Design of Naïve-Bayes Classifier for Class Prediction

Project Report for EAS 595: Introduction to Probability Theory

Anand
Graduate Student- Data Sciences
University at Buffalo
Buffalo, NY, US
Anand6@buffalo.edu

Rahul Patil
Graduate Student- Data Sciences
University at Buffalo
Buffalo, NY, US
rpatil6@buffalo.edu

Abstract

Data mining is widely used in a variety of applications such as medical diagnosis, targeted marketing, estimating the shares of television audiences, financial forecasting, product design, fraud detection, automated abstraction, etc. With such widespread use, classification becomes an important data mining technique with a wide range of applications to classify the various types of data existing in almost all areas of our life. In this project, we design a Naïve-Bayes classifier to predict the class of input data points. The project deals with data manipulation in order to measure the accuracy and error of the classifier. And finally, design of a univariate and multivariate classifier and comparison of accuracy of the two.

I. Introduction

For a given data set of 10,000 input data points from 1000 participants in 5 different tasks in two different measurements, a Naïve-Bayes classifier is implemented to compute the probability of each class and the best-fit class for each data point. Naïve Bayes is a simple learning algorithm that utilizes Bayes' rule together with a strong assumption that the attributes are conditionally independent, given the class. Coupled with its computational efficiency and many other desirable features such as low variance, incremental learning, direct prediction of posterior probabilities and robustness, Naïve Bayes is widely applied in practice. Bayes' theorem is a mathematical formula used to determine conditional probability, which is named after 18th century British mathematician Thomas Bayes given by:

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

where, P(A|B) is the probability of the occurrence of event A when event B occurs,
P(A) is the probability of the occurrence of A,
P(B|A) is the probability of the occurrence of event B when event A occurs,
P(B) is the probability of the occurrence of B

In this project, given a data point (x) and a class (C), the classifier predicts probability of each class such as the probability of a given data point belongs to a particular class. The class with highest probability is considered as most likely class.

II. Test Cases and Expected Results

Given two sets of data of 1000 participants over 5 different tasks, recorded in two different measurements, we build two classifiers, a univariate and a multivariate classifier to test for the following cases:

Case I. X = F1(First Measurement) scores
Case II. X = Z1(Normalized scores of First Measurement)
Case III. X = F2(Second Measurement) scores
Case IV. X = $\begin{bmatrix} Z1 \\ F2 \end{bmatrix}$ multivariate normal distribution

Using the first 100 subjects as training data, and the rest 900 as test data, we are calculating the accuracy of the classifier, given by:

$$Classification\ accuracy = \frac{Correct\ Predictions}{Total\ no.of\ Predictions}$$

And

$$\text{Error rate} = \frac{Incorrect\ Predictions}{Total\ no.of\ Predictions}$$

We expect to see a trend of different accuracy for each of the above test cases with a variation in the error rate. The outcome of the project is to predict which classifier is more accurate and why does the accuracy increase.

III. Class Probability and Normalization

Having an equiprobability of 1/5 for each class, using Bayes' theorem the conditional probability for each class is given as

$$P(C_k|x) = \frac{p(C_k) * p(x|C_k)}{p(x)}$$

Here, we ignore the denominator as it does not depend on C. Using the above equation, we can determine the class probability of each participant of a univariate distribution as,

$$P(C_k) = P(C_k) = P(C_k) * P(F_1|C_k),$$

And for a multivariate distribution as,

$P(C_k) = P(C_k) = P(C_k)*P(F_1|C_k) *P(F_2|C_k)*\dots$

which under the independence assumption is:

$$p(C_k| x_{1\dots.}x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

where $Z = p(x) = \sum p(C_k)*p(x|C_k)$ is a scaling factor dependent only on $x_1 \dots. x_n$, that is a constant if the value of features is known.

