Chang'an University

Report

On

"Naïve Bayes Classifier Algorithm"

25 June, 2022

Submitted By:
Manik Debnath |2021124913
Computer Science and Technology
School of Information Engineering
Chang'an University
Data Mining

Submitted To:
Prof. Ying Li & Prof. Yuande Jiang
School of Information Engineering
Chang'an University

Table of Contents

1. Introduction to Naive Bayes Classifier Algorithm

Every machine learning engineer works with statistics and data analysis while building any model and a statistician makes no sense until he knows Bayes theorem. We will be discussing an algorithm which is based on Bayes theorem and is one of the most adopted algorithms when it comes to text mining. We are talking about Naive Bayes. Naive Bayes is one such algorithm which is supervised and depends on the probabilities of the events to occur.

In machine learning, Naive Bayes classification is a straightforward and powerful algorithm for the classification task. Naive Bayes classification is based on applying Bayes' theorem with strong independence assumption between the features. Naive Bayes classification produces good results when we use it for textual data analysis such as Natural Language Processing.

Naive Bayes models are also known as **simple Bayes or independent Bayes**. All these names refer to the application of Bayes' theorem in the classifier's decision rule. Naive Bayes classifier applies the Bayes' theorem in practice. This classifier brings the power of Bayes' theorem to machine learning.

Naive Bayes is considered has naive because of the independence of one attribute in a class with respect to others. In simple words, it does not hold any dependence with two attributes having the same class. Before explaining further, an example is always a better choice. Let us take a few apples which are red and have an average diameter of 5 inches. Now if a Naive Bayes predicts the probability of a red fruit having 4 inches being an apple, then it holds no dependency between color and diameter. Before trying to deep dive in Naive Bayes, it is essential to learn the fundamentals.

What is Naive Bayes?

Naive Bayes is a simple learning algorithm that utilizes Bayes rule together with a strong assumption that the attributes are conditionally independent, given the class. While this independence assumption is often violated in practice, Naive Bayes nonetheless often delivers competitive classification accuracy.

Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London,* **53:370-418**

Why it’s called Naive Bayes?

The Naive Bayes algorithm is comprised of two words Naive and Bayes, Which can be described as:

- ✓ **Naive:** It is called Naive because it assumes class conditional independence. The effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is made to reduce computational costs and hence is considered Naive. Such as if the fruit is identified on the bases of colour, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

- ✓ **Bayes:** It is called Bayes because it depends on the principle of Bayes' Theorem.

What is Bayes’ Theorem?

Bayes’ Theorem is a statement from probability theory that allows for the calculation of certain conditional probabilities. Conditional probabilities are those probabilities that reflect the influence of one event on the probability of another event. The term generally used in Bayes’ theorem are prior probability and posterior probability. The prior probability of a hypothesis or event is the original probability obtained before any additional information is obtained. The posterior probability is the revised probability of the hypothesis using some additional information or evidence obtained.

- ❖ Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

- ❖ The **formula for Bayes' theorem** is given as:

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

Where,

$\boldsymbol{P(A)}$ is the prior probability of A
$\boldsymbol{P(B)}$ is the marginal probability of B
$\boldsymbol{P(A|B)}$ is the conditional probability of A given B
$\boldsymbol{P(B|A)}$ is the conditional probability of B given A

- **P(A|B) is Posterior probability**: Probability of hypothesis A on the observed event B.
- **P(B|A) is Likelihood probability**: Probability of the evidence given that the probability of a hypothesis is true.
- **P(A) is Prior Probability**: Probability of hypothesis before observing the evidence.
- **P(B) is Marginal Probability**: Probability of Evidence.

Since the denominator P(B) is the probability of the evidence without any knowledge of the event A, and since the hypothesis A can be true or false, Bayes' theorem can also be written as:

$$\boldsymbol{P(A|B) = \frac{P(B|A)P(A)}{P(B) * P(A) + P(B|\neg A) * P(\neg A)}}$$

Where,

P(¬A) is the probability of A being false
P(B|¬A) is the probability of B given A is false

Example of Bayes Theorem

- ✓ Given:

  -A doctor knows that Cold causes **fever 50%** of the time

  -Prior probability of any patient having cold is **1/50,000**

  -Prior probability of any patient having fever is **1/20**

- ✓ If a patient has fever, what's the probability he/she has cold?

$$P(C|F) = \frac{P(F|C)P(C)}{P(C)} = \frac{0.5 \times (\frac{1}{50000})}{1/20} = 0.0002$$

2. What is Naive Bayes Classification?

The Naive Bayes classification algorithm is a probabilistic classifier, and it belongs to Supervised Learning. It is based on probability models that incorporate strong independence assumptions. The independence assumptions often do not have an impact on reality. Therefore, they are considered naive.

Another assumption made by the Naive Bayes classifier is that all the predictors have an equal effect on the outcome. The Naive Bayes classification has the following different types:

- ✓ The **Multinomial Naive Bayes** method is a common Bayesian learning approach in natural language processing. Using the Bayes theorem, the program estimates the tag of a text, such as an email or a newspaper piece. It assesses the likelihood of each tag for a given sample and returns the tag with the highest possibility.

- ✓ The **Bernoulli Naive Bayes** is a part of the family of Naive Bayes. It only takes binary values. There may be multiple features, but each is assumed to be a binary-valued (Bernoulli, Boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors.

- ✓ The **Gaussian Naive Bayes** is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. To build a simple model using Gaussian Naive Bayes, we assume the data is characterized by a Gaussian distribution with no covariance (independent dimensions) between the parameters. This model may be fit simply by calculating the mean and standard deviation of the points within each label.

Naive Bayes classifier makes two fundamental assumptions on the observations.

- ❖ The target classes are **independent** to each other. Consider a rainy day with strong winds and high humidity. These two features, wind and humidity, would be treated as independent by a Naive classifier. That is to say, each feature would impose its own probabilities on the outcome, such as rain in this case.

- ❖ Prior probabilities for the target classes are **equal**. That is, before calculating the posterior probability of each class, the classifier will assign each target class the same prior probability.

2.1 When to use Naive Bayes Classifier?

Naive Bayes classifiers tend to perform especially well in any of the following situations:

- ✓ When the naive assumptions actually match the data.
- ✓ For very well-separated categories, when model complexity is less important.
- ✓ And for very high-dimensional data, when model complexity is again less important.

The last two points appear unrelated, but they are related to each other. As a dataset's dimension grows, it becomes considerably less likely that any two points will be discovered near together. This means that clusters in high dimensions tend to be more separated than clusters in low dimensions.

