**NLP Project 1: Document Classification**

**By Kulboboev Shukhrat**

**The name of the class: Natural Language Processing**

**Professor: Hyunsouk Cho**

**The name of University : Ajou University**

**The Date :**

**Contents:**

**Naive Bayes method**

1. Gaussian naive Bayes
2. Bernoulli naive Bayes
3. Multinomial naive Bayes

**Experimental Setting**

1. Dataset (statistical information, train/dev/test, etc.)
2. Evaluation measure
3. Baselines

**Experimental Result**

1. Overall performance
2. Qualitative results

**Conclusion**

**Introduction to the Naive Bayes Algorithms**

The simplest solutions are usually the most powerful ones in classification, and [Naive Bayes](https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html) is a good example of that. Despite the advances in Machine Learning in the last years, it has proven to not only be simple but also fast, accurate, and reliable.

It has been successfully used for many purposes, but it works particularly well with natural language processing (NLP) problems.

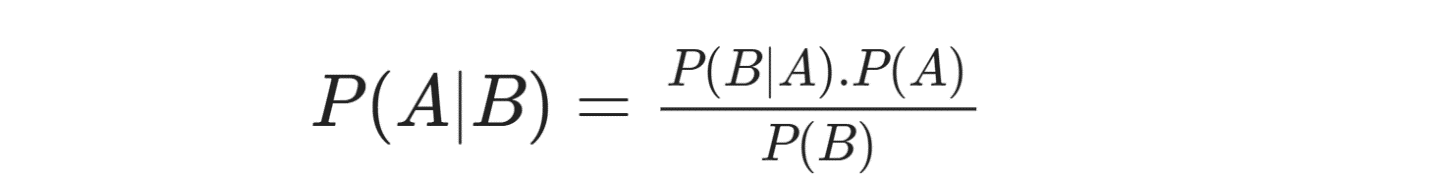
Naive Bayes is a probabilistic machine learning algorithm based on the **Bayes Theorem**, used in a wide variety of classification tasks. In this article, we will understand the Naïve Bayes algorithm and all essential concepts so that there is no room for doubts in understanding.

**Bayes Theorem**

Bayes’ Theorem is a simple mathematical formula used for calculating conditional probabilities.

**Conditional probability** is a measure of the probability of an event occurring given that another event has (by assumption, presumption, assertion, or evidence) occurred.

The formula is:



Where

* A and B are two events

P(A|B) is the probability of event A provided event B has already happened.

P(B|A) is the probability of event B provided event A has already happened.

P(A) is the independent probability of A

P(B) is the independent probability of B

Which tells us: how often A happens *given that B happens*, written **P(A|B)**also called posterior probability, When we know: how often B happens *given that A happens*, written **P(B|A)**, and how likely A is on its own, written **P(A)** and how likely B is on its own, written **P(B).**

**Types of Naive Bayes Classifiers**

1)**Gaussian Naive Bayes Classifier**:

In Gaussian Naïve Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution (**[Normal distribution](https://en.wikipedia.org/wiki/Normal_distribution)**)**. When plotted, it gives a bell-shaped curve that is symmetric about the mean of the feature values as shown below:

Letter

Description automatically generated with low confidence

Diagram

Description automatically generated with medium confidence

2. **Bernoulli Naive Bayes Classifier**:  
In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence (i.e. a word occurs in a document or not) features are used rather than term frequencies (i.e. frequency of a word in the document).



3. **Multinomial Naïve Bayes Classifier**  
Feature vectors represent the frequencies with which certain events have been generated by a **multinomial distribution**. This is the event model typically used for document classification.

Diagram

Description automatically generated with medium confidence

**Experimental Setting**

In the experimental section, I will choose adult.csv datasets (https://www.kaggle.com/datasets/shreyshi/adultcsv) from the Kaggle.com website for implementing my classification project namely comparing different methods with each other and checking their performances. Before using this data, we should clean and prepare it for our experiments. In the below, I made a data-checking process and prepare data for the test. I used python 3 computer language and Spyder counsel. My purpose is to predict whether income exceeds $50K/yr namely classifying incomes more than $50K/year based on given data.