If the data is unbalanced then the classifier would give more weight-age that set of group which has higher weights, to overcome such issues, we normalize the data by calculating their Z-scores. To calculate the Z-Scores, we use the formula:

$$Z = \frac{x-\mu}{\sigma}$$

Where, x = datapoint, μ = mean of each subject and σ = standard deviation of the subject.

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal distribution (Also known as Gaussian distribution). Suppose training data contains a continuous attribute x, then we first segment the data by class, and then compute the mean and variance of x in each class. If μ_k is the mean of values in x associated with class C_k and σ_k^2 is the variance, collected under observation value v, then the probability distribution of v given a class C_k, $p(x=v|C_k)$ is given by a normalized equation:

$$p(x=v \mid C_k) = \frac{1}{\sqrt[2]{2*sqr(\sigma_k)}} e^{-\frac{-(v-\mu_k)^2}{(2\sigma_k)^2}}$$

IV. Code Flow and Implementation

The implemented python code has the following flow:

1. Read data from the MATLAB file using scipy
2. Create data frames to tabulate the data from the file using pandas
3. Define a function to calculate the mean and variance
4. Define a function to calculate the Z-Score of the given subjects
5. Define a function to calculate the probability distribution of subjects
6. Define a function to predict the class of univariate data
7. Define a function to predict the class of bi-variate data
8. Define a function to calculate accuracy and error of the classifier for each of the given condition.
9. Define a function to plot the classified data points into a graph, and a bar plot of accuracy vs error for each given test case
10. Define a main function for all the above functions, output of which would be the plots of classified data, accuracy and error of all the test cases.

V. Graphs and Inferences

The accuracy and error of the Naïve-Bayes classifier for each of the cases is as follows:

Case	Accuracy (%)	Error (%)
1. X = F1	53	47
2. X = Z1	88.33	11.67
3. X = F2	55.16	44.84
4. $X = [^{Z1}_{F2}]$	97.98	2.02

With the plots as follows:

Fig. Classifier Output for X = F1

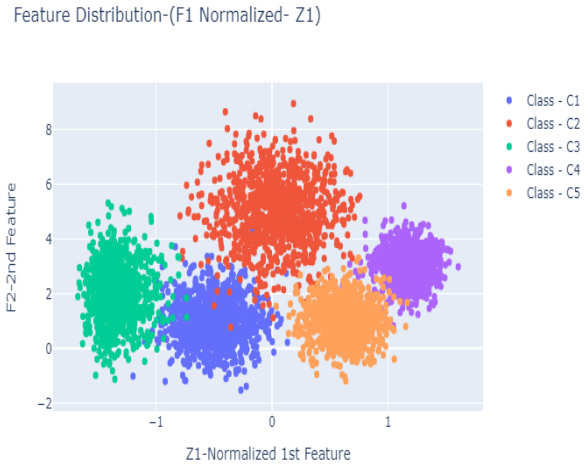
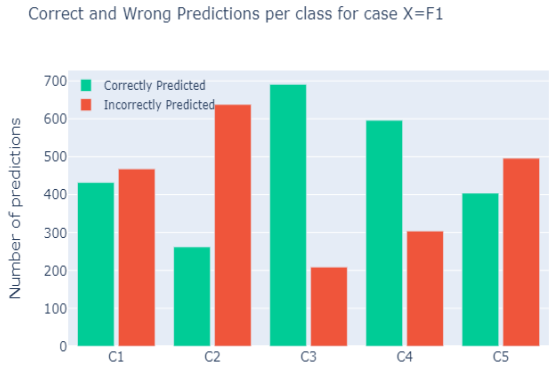


