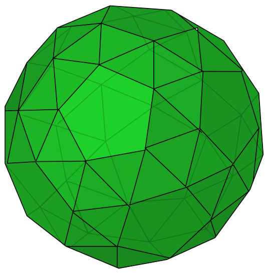
Projects in Mathematics and Applications

Naive Bayes Classification for Fraud Detection

August 8th, 2018

Nguyen Nguyen [*]   [†]Huynh Pham Phuong Mai
Nguyen Tan Dat [‡]   [§]Nguyen An Truong



[*]High School for the Gifted, Vietnam National University - Ho Chi Minh City
[†]Le Hong Phong High School for the Gifted - Ho Chi Minh City
[‡]Pedagogy High School for the Gifted - Hanoi
[§]Nguyen Tat Thanh High School for the Gifted - Kon Tum

Contents

Acknowledgement

The authors would like to give special thanks to the organizers of Projects in Mathematics and Application 2018 (PiMA), which has provided great environment and inspiration for us to dig deeper into the knowledge of Applied Mathematics, in general, and Machine Learning, in particular. Not only with mathematics knowledge, but PiMA also equipped us with necessary soft skills such as teamwork, scientific research, and presenting in front of famous professors and mentors in the field. These are the skills we lack exposure to in public high schools. Furthermore, we would like to express our appreciation to our mentors: Mr. Manh Nguyen

Pham and Mr. Nhat Hoang Pham, whose guidance were invaluable during this project.

Introduction

In this research, we mostly focus on classification method, Naive Bayes Classifier (NBC), rooting in Bayes' theorem - Naive Bayes which describes the probability of an event, based on prior knowledge of conditions that might be related to the event. Besides, we would like to introduce NBC's application in detecting fraud and related problems such as: imbalanced data and updated data.

Naive Bayes has been studied extensively since the $1950s$. It was introduced under a different name into the text retrieval community in the early 1960s, and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification. With a small amount of training data, if the condition of independent data is qualified, this method is believed to have better results compared to Support Vector Machine (SVM) and logistic regression.

NBC can work with the vectors of features which are partially continuous (by using Gaussian Naive Bayes) and discrete (by using Multinational or Bernoulli Naive Bayes). When using Multinational Naive Bayes, Laplace smoothing is usually used to avoid the case that a part of testing data does not appear in the training data.

1 Introduction to Naive Bayes classifier

Naive Bayes is a Machine Learning Model (MLM) used for classifying data on the basis of **Maximum a Posterior Estimation** and Bayes' Theorem. It is a simple technique for constructing classifiers yet fundamental and vital to build more complicated ML model.

1.1 Maximum a Posterior Estimation

One of the most indispensable basis of **Naive Bayes classifiers** is the **estimation the Maximum of a Posterior Probability** (MAP). To understand the probabilistic algorithms, we shall recall some notions of the MAP estimation.

Definition 1.1. Given a family of distributions $D(\theta)$, where θ is a random variable with prior probability $\theta \sim P$ and a random variable $X \sim D(\theta)$ and n values attained from it $x_1, x_2, ..., x_n$. The **maximum estimation of Posterior** of θ is the value θ_0 so that the probability:

$$P(\theta = \theta_0 | (X_1 = x_1)(X_2 = x_2)\dots(X_n = x_n))$$

is maximal, where $X_i \sim D(\theta_0)$ is the value of X in the i^{th} try.

In short, we denote the maximal posterior estimation of θ as:

$$\theta^* = \arg\max_{\theta \sim P} P(\theta | x_1, x_2, ..., x_n).$$

By using Bayes' theorem, we have:

$$\begin{aligned}
\theta^* &= \arg\max_{\theta \sim P} P(\theta | x_1, x_2, ..., x_n) \\
&= \arg\max_{\theta \sim P} \frac{P(x_1, x_2, ..., x_n | \theta) . P(\theta)}{P(x_1, x_2, ..., x_n)} \\
&= \arg\max_{\theta \sim P} P(x_1, x_2, ..., x_n | \theta) . P(\theta).
\end{aligned}$$