**Import libraries**

Text

Description automatically generated

**Import dataset**

I imported data from a file on my computer by using pandas libraries.



A picture containing text

Description automatically generated

**Exploratory data analysis**

Now, I will explore the data to obtain insights into the data.

Graphical user interface, application

Description automatically generated

We can see that there are 32560 instances and 15 attributes in the datasets.

**Top 5 rows**

Graphical user interface, text, application

Description automatically generated

**Rename column names.**

We can see that the dataset does not have proper column names. I should give proper names to the columns:

Text, application

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

**View the whole dataset information:**

Graphical user interface, text

Description automatically generated

There are no missing values in the above image.

**Explore variables**

We know that there are two variables in the dataset namely numerical and categorical. I will explore these variables in order to make my dataset good for working on it.

**Categorical variables**

Find categorical values:

Text

Description automatically generated

Checking missing values in categorical values:

Table

Description automatically generated

But it is early said to data is proper. I will check again because missing data may be another form like

“? “ and not ” NaN”.I only checked the data for the “NaN” form above. I will check another form :

Graphical user interface, text, application, Word

Description automatically generated

As you can see that there are “?’ missing values in my dataset so I replace them with the ‘NaN” common form for every categorical value. Graphical user interface, text, application, chat or text message

Description automatically generated

Using this code.

**Numerical variables**

Find numerical variables:

Graphical user interface, text, application

Description automatically generated

Checking missing values in numerical values:

Graphical user interface, text

Description automatically generated

It is ok with these numerical variables. There are no missing values.

**Announce the feature vector and the target variable**

Text

Description automatically generated with medium confidence

There are X that can be obtained from a dataset

**Split data into separate training and test data**

Graphical user interface, application, Word

Description automatically generated

**Feature Engineering**

As above mentioned my data contains missing values so I will remove these missing values using a feature engineering process namely transforming raw data into useful features that help us to understand our model and increase its predictive power. I will do it separately with numerical and categorical variables. I will work just with categorical values because numerical values do not include missing values.

Check data types in X\_train:

Table

Description automatically generated

Display categorical values:

Graphical user interface, text, application, email

Description automatically generated

In this case, my task is to remove “NaN” from X\_train and X\_test data in order to make my classification properly. And below, we can see that after removing my dataset is without “ NaN” and “?”.I will output my result after cleaning.

A picture containing text

Description automatically generated Table

Description automatically generated with medium confidence

We can see that there are no missing values inX\_train and X\_test.

**Encode categorical values.**

Many machine learning algorithms are not able to use non-numeric data. While many features we might use, such as a person's age, or height, are numeric there are many that are not. Usually, these features are represented by strings, and we need some way of transforming them into numbers before using sci-kit-learns algorithms. The different ways of doing this are called encodings. So that, I will implement the encoding process below and also show the outputs.

Chart, scatter chart

Description automatically generated

A picture containing table

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

A picture containing table

Description automatically generated

We can see that from the initial 14 columns, we now have 108 columns. We now have training and testing set ready for model building. Before that, we should map all the feature variables onto the same scale. It is called feature scaling. I will do it as follows.

Table

Description automatically generated

We now have X\_train dataset ready to be fed into the classifier.

**Experimental Result**

**Model training**

**Train a Gaussian Naive Bayes classifier on the training set:**

Graphical user interface, text, application, email

Description automatically generated Graphical user interface, text, application

Description automatically generated

Here, I used the Gaussian Naïve Bayes model for the classification task.

As you can see the model accuracy score is 0.8062 and whicht is a good result.

**Train a** Bernoulli **Naive Bayes classifier on the training set:**

Graphical user interface, text, application, email

Description automatically generated

Here, I used the Bernoulli Naïve Bayes model for the classification task.

As you can see that model accuracy score is 0.8007 and it is a good result.