The Naive Bayes classifier has the following advantages:

- ❖ Naive Bayes classification is extremely fast for both training and prediction.
- ❖ It provides straightforward probabilistic prediction.
- ❖ Naive Bayes has very low computation cost.
- ❖ It can efficiently work on a large dataset.
- ❖ It performs well in case of discrete response variable compared to the continuous variable.
- ❖ It can be used with multiple class prediction problems.
- ❖ It also performs well in the case of text analytics problems.
- ❖ When the assumption of independence holds, a Naive Bayes classifier performs better compared to other models like **Logistic Regression**.

2.2 Real-life applications using Naive Bayes Classification

The Naive Bayes algorithm offers plenty of advantages to its users. That's why it has a lot of applications in various industries, including Health, Technology, Environment, etc. Here are a few of the applications of Naive Bayes classification:

- ✓ It is used in **text classification**. For example, News on the web is rapidly growing where each news site has its own different layout and categorization for grouping news. In order to achieve better classification result, we apply the naive Bayes classifier for classification of news contents based on news code.

- ✓ Another application of Naive bayes classification is **Spam filtering**. It typically uses a bag of words features to identify spam e-mail. Naive Bayes classifiers work by correlating the use of tokens (typically words, or sometimes other things), with a spam and non-spam e-mails and then using Bayes' theorem to calculate a probability that an email is or is not spam.

- ✓ One of the advantages of Naive Bayes Classifier is that it takes all the available information to explain the decision. When dealing with **medical data**, Naive Bayes classifier takes into account evidence from many attributes to make the final prediction and provides transparent explanations of its decisions. That is why it has many applications in health sector as well.

- ✓ **Weather prediction** has been a challenging problem in the meteorological department for years. Even after the technological and scientific advancement, the accuracy in prediction of weather has never been sufficient. However, Naive Bayes classifiers give high accuracy result when predicting the weather conditions.

- ✓ Its assumption of feature independence, and its effectiveness in solving multi-class problems, makes it perfect for performing **Sentiment Analysis**. Sentiment Analysis refers to the identification of positive or negative sentiments of a target group.

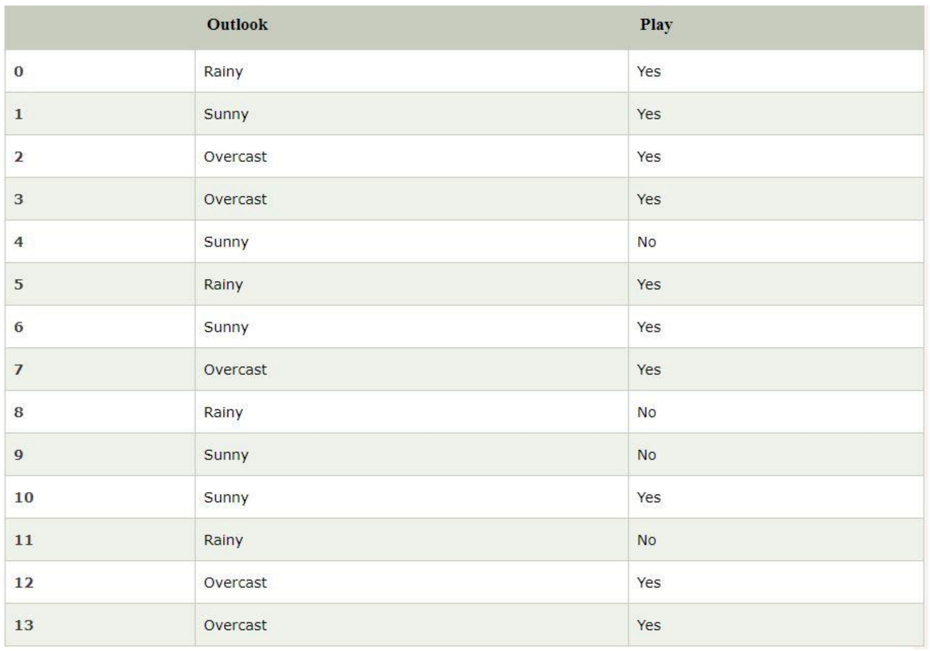
- ✓ Collaborative Filtering and the Naive Bayes algorithm work together to build **recommendation systems**. These systems use data mining and Machine Learning to predict if the user would like a particular resource or not.

2.3 Mathematical calculations behind Naive Bayes Classification

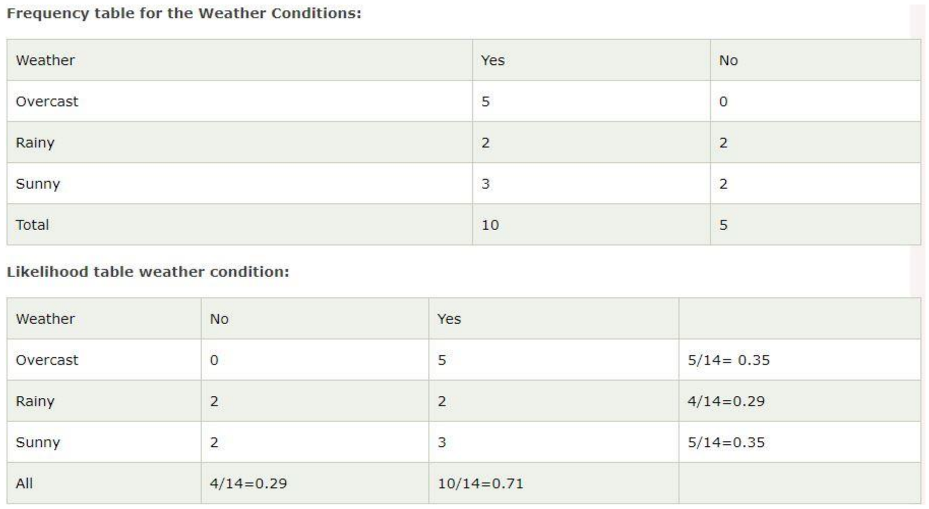
Let us now calculate the event occurring/classifying using the Naive Bayes classification method by taking a simple example. Given an example of weather conditions and playing sports, we need to calculate the probability of playing sports and not playing sports. And classify whether players will play or not depending on the weather condition.

If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset



	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes



Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)
P(Sunny|Yes)= 3/10= 0.3
P(Sunny)= 0.35 , P(Yes)=0.71
So, P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**
P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)
P(Sunny|NO)= 2/4=0.5
P(No)= 0.29 , P(Sunny)= 0.35
So, P(No|Sunny)= 0.5*0.29/0.35 = **0.41**
So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

3. Literature Review of Naive Bayes Classifier

The Naive Bayes classifiers are based on the Bayes Theorem and are therefore sometimes referred to as Bayesian classifiers. Bayesian classifiers work on the idea that a class can possibly predict the features of the members of that class because of the commonality of the feature values in the class. If a class is known to an agent, prediction of feature values is easy. However, if the class itself is not known and the only source of knowledge are some of the values of the features, then the Bayes rule can be applied to predict the corresponding class. A learning agent can therefore build a probabilistic model of the available features and use the same for predicting the classification.