Assume that all features are mutually independent, and using logarithm, we get:

$$\begin{aligned}
\theta^* &= \arg\max_{\theta \sim P} P(\theta | x_1, x_2, ..., x_n) \\
&= \arg\max_{\theta \sim P} P(x_1, x_2, ..., x_n | \theta) . P(\theta) \\
&= \arg\max_{\theta \sim P} P(x_1|\theta).P(x_2|\theta)...P(x_n|\theta)P(\theta) \\
&= \arg\max_{\theta \sim P} \left[\sum_{i=1}^{n} \log(P(x_i|\theta)) + \log(P(\theta)) \right].
\end{aligned}$$

The maximum estimation of θ^* depends on determining the prior probability $P(\theta)$ and the likelihood of the data $P(x_i|\theta)$. Defining the prior probability is really essential to get exact results from MAP. However, we can't choose these distributions exactly, yet in experiment we can approximate it by using **Beta distribution** or **normal distribution**.

Beta distribution is define as:

$$(P(\theta)) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{a-1} (1-\theta)^{b-1},$$

where a,b are hyper-parameters fixed before. When no information about prior distribution is known, we usually choose $a = b = 1$.

Normal distribution is a distribution described by two parameters of the expectation μ and variance σ^2 of the distribution. It is represented by a probability density function:

$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

1.2 Naive Bayes Classifier

In fact, when we work with statistics and classification problems, one of the most effective method is using probabilistic models. The simple idea is that: For a given vector of features, we classify it to a class with the highest possibility. A popular method to estimate this probabilistic factor is Naive Bayes Classifier.

Definition 1.2. (Classification problem)
Given an unknown **objective function** $c : \mathrm{R}^d \to C$, where R^d is an input space and $C = c_1, ..., c_m$ are m **classes**, and n training data:

$$D = \{\langle \mathbf{t}_1, c(\mathbf{t}_1)\rangle ; \langle \mathbf{t}_2, c(\mathbf{t}_2)\rangle ; ...; \langle \mathbf{t}_n, c(\mathbf{t}_n)\rangle\},$$

with $\mathbf{t}_i \in \mathrm{R}^d$, $c(\mathbf{t}_i) \in C$. For an untrained data $\mathbf{x}$, estimate a best suitable class $c(\mathbf{t}) \in C$ for $\mathbf{x}$.

The main idea of the problem is to find a class that has the highest possibility of containing $\mathbf{x}$ based on the classification of $\mathbf{t}_1, \mathbf{t}_2, .., \mathbf{t}_n$. To estimate this, we use **MAP**. Our Naive Bayes algorithms based mainly on this tool.

Assume that we have already been given classes c and n training data. Then a new untrained data will be classified by

$$\begin{aligned} c^*(\mathbf{x}) &= \arg\max_{c\in C} P\left(c(\mathbf{x}) = c | x_1, x_2, ..., x_d\right) \\ &= \arg\max_{c\in C} P\left(x_1, x_2, ..., x_d | c(\mathbf{x}) = c\right).P(c). \end{aligned}$$

The method strongly rely on Bayes' theorem, a fundamental basis of MAP estimation. To make the last value easier to verify, we "naively" presume that all the features are mutually independent. In order words, we assume that $x_1, ..., x_d$ are independent variables. Then we have:

$$\begin{aligned} c^*(\mathbf{x}) &= \arg\max_{c\in C} P\left(x_1, x_2, ..., x_d | c(\mathbf{x}) = c\right).P(c) \\ &= \operatorname*{argmax}_{c\in C} \prod_{i=1}^{d} P(x_i|c).P(c) \\ &= \operatorname*{argmax}_{c\in C} \sum_{i=1}^{d} \log(P(x_i|c)) + \log(P(c)). \end{aligned}$$

In the last, to estimate the optimal class $c(\mathbf{x})$, we need to verify the prior distribution and the likelihood of the features with their respective classes. These two values depend on the input data and data processing (for example, the prior prob. $P(c)$ changes every time a new datum is

trained). Proper calculation of these processes and choice of suitable data for the Naive Bayes model is essential to make the most exact estimation.
One simple way to estimate $P(c)$ and $P(t_i|c)$ is to list form the data themselves. For instance, we can estimate $P(c)$ by:

$$P(c) = \frac{|\{1 \leq k \leq n : c(\mathbf{t}_k) = c\}|}{n}.$$

With the likelihood, we can calculate it by using the ratio:

$$P(c) = \frac{|\{1 \leq k \leq n : (c(\mathbf{t}_k) = c) \wedge (\mathbf{t}_k(i) \approx x_i)\}|}{|\{1 \leq k \leq n : c(\mathbf{t}_k) = c\}|}.$$

This estimation has various disadvantages. Still, we can use some more effective estimation like assigning the Beta distribution for the prior distribution.

