**Calculating P-value Using Gaussians Models**

Yaosheng Xu

2020 Fall

**Abstract**

One problem of modern Machine Learning classification methods is that they always hard predict a class, even if the given data is not from any of the classes. To solve this problem, our research focuses on finding the data point’s P-value of each class. From previous work, we have a Nearest Neighbor Distance Model, which fits a distribution of nearest neighbor distances to calculate the given data point’s P-value of each class. However, there are two issues: 1) it cannot detect an outlier. 2) it cannot predict if two clusters are differentiate not based on distance (overlap clusters).

I tried two ways to solve the problem. The first one is to use a Gaussian Naïve Bayes Model. The second one is to fit the data into Multivariate Gaussian Distribution. The main goal is to design a model that performs better on above two situations, and then we can use ensemble methods to combine all models. (Ensemble details are in another report).

**Introduction**

Gaussian Naïve Bayes Model will give us the posterior probability of each class given a data point. Using these posterior probabilities, we can empirically count how many more points are more extreme than the given one. This is an empirical way to calculate P-values.

Multivariate Gaussians Model calculates the mean and the covariance for each class, and then fit a Gaussian Distribution (a bell shape curve) for each of the class. When predicting, it gives P-value by calculating the given data point’s CDF for each Gaussian Distribution, namely how many more extreme cases (in percent) are there. So close to 0 means unlikely, and close to 1 means likely. This is a theoretical way to calculate P-values.

Both methods will perform better on detecting outliers and overlap clusters compared to Nearest Neighbor Distance Model, because Gaussian models take not only distance but also covariance into account.

**Detailed Approach**

1. **Gaussian Naïve Bayes Model**

Gaussian Naïve Bayes Model train data by calculating the posterior probability using Naïve Bayes Rule:

When predicting, it calculates P-values by empirically counting how many training data has lower posterior probability than itself for class k, then divide by the number of all training data in class k:

1. **Multivariate Gaussians Model**

Multivariate Gaussians Model train data by fitting all the training data into multiple Gaussian Distributions as the following formula:

where *d* is a real k-dimensional column vector, and is the determinant of . After that, we will get a “bell shape curve” *for each class*.

After fitting, Multivariate Gaussians Model gives P-values of class k by calculating the CDF of the given data in the kth Gaussian Distribution:

If we want to predict a class, we can simply pick the class who has the highest P-value:

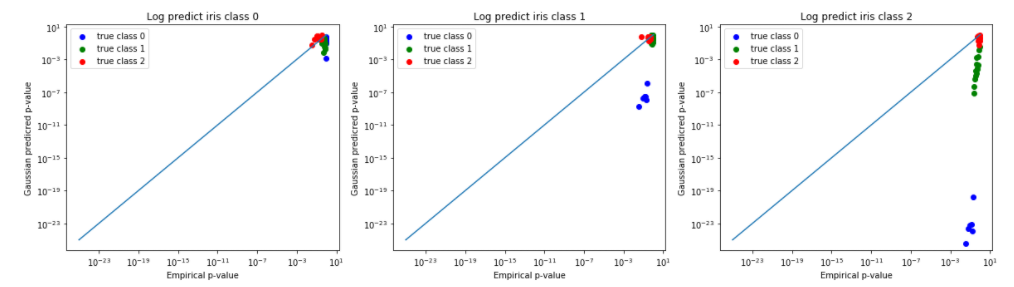
**Performance:**

1. **Accuracy (Train: 80%, Test: 20%)**

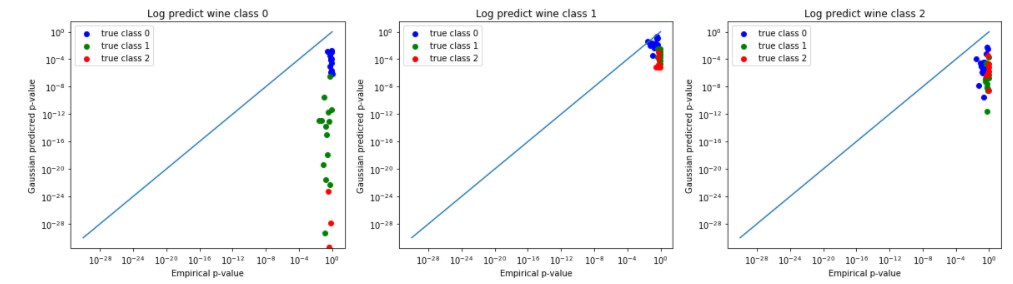
|  |  |  |  |
| --- | --- | --- | --- |
| Classifier/Dataset | Iris | Wine | Bio grid |
| Gaussian Naïve Bayes Model | 0.9 | 0.91 | 0.39 |
| Multivariate Gaussian Model | 0.8 | 0.75 | <0.1 & time-consuming |

\*Accuracy may flow because of random split of dataset

1. **Plot**
   1. Iris dataset using Multivariate Gaussian Model:



* 1. Wine dataset using Multivariate Gaussian Model:



**Conclusion:**

We can see that Naïve Bayes Model has a better performance than Multivariate Gaussians Model. This may because the Multivariate Gaussians Model over-fits the dataset. Also, both models perform poor on Bio Grid dataset, because there are banana-shaped clusters, which is more suitable for Nearest Neighbor Distance Model. Having all these models, next step is to use an ensemble method to combine them and give a more reasonable result.