Borkar and Deshmukh proposed using Naive Bayes classifier for detection of Swine Flu disease. The process starts with finding probability for each attribute of Swine flu against all output. The probabilities of each attribute are then multiplied. Selecting the maximum probability from all the probabilities, the attributes belong to the class variable with maximum value. The promising results of the proposed scheme can be used for investigating further the Swine flu disease in patients using Information technology.

Patil worked in the direction of diagnosing whether a patient with his given information regarding age, sex, blood pressure, blood sugar, chest pain, ECG reports etc. can have a heart disease later in life or not. The experiments involve taking the parameters of the medical tests as inputs. The proposal is effective enough in being used by nurses and medical students for training purposes. The data mining technique used is Naive Bayes Classification for the development of Decision Support System in Heart Disease Prediction System (HDPS). The performance of the proposal is further improved using a smoothing operation. The implementation of HDPS is done through a MATLAB application able to detect and extract hidden knowledge related to heart diseases from a historical heart disease database.

Kharya et al proposed detecting in patients the chances of having Breast Cancer later in life. Severity in Breast Cancer is necessary seeing it becoming the second most cause of death among women. A Graphical User Interface (GUI) is designed for entering the patient‘s record for the prediction. The records are mined through the data repository. Naive Bayes classifier, being simple and efficient is chosen for the prediction. The results obtained by the Naive Bayes classifier are accurate, have low computational effort and fast.

Stephanie J. Hickey proposed using Naive Bayes Classifier for public health domain combined with greedy feature selection. The input was a public health dataset and the objective behind the proposal was to identify one or several attributes that best predict a selected target attribute without the need for searching the input space exhaustively. The proposal achieved its goal with increase in accuracy of classification. The target attributes were related to diagnosis or procedure codes.

Ambica et al proposed using Naive Bayes for an efficient decision support system for Diabetes disease. The proposed classification system was divided into two steps. The first step includes analysis of how optimal the dataset is and accordingly extraction of the optimal feature set from the training data is done. The second step forms the new dataset as the optimal training dataset and the proposed classification scheme is now applied on the optimal feature set. The mismatched and unavailable features from the training and testing datasets are ignored and the dataset attributes are used for the calculation of posterior probability. The proposed procedure therefore shows elimination of unavailable features and document wise filtering.

Gamollo et al proposed sentiment analysis on Spanish tweets. The proposal was more inclined towards detecting sentiments on Twitter, a very popular microblogging service consisting of millions of tweets throughout the day from celebrities and the media and people following these celebrities. The notion of considering polarity for the classification is used. A total of six sentiment categories are detected in the proposed system with a resulting accuracy of 67%. Apart from the positive and negative tweets, polarity levels have been extended to strong, average, weak or neutral tweets by setting thresholds for experimental purposes. For detecting whether the analyzed text has some polarity or not, searching of polarity words is done within the same.
Gamallo and Garcia implemented two Naive Bayes Classifiers-Baseline and Binary for the task of Sentiment Analysis on English tweets. English tweets are considered difficult to classify because of them based on the uncertain human subjectivity and being too small to be linguistically analyzed. The SemEval-2014 classifies a given message into positive, negative or neutral sentiments. For the message containing both the sentiments, majority of the two sentiments is considered. The work of the proposed Baseline Naive Bayes Classifier is to classify from the training corpus, the positive, negative and neutral categories thereby introducing no modifications in the classifier. The Binary classifier uses the polarity lexicon and further simplifies the corpus, categorizing the tweets as negative or positive. In other words, a Boolean classifier was trained to distinguish between the polarity-constrained tweets. For the neutral tweets, if at least one word was found in the polarity lexicon, then the tweet was considered to be of some degree of that polarity. If no such word was found, then the tweet was considered neutral. The binary classifier used gave more than 80% precision in the trained corpus with the positive and negative categories. Their proposed scheme is being used by a company specializing in Natural Language Technology, Cilenis S. L. for four languages; English, Spanish, Portuguese and Galician.
Another similar classification was done on SemEval-2015 by **Talbot et al** authors designed a sentiment classification system employing a supervised text classification approach based on constraints. The two major contributions include enhancement of quality related to lexical information by introduction of various preprocessing steps for tweeting data and use of Naive Bayes Classifier for the detection of tweet sentiments with the training data provided by the task organizers. The positive and negative words in the external human-generated lists are often used for the classification. A F-Score of 59.26 is observed by their proposal on an official test dataset.
Kim et al modified the Naive Bayes classifier for an e-catalogue classification of online business transactions. The increasing product information and its heterogeneous nature have correspondingly increased the complexity in the classification task. The authors addressed this problem by extension of the Naive Bayes classifier for utilizing the structural characteristics of the e-catalogs. Use of such appropriate characteristics for the purpose of classification improves the accuracy of classification. Instead of taking each word position as an attribute as done in the conventional Naive Bayes classifier for text classification, sets of attribute value pairs are considered for classification as per the underlying objective. Every individual attribute is assigned a weight according to the importance they hold in the classification. Before the weights are assigned, normalization of the attributes is done. The accuracy is further improved by the use of 'category name' attribute that is available in the training set. The proposed classification is observed more accurate even in the real-world scenario consisting of noisy attributes.
Salperwyck et al used Naive Bayes for data streams. The already proposed methods of improving the Naive Bayes by the use of a variable selection method or by weighting of the explanatory variables are used for off-line learning. Plus, they needed to store all the data available in memory or requiring reading each example more than once. These limitations were addressed by the authors