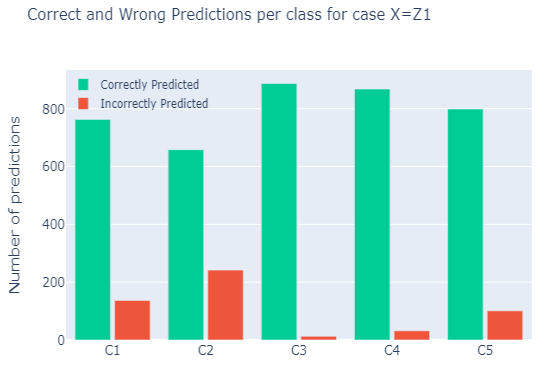
Fig. Classifier Output for X = Z1

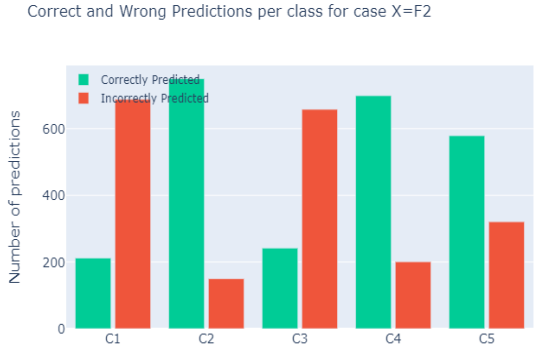
From the above two figures, we can infer that, for normalized data, there is a distinct classification of subjects as compared to input of the subjects to the classifier as was provided. This clear distinction lies with the fact that the normalized data are centered over the mean 0 with standard deviation 1. Therefore, negating much of the overlap of classes, thereby resulting in better classification of data.

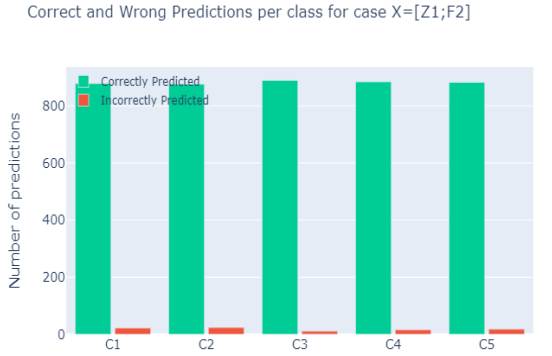
The plot of correct vs. wrong predictions and accuracy vs. error for the given cases clearly illustrates the

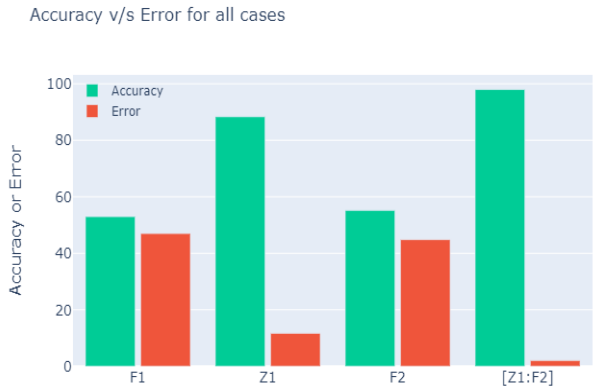
difference in the accuracy of a univariate distribution, univariate distribution with normalized subjects and multivariate data as follows:











VI. Conclusion

From the graphs and the accuracy table, it is clear that when the subjects are passed to the classifier without normalizing, the accuracy tends to drop to less than 50%, with a error rate of more than 50%, the reason for it being the range of values overlap each other and hence the model obtained is unable to classify the data as designed. However, when we normalize the subjects, each of these classes have a mean centered at 0 and a standard deviation of 1, which defines the range of classes more distinctly, thereby increasing accuracy. However, in Case IV, for the multivariate distribution, we see that the accuracy is the highest. This is due to the fact that, apart from having a larger dataset with more training data, the model also has a set of normalized subjects which increase the accuracy of the model up to ~98%.

By the results above, we can conclude how Bayes' theorem can be used to successfully classify the data. However, this accuracy is obtained keeping in mind the fact that a Naïve-Bayes' classifier considers each subject to be independent, which is hard to find in practical applications.

VII. References

1. https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Examples
2. John Tsitsiklis, Dimitri Bertsekas: Introduction to Probability (2nd edition)
3. https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf