, we achieved marginally better accuracy (91.53% to 91.51%). When we trained the model on the entire training data split at once, we achieved even better results (an accuracy of 92.74%). This also goes to show that (at least upto a certain point) exposing the model to more data leads to better generalization over unseen data.

In conclusion, the project successfully trained a Gaussian Naive Bayes classifier from scratch and demonstrated that its performance is comparable to the performance of the inbuilt classifier provided by the scikit-learn library.

A note on support values for the cross-validation experiment: In cross-validation, the data is split into multiple folds, usually into training and validation sets. For each fold, the model is trained on the training data and evaluated on the validation data. The overall performance is then reported as the average of the performance of the model on each fold.

Support is the number of samples of the true response values in a classification problem. When the data is split into folds for cross-validation, each fold only includes a portion of the total data, meaning that the support will also be halved. The support for each fold will be the number of samples of the true response values in that particular fold, not the total number of samples in the complete dataset. This is why the support is halved in cross-validation, as each fold is only a portion of the total data.