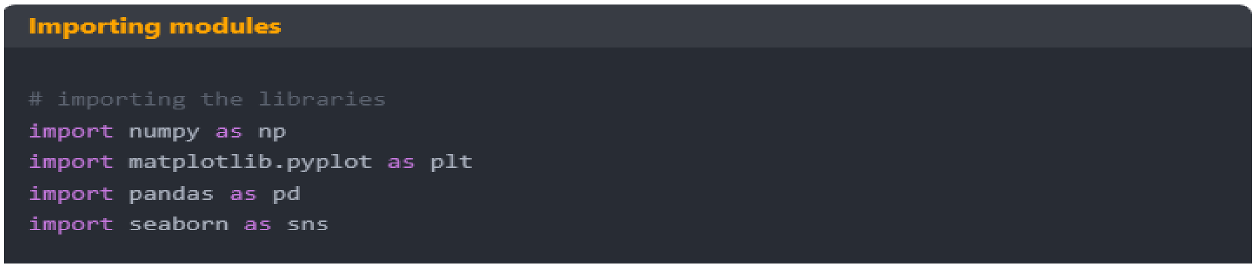
method based on a graphical model where scholastic estimation was used for the computation of weights to be imposed on the input variables. The proposed method was incremental and the results when compared with the conventional Naive Bayes classifier were better.
Rabenoro et al focused on the application papers with expert knowledge, both simple and with low level parameter scores and build an interpretable classifier based on this knowledge and a training set. However, the estimation of scores requires other parameters and thresholds which are seldom available through the experts. Combination of these scores for achieving rates acceptable for classification is another limitation. The authors propose selecting features using the filter mRMR approach with the idea to keep only the useful features for consideration from the large set of redundant binary features obtained from the data under study. The advantages from the proposed solution include easy finding of required parameters and thresholds with the reasonable features. The decision by the interpretable Naive Bayes classifier is also reduced.
Jing et al aimed at improving the conventional Naive Bayes classifier and proposed a Semantic Naive Bayes classifier (SNBC) through the incorporation of document level semantic information. The semantic information is captured from each document by the proposed semantic feature extraction and modeling algorithms. The semantic feature extraction is further done in two steps. In the first step, each word in the document is transformed into a semantic vector through Log-Bilinear document modeling (LBDM). In the second step, the formed word vectors space can extract the semantic feature sets for each document through the use of Principal Component Analysis (PCA). The semantic features of the training document are then used for the semantic modeling. This semantic model is then combined with the conventional Naive Bayes classifier in the testing phase of the proposed SNBC for document classification. The proposed SNBC shows outperformed results when compared to the original Naive Bayes Classifier.

4. Naive Bayes Classification Implementation

Let's implement the Naive Bayes Classification using sklearn module. We will use a sample data set, which you can download here. This is a simple data set containing only two output classes (binary classification dataset). The input values contain the age and salary of a person, and the output class contains whether the person purchased the product or not.

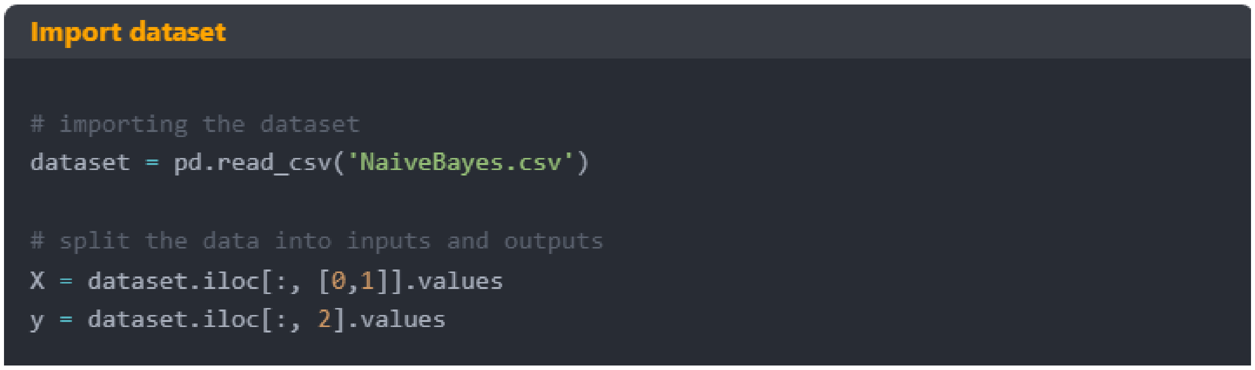
4.1 Importing and defining Binary Dataset for Naive Bayes Classification

We will use the following modules to implement Naive Bayes classification using Python. You can install them using the pip command on your system.


Importing modules

```
# importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

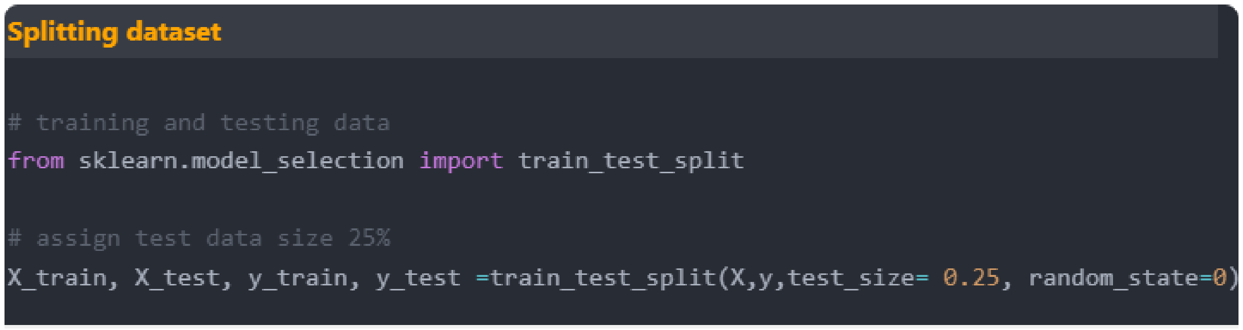
Once we have imported all the required modules, the next step is to import the data set and split the data sets into inputs and outputs. We can get access to the data set from this link.



Import dataset

```
# importing the dataset
dataset = pd.read_csv('NaiveBayes.csv')

# split the data into inputs and outputs
X = dataset.iloc[:, [0,1]].values
y = dataset.iloc[:, 2].values
```

The next step is to divide the input and output values into the training and testing part so that once the training of the model is complete, we can evaluate its performance using testing data.

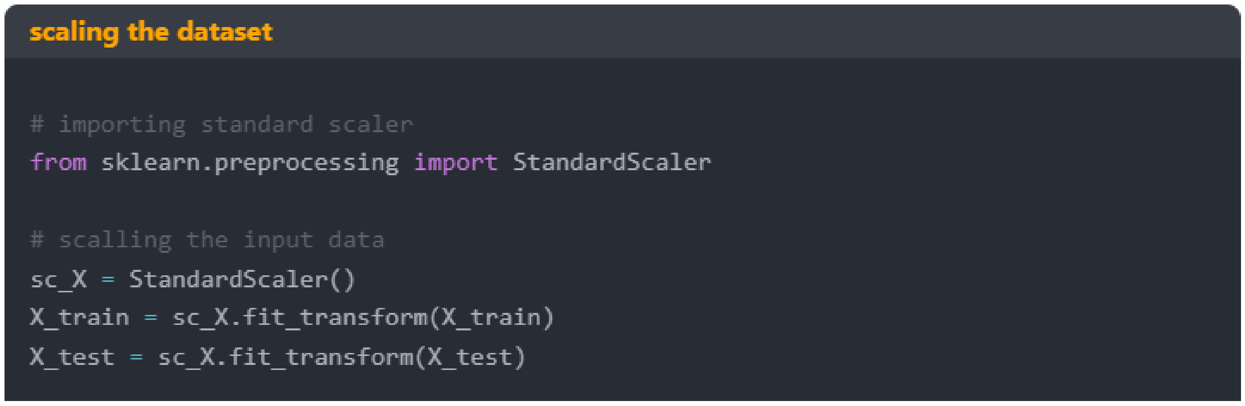


Splitting dataset

```
# training and testing data
from sklearn.model_selection import train_test_split

# assign test data size 25%
X_train, X_test, y_train, y_test =train_test_split(X,y,test_size= 0.25, random_state=0)
```

We set test_size=0.25, which means 25% of the whole data set will be assigned to the testing part and the remaining 75% will be used for the model's training.

The next step is to scale our dataset to be ready to be used for the training.



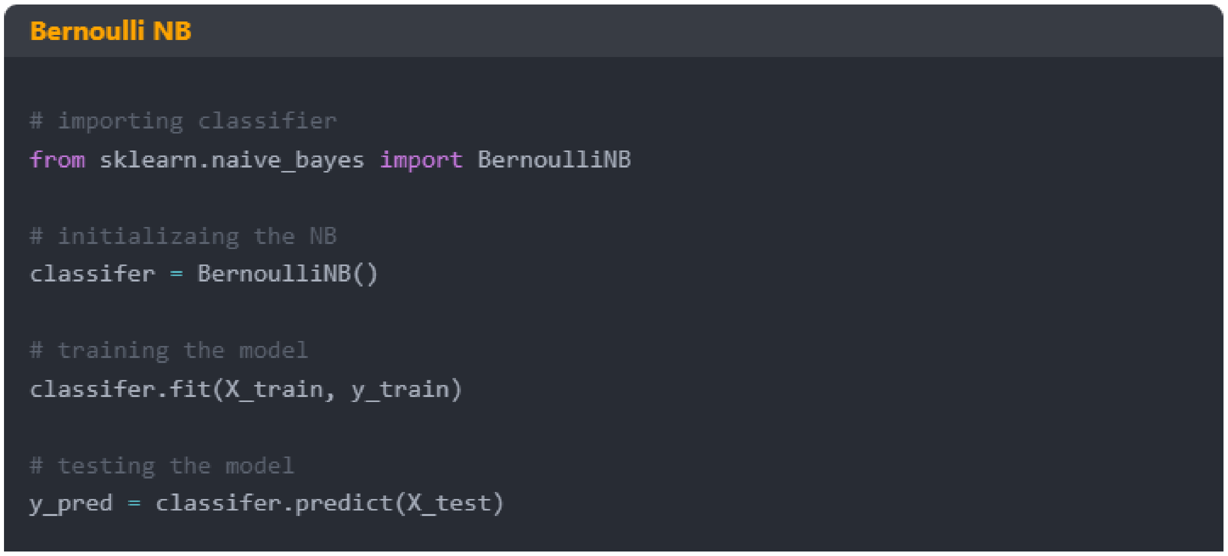
scaling the dataset

```
# importing standard scaler
from sklearn.preprocessing import StandardScaler

# scalling the input data
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

Note: scaling (or standardization) of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data.

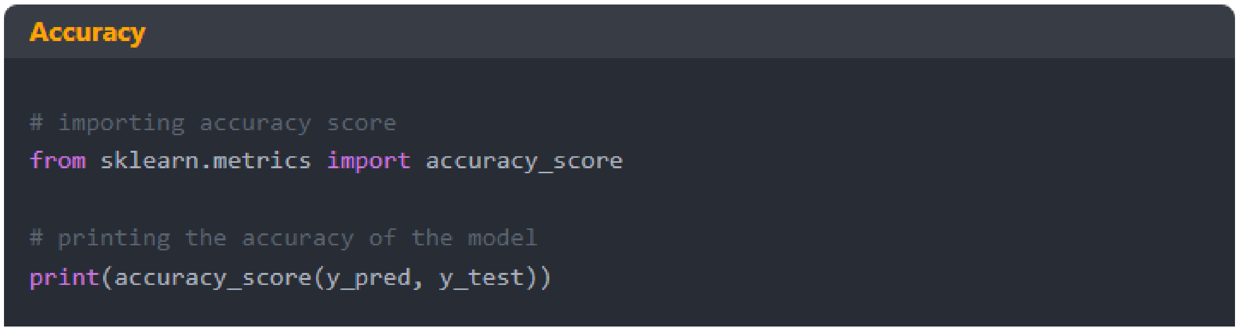
4.2 Training the model using Bernoulli Naive Bayes classifier

In this section of the article, we'll use the Bernoulli Naive Bayes classifier to train our model.


Bernoulli NB

```
# importing classifier
from sklearn.naive_bayes import BernoulliNB

# initializaing the NB
classifer = BernoulliNB()

# training the model
classifer.fit(X_train, y_train)

# testing the model
y_pred = classifer.predict(X_test)
```

Now let us check the accuracy of the predicted values using the Bernoulli Naive Bayes classifier.


Accuracy

```
# importing accuracy score
from sklearn.metrics import accuracy_score

# printing the accuracy of the model
print(accuracy_score(y_pred, y_test))
```

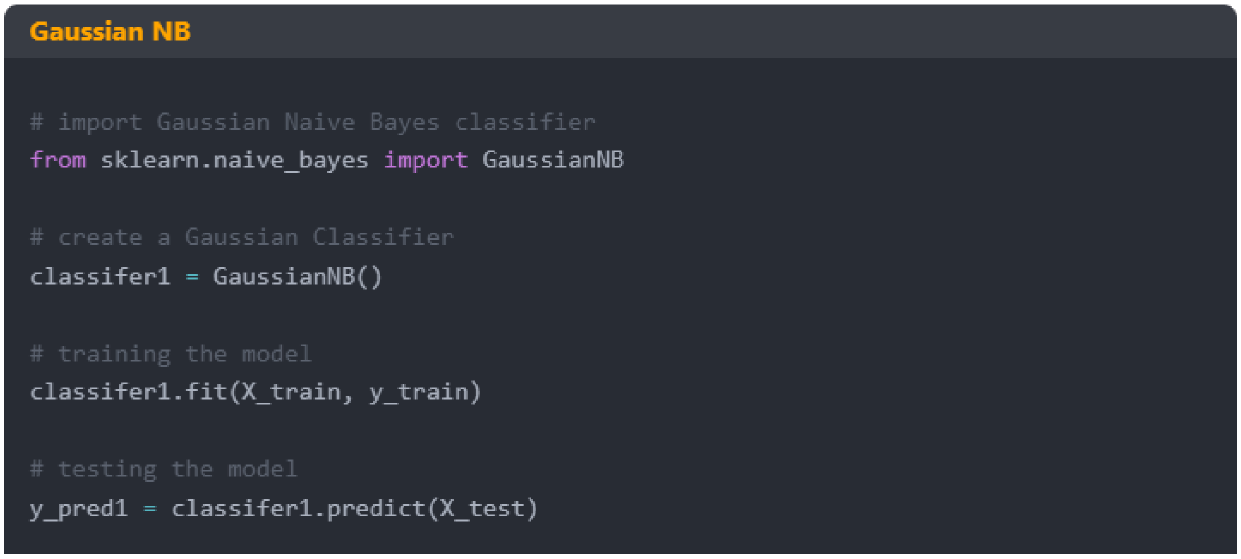
Output:

```
0.8
```

We got an **accuracy of 80%** when we trained our model using Bernoulli Naive Bayes classifier.

4.3 Training model using Gaussian Naive Bayes Classifier

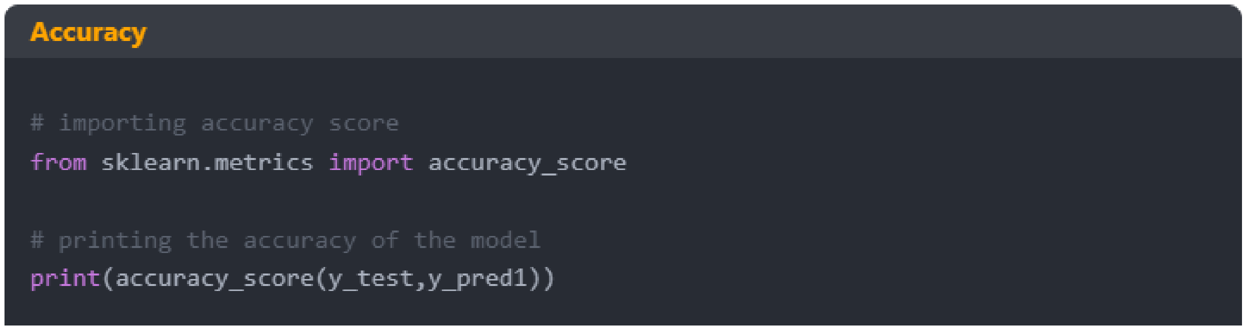
Now, let's train our model using the Gaussian Naive Bayes classifier (a type of Naive Bayes Classifier).



Gaussian NB

```
# import Gaussian Naive Bayes classifier
from sklearn.naive_bayes import GaussianNB

# create a Gaussian Classifier
classifer1 = GaussianNB()

# training the model
classifer1.fit(X_train, y_train)

# testing the model
y_pred1 = classifer1.predict(X_test)
```

Let's check the accuracy of our model:



Accuracy

```
# importing accuracy score
from sklearn.metrics import accuracy_score

# printing the accuracy of the model
print(accuracy_score(y_test,y_pred1))
```

Output:

```
0.91
```

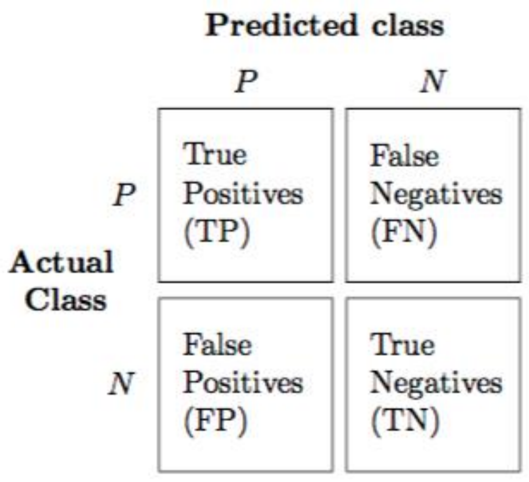
This time we got an **accuracy of 91%** when we trained the model on the same dataset.

4.4 Evaluating Naive Bayes Classification performance

In this section of the article, we'll demonstrate how to evaluate Naive Bayes classification model performance.

❖ Confusion Matrix for Binary classification

Confusion Matrix is also known as error matrix. It is a table layout that allows visualization of the performance of a classification algorithm. Each row of the matrix represents the instances in an actual class, while each column represents the instances in a predicted class, or vice versa.



The confusion matrix for binary classification is a **2×2 matrix** representing the actual and predicted values. It helps us calculate accuracy, precision, recall, and f1-score, which helps us evaluate the model's performance.

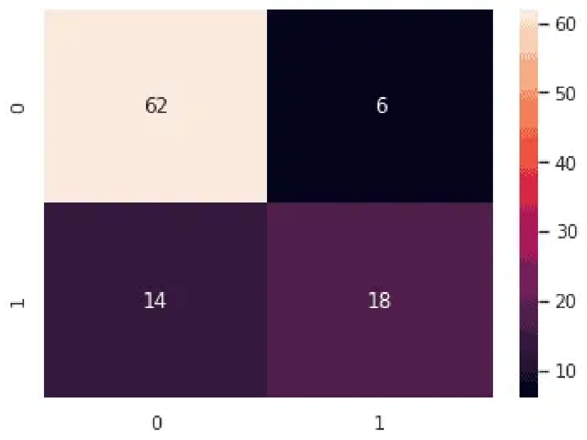
❖ Evaluation of Bernoulli Naive Bayes classifier

Let's evaluate our Bernoulli Naive Bayes model using a confusion matrix that will visually help us see the number of correct and incorrect classified classes. First of all, we'll visualize our model's results. The predicted values are stored in a variable named **`y_pred`**.

Confusion matrix

```
# importing the required modules
import seaborn as sns
from sklearn.metrics import confusion_matrix

# passing actual and predicted values
cm = confusion_matrix(y_test, y_pred)

# true write data values in each cell of the matrix
sns.heatmap(cm, annot=True)
plt.savefig('confusion.png')
```

Output:



The confusion matrix helps us know which class has been mispredicted.

We can also print the classification report, which will help us further evaluate our model's performance.

Classification report

```
# importing classification report
from sklearn.metrics import classification_report

# printing the report
print(classification_report(y_test, y_pred))
```

Output:

```
              precision    recall  f1-score   support

           0       0.82      0.91      0.86        68
           1       0.75      0.56      0.64        32

    accuracy                           0.80       100
   macro avg       0.78      0.74      0.75       100
weighted avg       0.79      0.80      0.79       100
```

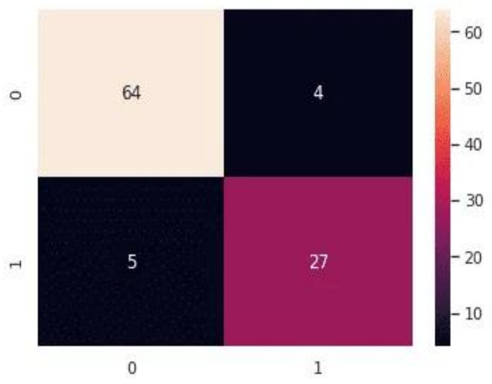
❖ Evaluation of Gaussian Naive Bayes Classifier

Let's evaluate the Gaussian Naive Bayes model. The predicted values are stored in a variable named `y_pred1`.

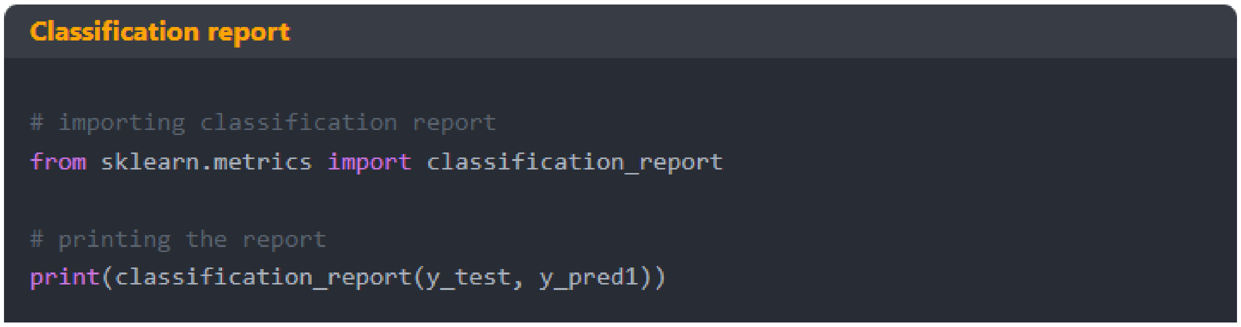
Confusion matrix

```
# importing the required modules
import seaborn as sns
from sklearn.metrics import confusion_matrix

# passing actual and predicted values
cm = confusion_matrix(y_test, y_pred1)

# true write data values in each cell of the matrix
sns.heatmap(cm,annot=True)
plt.savefig('confusion.png')
```

Output:



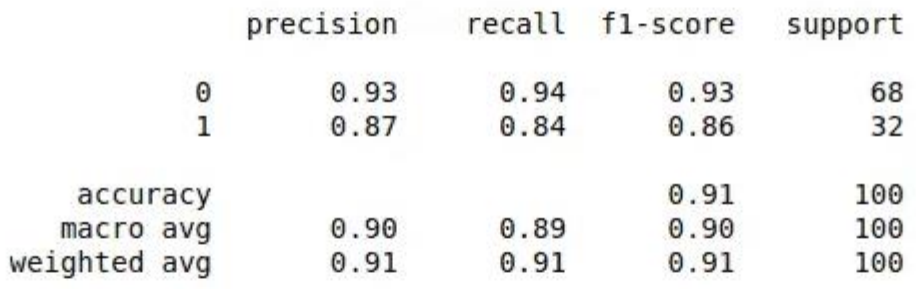
Note: the Gaussian naive Bayes classifier performed very well on this dataset, **as shown in the confusion matrix**. Let us now print out the classification report as well,



Classification report

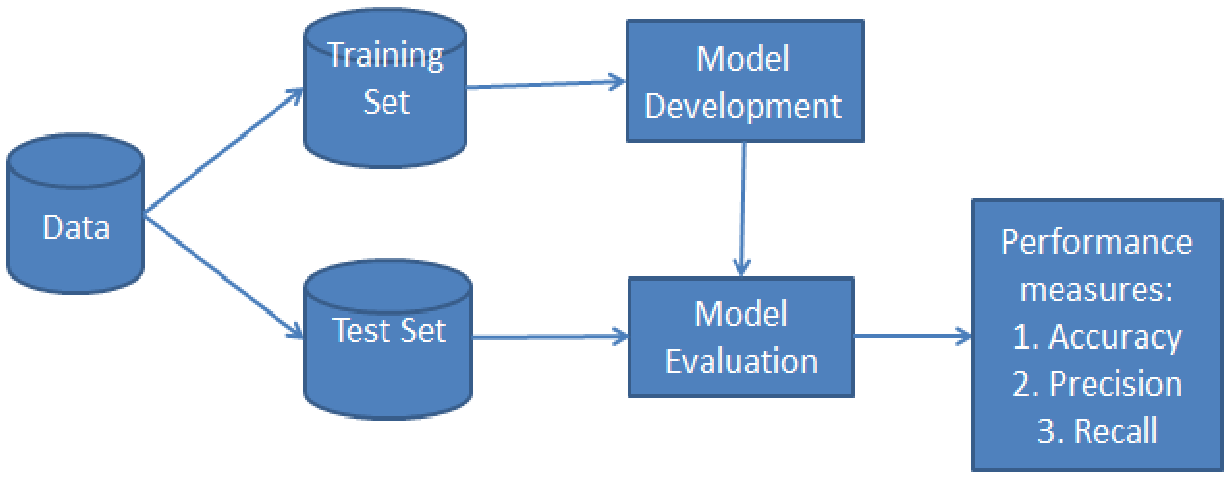
```
# importing classification report
from sklearn.metrics import classification_report

# printing the report
print(classification_report(y_test, y_pred1))
```

Output:

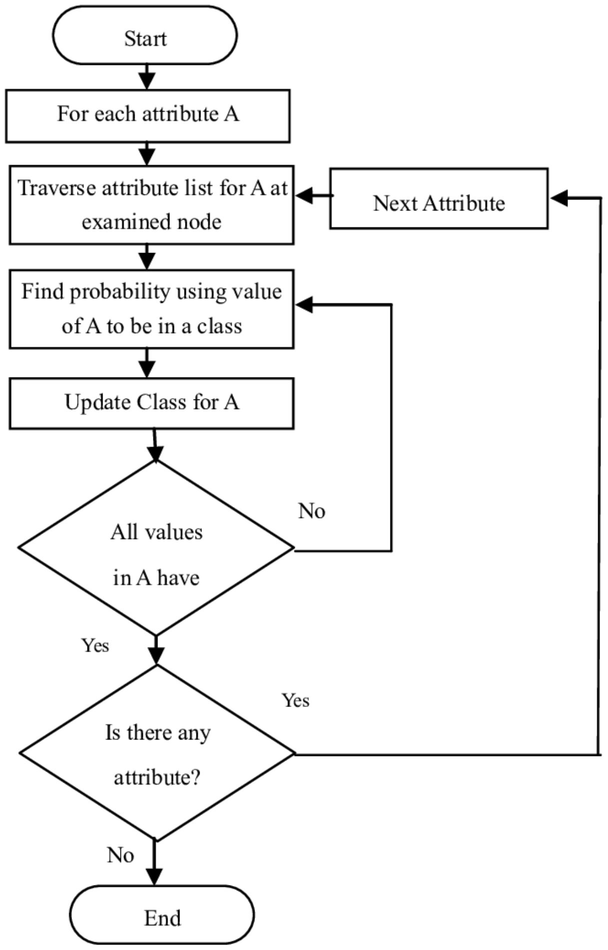


```
              precision    recall  f1-score   support

           0       0.93      0.94      0.93        68
           1       0.87      0.84      0.86        32

    accuracy                           0.91       100
   macro avg       0.90      0.89      0.90       100
weighted avg       0.91      0.91      0.91       100
```