One note is that in reality, these features are rarely independent. However, we can solve this by using others method (like Principle Component Analysis (PCA),...) to increase the independence of features for better accuracy.

1.3 Naive Bayes Classifier implications

1. Used in text categorization model.
2. The data training and testing time is fast due to independence of features.
3. When the data is few, the accuracy is higher than those of SVM and logistic regression.
4. Can be used for featured vector that is partially continuous and discrete.

2 Bayes classification of continuous variables

For the introduced Naive Bayes classifier, the number of classes is assumed to be discrete, which means there are either finite or countably infinite classes. A question that arises is how it can be used for continuous variables classification.

An example is the problem of predicting a person's age given some features such as height, sex and weight. Here, age is a continuous variable.

In Naive Bayes, two main elements which estimate the posterior is prior and likelihood. A common continuous probability distribution (especially for wide continuous range) is normal distribution (Gaussian distribution). Using those factors, we will derive a method for approximating posterior of a continuous variable with given data.

Supposed that, we need to estimate the posterior $p(\theta|x_1, ..., x_d)$ for $x_1, ..., x_d$ is the given features and θ is the continuous variable to predict. By Bayes theorem, θ is estimated by

$$\theta^* \sim p(\theta)\prod_{i=1}^{d} p(x_d|\theta).$$

Since θ is continuous, we can assume that $p(\theta)$ follows normal distribution with known mean μ_0 and variance σ_0^2.

$$\theta \sim N(\mu_0, \sigma_0^2).$$

Then, for each i, let:

$$x_i|\theta \sim N(\theta, \sigma^2).$$

Hence, we have:

$$\begin{aligned}\theta^* &\sim p(\theta)\prod_{i=1}^{d} p(x_d|\theta)\\ &\sim \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(-\frac{(\theta-\mu_0)^2}{2\sigma_0^2}\right)\cdot\left(\prod_{i=1}^{d}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x_i-\theta)^2}{2\sigma^2}\right)\right)\\ &\sim \left(-\frac{(\theta-\mu_0)^2}{2\sigma_0^2}-\sum_{i=1}^{d}\frac{(x_i-\theta)^2}{2\sigma^2}\right).\end{aligned}$$

A possible method is to separate the variable that needs classification into multiple continuous ranges. By this way, we can have finite number of classes. To derive the posterior given data for a class, we need to specify a probability density function $D(\theta)$

$$p(\theta|x_1, ..., x_d) = \frac{p(\theta)\prod_{i=1}^{d} p(x_d|\theta)}{p(x_1, ..., x_d)} = c.f(\theta;\mu_0,\sigma_0).\prod_{i=1}^{d}.f(x_i;\theta,\sigma) = D(\theta).$$

For $f(x; a, b)$ is the probability density function of normal distribution with mean a and standard deviation b and $c = \frac{1}{p(x_1,...,x_d)}$. As the features $(x_1, ..., x_d)$ and parameters $\mu, \sigma, \mu_0, \sigma_0$ are given, the expression can be written as a function of θ, which is $D(\theta)$. Thus, a class specified by $a \le \theta \le b$ has posterior

$$p(a \le \theta \le b|x_1, ..., x_d) = \int_a^b D(\theta)d\theta.$$

In real life, a variable may be classified in to both continuous and discrete classes. Each has a different method of estimating posterior. Finally, Maximum a Posterior Estimation is given by the formula

$$c^*(\mathbf{x}) = \underset{c \in C}{\operatorname{argmax}}\; p(c(\mathbf{x}) = c | x_1, ..., x_d).$$

For C is the set of classes.

3 Updating data by Bayes inference

We can see that updating data, information, or evidence, to some extent, affects to the initial prior probability. For example, the chair of court is judging whether a man is a crime or not with a prior probability of guilty is 80% and of innocence is 20%. After updating new evidence to prove that the man is innocent, initial prior probability changes respectively (the increase in probability of innocence and the decrease in the probability of guilty). We can understand that the change in prior probability is deemed as the changing belief in null hypothesis.