**Train a** Multinomial **Naive Bayes classifier on the training set:**

Graphical user interface, text, application, email

Description automatically generated

Here, I used the Multinomial Naïve Bayes model for the classification task.

As you can see the model accuracy score is 0.6425 and it is not a good result as other models.

Because this model can not accept negative values but there are negative values in my dataset. Then, I used scaler = MinMaxScaler() instead of scaler= RobustScaler() in order to classify data in this model but the result is not good I think.

Text

Description automatically generated

Compare the train-set and test-set accuracy

Now, I will compare the train-set and test-set accuracy to check for overfitting.

Graphical user interface, text, application

Description automatically generated

Check for overfitting and underfitting

Graphical user interface, text, application

Description automatically generated

The training-set accuracy score is 0.8037 while the test-set accuracy to be 0.8084. These two values are quite comparable. So, there is no sign of overfitting.

### **Compare model accuracy with null accuracy**

So, the model accuracy is 0.8062,0.8007, and 0.6425. But, we cannot say that our model is very good based on the above accuracy. We must compare it with the **null accuracy**. Null accuracy is the accuracy that could be achieved by always predicting the most frequent class.

So, we should first check the class distribution in the test set.

Text

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

We can see that our model accuracy score is 0.8062,0.8007, and 0.6425 but the null accuracy score is 0.7668. So, we can conclude that our Gaussian Naive Bayes Classification model is doing a very good job of predicting the class labels.

Now, based on the above analysis we can conclude that our classification model accuracy is very good. Our model is doing a very good job in terms of predicting the class labels. But, only, in the third scenario, the result is bad and why it is the low result I said above.

**Confusion matrix**

A confusion matrix is a tool for summarizing the performance of a classification algorithm. A confusion matrix will give us a clear picture of classification model performance and the types of errors produced by the model. It gives us a summary of correct and incorrect predictions broken down by each category. The summary is represented in a tabular form.

Four types of outcomes are possible while evaluating a classification model performance. These four outcomes are described below:-

**True Positives (TP)** – True Positives occur when we predict an observation belongs to a certain class and the observation actually belongs to that class.

**True Negatives (TN)** – True Negatives occur when we predict an observation does not belong to a certain class and the observation actually does not belong to that class.

**False Positives (FP)** – False Positives occur when we predict an observation belongs to a certain class but the observation actually does not belong to that class. This type of error is called **Type I error.**

**False Negatives (FN)** – False Negatives occur when we predict an observation does not belong to a certain class but the observation actually belongs to that class. This is a very serious error and it is called **Type II error.**

Now, we will use this tool for checking the performance of classification models namely Gaussian, Bernoulli, and Multinomial Naïve Bayes Classifier.

**Gaussian Classifier**

Graphical user interface

Description automatically generated with low confidence

Chart, treemap chart

Description automatically generated

**Bernoulli Classifier**

A picture containing graphical user interface

Description automatically generated

Chart, treemap chart

Description automatically generated

# ****Results and conclusion****

1. In this project, I build a Gaussian, Bernoulli, and Multinomial Naïve Bayes Classifier model to predict whether a person makes over 50K a year. The model yields a very good performance except for one as indicated by the model accuracy which was found to be 0.8062,0.8007, and 0.6425 respectively.
2. The training-set accuracy score is 0.8037 while the test-set accuracy is 0.8084. These two values are quite comparable. So, there is no sign of overfitting.
3. I have compared the model accuracy score which is 0.8062,0.8007, and 0.6425 with the null accuracy score which is0.7668. So, we can conclude that our Gaussian, Bernoulli Naïve Bayes classifier model is doing a very good job of predicting the class labels and the Multinomial classifier is not doing a good job of predicting due to the negative values in the dataset.
4. I also indicated an exact number of error predictions and predictions in using the heatmap graph function which makes it easy to see the number distribution. But I showed only it for only two graphs because the third model result is bad so I found showing it unreasonable. As you can see that Multinomial model is not proper for my dataset, in this case, it would be better to use Gaussian and Bernoulli models.