Bayesian inference computes the posterior probability according to Bayes' theorem:

$$P(H_0|E) = \frac{P(E|H_0).P(H_0)}{P(E)}.$$

In which:

- E updated data
- H_0 null hypothesis before E
- $P(H_0)$ prior probability of H_0
- $P(E|H_0)$ conditional probability when new evidence E is updated knowing that H_0 is correct
- $P(E)$ marginal probability of E
- $P(H_0|E)$ posterior probability of H_0 knowing E.

The factor $\frac{P(E|H_0)}{P(E)}$ can be interpreted as the impact of E on the probability of H. If the updated data results in correct hypothesis, this quotient will be big leading to big value of $P(H_0|E)$. Therefore, in Bayes inference, Bayes theorem can make changing belief in a hypothesis.

Example 3.1. Wrong positive results in medical analysis

Supposing that a medical analysis for a certain type of disease have the results of probability for:

- Correct positive result: $0,99$
- Correct negative result: $0,95$

Assuming that there is only $0,1\%$ of the population having this type of disease (of course, we just stop at the initial estimation without knowing exactly about its probability), randomly pick patient A then his **prior probability** of being infected is $0,001$.

Let's assume that

- A: the event of the patient A being infected
- B: updated data

The question is: "If there is one more patient, patient B, doing this medical analysis and receive a positive result, what will be his probability of truly having this disease?" The answer for this problem lays in Bayes' theorem. Exactly, patient B's probability of being infected while receiving a positive result is:

$$\begin{aligned} P(A|B) &= \frac{P(B|A).P(A)}{P(B|A).P(A) + P(B|notA).P(notA)} \\ &= \frac{0,99.0,001}{0,99.0,001 + 0,05.0,999} \approx 0,019. \end{aligned}$$

Though it may seem that the probability of a correct result is small yet in comparing to the prior probability of being infected $(0,1\%)$, this new result is 19 times higher. The meaning of this result is: based on the new data that we observe (positive result), our belief in the ability of getting the disease has changed, for example we think that the risk of getting a disease is higher than its initial probability (and it's truly 19 times higher!, though the posterior probability is still small) because the ability of predicting a positive result is still high. Therefore, this result is not useless yet it changes our belief.

If we explain this example by the language of Bayes classification, we can see there are two classes: infected or not infected with the condition of negative results. Then, when a new data is updated (a positive result, for example), our belief in the risk of getting infected will be changed, as well as our prior probability.

Considering the same example, we observe how our belief will change based on the input data. Before having the diagnose result, the probability of infected is 0.001. However, after observing the analysis' result, we operate the prior probability from $P(\text{infected}) = 0,001$ to $P(\text{infected}|\text{positive}) = 0.019$. Therefore, by every time updating a datum, we will update **prior probability** by **posterior probability**.

Considering the Bayes classification's problem: with n data points $x_1, ..., x_n$ of d properties classified in m classes $c_1, ..., c_m$. Then, we can observe data point x. Assuming after that, we have the prior probability of classes is $p(c_1), ..., p(c_m)$. Then, with $1 \leq k \leq m$, we have the posterior probability is $p(c_k|x)$, which is the probability showing our belief in class c_k after analyzing x. Therefore, when observing x, we can operate the prior probability: with $1 \leq k \leq m$

$$p(c_k) := p(c_k|x)$$

The purpose of this formula is to assign the posterior probability $p(c_k|x)$ to the prior probability $p(c_k)$.

Note Bayes inference is used to calculate the decision probability in uncertain situation. Beside the probability, we should calculate a loss function so as to reflect the consequence of committing a mistake.

We could look at another example of updating data. In this case, the variables are continuous. Based on the above formula, we know that

$$p(\theta|x_1, ..., x_d) = c.f(\theta; \mu_0, \sigma_0). \prod_{i=1}^{d} .f(x_i; \theta, \sigma) = D(\theta).$$

with θ is a continuous variable respecting with a continuous class, $(x_1, ...x_d)$ is a training data point.

Through some transformations, we could point out that

$$P(\theta|x_1, ..., x_d) \sim f(\theta; \mu_1, \sigma_1)$$

with

$$\sigma_1^2 = \left(\frac{1}{\sigma_0^2} + \frac{d}{\sigma^2}\right)^{-1}$$

and

$$\mu_1 = \sigma_1^2 \left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{d} x_i}{\sigma^2}\right)$$

It means that after updating the prior probability of each continuous variable θ by the posterior probability $P(\theta|x_1, ..., x_d)$ based on new data $(x_1, ..., x_d)$, the prior probability is still a normal distribution, with its mean value and variance are updated as below. This makes sense and allows us to continue updating data. Assuming that our class is a continuous interval $[a, b]$, we will update each prior probability by new data, then updating the prior probability for class by:

$$P'([a, b]) = \int_a^b P'(\theta) d\theta$$

with $P'(\theta)$ is the prior probability after updating θ.

4 Imbalanced data

Let's imagine you're a teacher. Your principal ask you to predict the final results of your students in the incoming school year: pass of fail. In order to finish this task, you have to gather data of your students' results from previous school year: grade, attendance, and final score. Assume that you're an excellent teacher so that almost none of your students failed the class. Let's say 99% of your old students passed your class.
The fastest and most direct way is to predict that 100% of your students will pass the class. The accuracy in this case is 99% compared to previous school years. However, this model is no longer correct if new evidence from a new student is added that contradicts your prediction.
You can estimate that all of your students in this school year will pass the class based on previous semesters and still receives high accuracy. However, this prediction is not valuable and unable to express the essence of the data set.

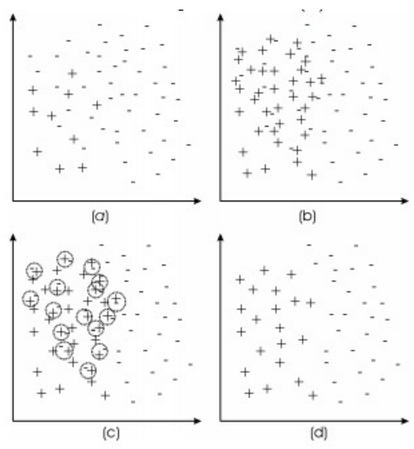
In applying Naive Bayes in calculating probability and categorizing data, imbalanced data may result in conclusion with high accuracy yet not really express thoroughly the inside feature of the data set. Therefore, dealing with imbalanced data is an important problem that will be discussed in this section with two most common given solutions.

- **Random-over sampling:** is a non-heuristic method that balances the class distribution through the random replication of positive examples.
- **Random-under sampling:** is also a non-heuristic method that balances the class distribution through the random abolition of positive examples.

There are lots of beliefs that Random-over sampling increases the potential of over-fitting as it replicates exactly the data of the majority class. With Random-under sampling, the biggest drawback is the risk of losing important data. Therefore, two aforementioned methods are used along with other methods to optimize the results.

- **Tomek link** Assuming 2 data E_i and E_j belong to 2 different classes and $d(E_i, E_j)$ is the distance between E_i and E_j. Pair $A(E_i, E_j)$ is called Tomek link if there is no E_l that $d(E_i, E_l) < d(E_i, E_j)$ or $d(E_j, E_l) < d(E_i, E_j)$. If there are 2 data that form Tomek link, one of them will be interfered or both of them will lay at the border line. Tomek links could be used as:
  - under-sampling method: abolish data from the majority class.
  - data cleansing method: abolish data from both majority and minority examples.
- **One-sided selection (OSS)** a method of under-sampling inferred from the application of Tomek links in Convolutional Neural Network (CNN). Tomek links is used as an under-sampling method and able to clean interfered data and data at border line from the majority class. Data at border line cause "danger" as it only needs one wrong categorized datum to misclassify the data at the decision line. The rest of the data set (safe data from the majority class and all data in the minority one) will continue be used to examine and calculate.
- **Smote Synthetic Minority Over-sampling Technique (Smote)** an over-sampling method whose main idea is to insert new data into available data lying next to each other in the minority class. Therefore, we could avoid over-fitting and expand the decision line of the minority class, approaching closer to the majority class's space. However, some data of

the majority class may poach to the minority class' space. The opposite case may happen as inserting data into the minority class could expand its area and bring the synthetic minority class examples deeply into the majority class' space. Categorization in this situation could lead to over-fitting. Therefore, instead of deleting only positive data to form Tomek links, we could use Tomek links along with SMOTE so the data from both classes will be abolished naturally. The application of this combined method is described in the below figure.



Firstly, original data set (a) is over-sampled with SMOTE (b). Following that, Tomek links is formed (c) and deleted, which then creates a balance data set with clear divided clusters (d).
SMOTE + Tomek links method was first used to improve the categorization for explanation and names of proteins in Bioinformatics.

- **Neighborhood Cleaning Rule (NCL)** using Wilson's Edited Nearest Neighbor Rule (ENN) to eliminate data from the majority class. ENN deletes data whose class is labeled different from at least 2 in 3 adjacent data. NCL is used along with ENN to increase the amount of cleansed data. With the unbalance between 2 classes, for each of the data E_i in the training data set, 3 adjacent data will be examined. If E_i is in the majority class and 3 adjacent data are not, then E_i will be eliminated. Otherwise, if E_i is in the minority class and 3 adjacent data are not, then the nearest datum from the majority class will be eliminated.

In the aforementioned methods, over-sampling method in general, and SMOTE + Tomek and SMOTE + ENN in specific, give out the best result when they are applied for data set with the minority class of positive examples. Furthermore, Random-over sampling is usually seen as unstable method, though it may result in competitive conclusions compared to other more sophisticated methods. SMOTE + Tomek or SMOTE + ENN is proposed to be applied in data set with a small amount of positive examples. For data set with huge amount of positive examples, Random-over sampling method not only is conducted more easily on computers than other methods but also leads to trustful results.

5 Selecting independent features

5.1 Problem

Machine learning is conducted by a simple rule: if the input data are disturbed, so are the output.
If the input data are too huge, it's unnecessary to use the whole data set for machine to learn. We just have to provide with paramount data so as to:

1. Fasten the process.
2. Decrease the complication of the model.
3. Increase the accuracy of the model.
4. Minimize the risk of over-fitting.

Therefore, it's important to find a suitable method for selecting necessary data to increase the efficiency of calculation. One of the most simple method is **Pearson's Correlation**

5.2 Pearson's Correlation

5.2.1 Pearson correlation coefficient (PCC)

Before getting to know about Pearson's Correlation, we need to have a brief understanding of one of the most common coefficient in statistics - Pearson correlation coefficient, also referred to as Pearson's r, the Pearson product-moment correlation coefficient (PPMCC) or the bi-variate correlation.

PCC is a measure of the linear correlation between two variables X and Y, for example between SATISFACTION LEVEL (y) and SALARY (x). It has a value between -1 and 1, where:

- 0 (or nearly 0) is no linear correlation
- -1 is total negative linear correlation
- 1 is total positive linear correlation
- negative means when x highly increases, y decreases, and vice versa
- positive means when x highly increases, y also increases

5.2.2 Idea

In statistics, Pearson's correlation coefficient is the co-variance of the two variables divided by the product of their standard deviations. The form of the definition involves a "product moment", that is, the mean (the first moment about the origin) of the product of the mean-adjusted random variables; hence the modifier product-moment in the name.

5.2.3 Method

Given two variables X, Y, before applying Naive Bayes in categorizing, we need to test the independence of X, Y (it's similar for other problems that have more than two variables).

The correlation coefficient $\rho_{X,Y}$ of two variables X, Y with respective expectation μ_X, μ_Y and standard deviation σ_X, σ_Y is calculated:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E\left((X-\mu_X)(Y-\mu_Y)\right)}{\sigma_X \sigma_Y}$$

We have:

$$\mu_X = E(x)$$
$$\sigma_X{}^2 = E\left[(X - E(X))^2\right]$$

Similarly to Y.

Furthermore, we have:

$$E\left((X-\mu_X)(Y-\mu_Y)\right) = E(XY) - E(X)E(Y)$$

By transforming the above formula, we get:

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}.\sqrt{E(Y^2) - E^2(Y)}}$$

The correlation coefficient's value is in [-1;1]. Therefore, there are some notes for the result of the correlation coefficient:

- If $\rho_{X,Y} = 0$ then two variables X, Y are independent. We could use data from both variables to work on.
- The more its absolute value comes close to 1, the higher level of correlation between two variables. We can randomly choose one of them to be the input data.
- Another values show respective level of correlation between two variables.
- When there are too many input data, we have to compare them in pair to choose the most suitable one.

5.3 Introduction to Principle Component Analysis (PCA)

One of the most common and efficient method to process and increase the independence among conditions is **Principle Component Analysis (PCA)**. Using PCA, we could change the initial conditions to new conditions that are both connective and independent from each other. To make these conditions become independent, we need to calculate on co-variance matrix and find an optimized vector that is satisfied with a given feature. Basically, the idea of PCA is quite close to and more generic than Pearson's Correlation.

6 Applications

Implementing Naive Bayes in a practical situation: fraud credit cards.

Note:

- Assuming that our features are independent from each other and there are 2 classes:
  - Label 1: fraud
  - Label 2: not fraud
- Choosing data and training data with a decent amount of credit cards as in reality, the amount of fraud credit cards is small so it's easy to have imbalanced data. After testing, the probability of fraud card is 0 yet it's not true in reality.

Code:

- Import modules

```
import numpy
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, classification_report,
    accuracy_score
```

- Import data

```
Location = r'D:\python\creditcard_it.csv'
df = pd.read_csv(Location)
```

- Select features and data labels

```
X = df.drop('Class', axis = 1)
y = df['Class']
```

- Data split: 80% for training and 20% for testing

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

- Since features are continuous, we use Gaussian distribution (Normal distribution)

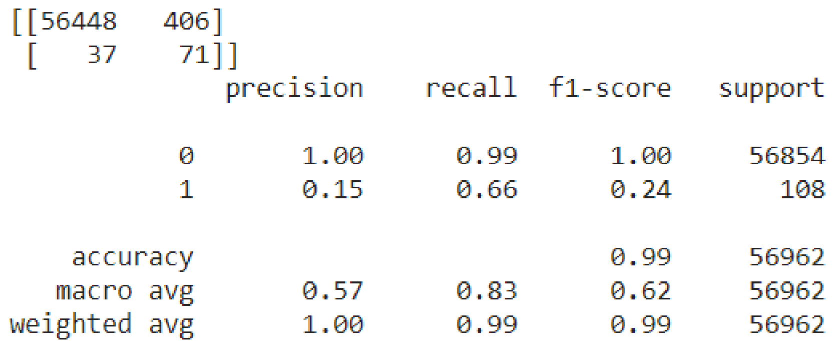
```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

- Make prediction on the test set

```
test_predict = gnb.predict(X_test)
print('Predictions using the testing set:\n',test_predict)
```

- After finishing predicting the result of the training process, it's impossible to get a perfectly correct result. Instead, we have to test their accuracy by using accuracy score, average precision score, and recall score

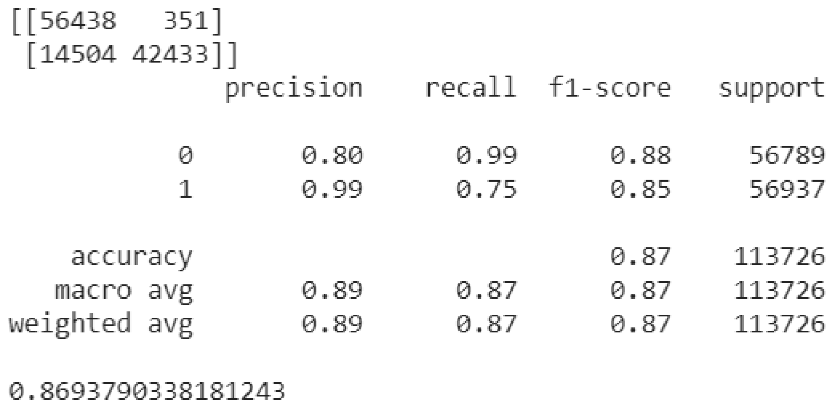
  ```
  print(confusion_matrix(y_test, test_predict))
  print(classification_report(y_test, test_predict))
  ```

  ```
  [[56448   406]
   [   37    71]]
                precision    recall  f1-score   support

             0       1.00      0.99      1.00     56854
             1       0.15      0.66      0.24       108

      accuracy                           0.99     56962
     macro avg       0.57      0.83      0.62     56962
  weighted avg       1.00      0.99      0.99     56962
  ```

- The accuracy score of the training is 0.869. The result looks good for 0, but it is significantly bad when predicting 1. This is due to the fact that our data is highly imbalanced. However, some features may be correlated, so we can try to remove them and retrain the model. To fix that, we process the imbalanced data with SMOTE method:

  ```
  X = df.drop('Class', axis = 1)
  y = df['Class']
  oversample = SMOTE()
  X, y = oversample.fit_resample(X, y)
  X = pd.DataFrame(X)
  y = pd.DataFrame(y)
  ```

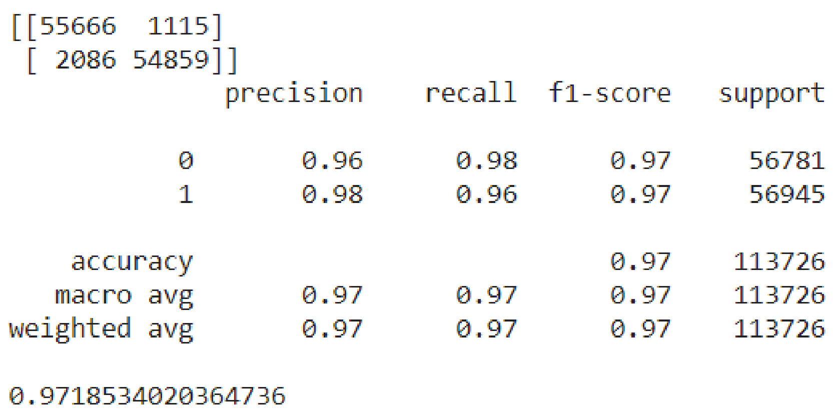
  

  ```
  [[56438   351]
   [14504 42433]]
                precision    recall  f1-score   support

             0       0.80      0.99      0.88     56789
             1       0.99      0.75      0.85     56937

      accuracy                           0.87    113726
     macro avg       0.89      0.87      0.87    113726
  weighted avg       0.89      0.87      0.87    113726

  0.8693790338181243
  ```

  The precision when predicting fraud now improves, but the accuracy score remains roughly the same.

  In reality, 86% **of fraud credit cards is detected.**

- We compare our method efficiency with logistic regression. The following are the confusion matrix and the classification report when training our balanced data using logistic regression.



```
[[55666  1115]
 [ 2086 54859]]
              precision    recall  f1-score   support

           0       0.96      0.98      0.97     56781
           1       0.98      0.96      0.97     56945

    accuracy                           0.97    113726
   macro avg       0.97      0.97      0.97    113726
weighted avg       0.97      0.97      0.97    113726

0.9718534020364736
```

We can see that using logistic regression gives a significantly better prediction. So we can fairly say that Gaussian Naive Bayes, even though it is easy to implement and run faster, but it has low predicting capability.

7 Conclusion

Naive Bayes Classification (NBC) is an effective method for categorizing sophisticated data, different kinds of variables which are both continuous and discrete. Moreover, even if the features are not too independent, it still leads to a result with a high level of accuracy. This means that sometimes, we don't have to process features before working and training data yet still receive an accurate conclusion.

Furthermore, NBC could be used to predict the change in a prior distribution system after a data set is trained. Therefore, NBC has lots of applications in life, as well as presents the close relation between different data. Based on this, we can operate, organize data more suitably to get more possible and accurate result.

However, in some cases, this method have some specific drawbacks. For example, as naive Bayes is easily get affected by the difference among data, it's hard to process data in case of imbalanced data. It means that there are great differences among data so we have to deal with imbalanced data before going to train and process them. Besides, with naive Bayes' specific characteristic of having great independence between two features, we need to operate its error value and our expectation in imbalanced data to receive the most accurate result.

In general, naive Bayes Classification is a good model for classifying probability and has lots of applications in machine learning. However, some points in these formulas still need to be investigate more and can be expanded to more generic model to receive better results.

References

[1] *Bayes Inference for the Normal Distribution*.

[2] Maria Carolina Monard Gustavo E. A. P. A. Batista, Ronaldo C. Prati. *A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data*. Instituto de Ciências Matemáticas e de Computação, 2004.

[3] Kaggle. *Fraud Detection with Naive Bayes Classifier*. Credit Card Fraud Detection, 2018.

[4] Rafael Pierre. *Detecting Financial Fraud Using Machine Learning: Winning the War Against Imbalanced Data*. 2018.

[5] Tran Hoang Bao Linh Vu Le The Anh, Pham Nguyen Manh. *Probability and Distribution*. Projects in Mathematics and Applications, 2018.