❖ Working Diagram

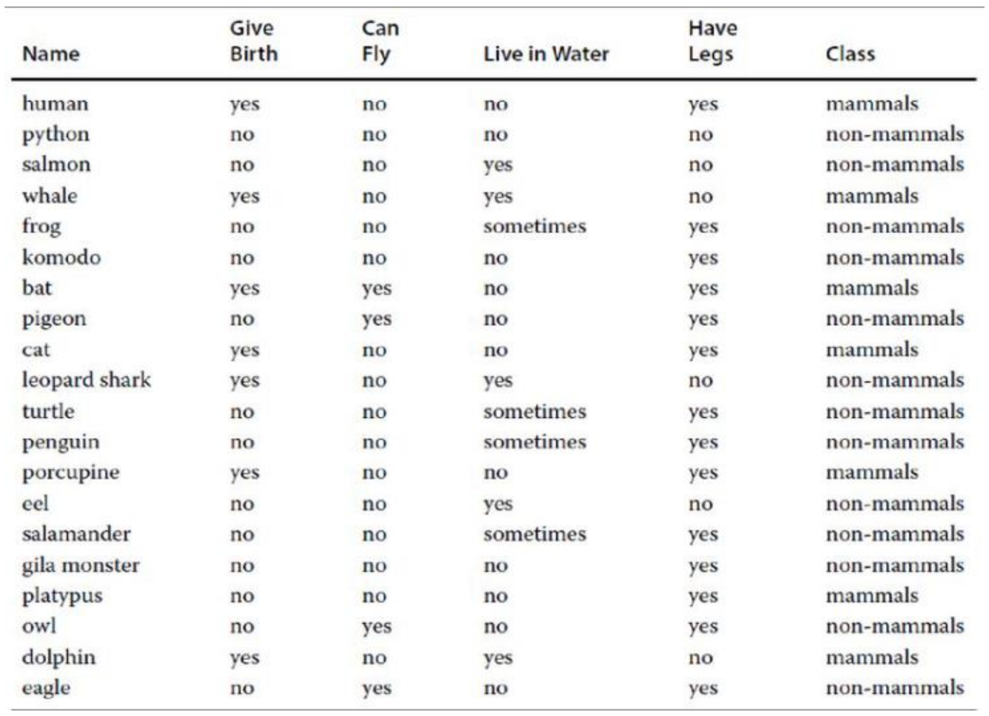


❖ Flowchart Diagram



❖ Example of Naive Bayes Classifier

✓ **What is the highest possibility of the class mammals or non-mammals?**

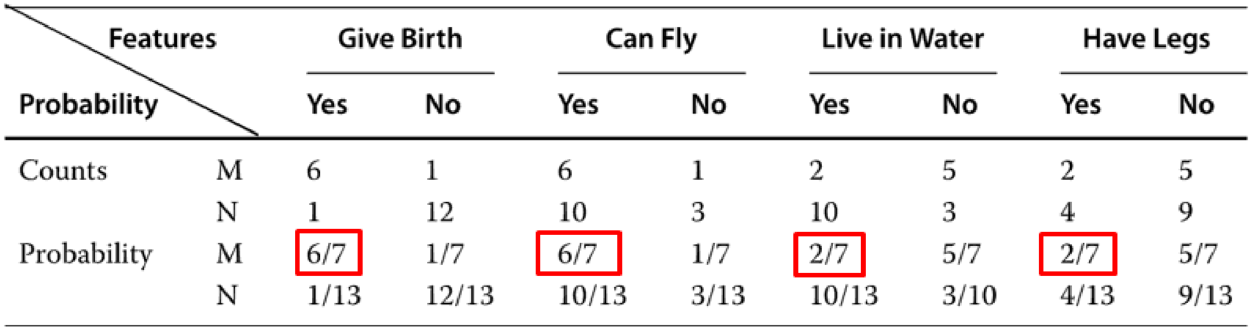


Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

The prior probabilities

$$P(M) = \frac{7}{20} \quad \text{and} \quad P(N) = \frac{13}{20}$$

The posterior probabilities $P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$



Features / Probability		Give Birth		Can Fly		Live in Water		Have Legs	
		Yes	No	Yes	No	Yes	No	Yes	No
Counts	M	6	1	6	1	2	5	2	5
	N	1	12	10	3	10	3	4	9
Probability	M	$\boxed{6/7}$	1/7	$\boxed{6/7}$	1/7	$\boxed{2/7}$	5/7	$\boxed{2/7}$	5/7
	N	1/13	12/13	10/13	3/13	10/13	3/10	4/13	9/13

The posterior probabilities $P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$

Features / Probability		Give Birth		Can Fly		Live in Water		Have Legs	
		Yes	No	Yes	No	Yes	No	Yes	No
Counts	M	6	1	6	1	2	5	2	5
	N	1	12	10	3	10	3	4	9
Probability	M	6/7	1/7	6/7	1/7	2/7	5/7	2/7	5/7
	N	$\boxed{1/13}$	12/13	$\boxed{10/13}$	3/13	$\boxed{10/13}$	3/10	$\boxed{4/13}$	9/13

The highest possibility of being mammals are = $P(A|M)P(M) = 0.06 \times \frac{7}{20}$ = 0.021
The highest possibility of being non-mammals are = $P(A|N)P(N) = 0.0042 \times \frac{13}{20}$ = 0.0027

$$P(A|M)P(M) > P(A|N)P(N)$$

Mammals

5. Project Implementation

Our project tittle is **Credit Card Fraud Detection using Naive Bayes Classifier Algorithm**. The goal of this project is to create a classifier which classify bank transactions into **legitimate** and **fraudulent**.

❖ Implementation Process

We implement a Naive Bayes Algorithm using **Jupyter Notebook** for our project. So, for this we use the "user data" as a dataset. Therefore, we can easily compare the Naive Bayes model with the other models.

❖ Steps to Implement:

1) Import Libraries and Data Set
2) Visualize Data
3) Create Training and Testing Set
4) Train Model
5) Evaluate Model

❖ Overview of the Project:

Credit card fraud is increasing considerably with the development of modern technology and the global superhighways of communication. Credit card fraud costs consumers and the financial company billions of dollars annually, and fraudsters continuously try to find new rules and tactics to commit illegal actions. Thus, fraud detection systems have become essential for banks and financial institution, to minimize their losses.

About Dataset:

The datasets contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

Dataset available this link: https://www.kaggle.com/mlg-ulb/creditcardfraud

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example: dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

❖ Step 1 - Import Libraries and Data Set:

In addition to the **scikit-learn library** which we use to perform the classification analysis, we also use several other packages and modules:

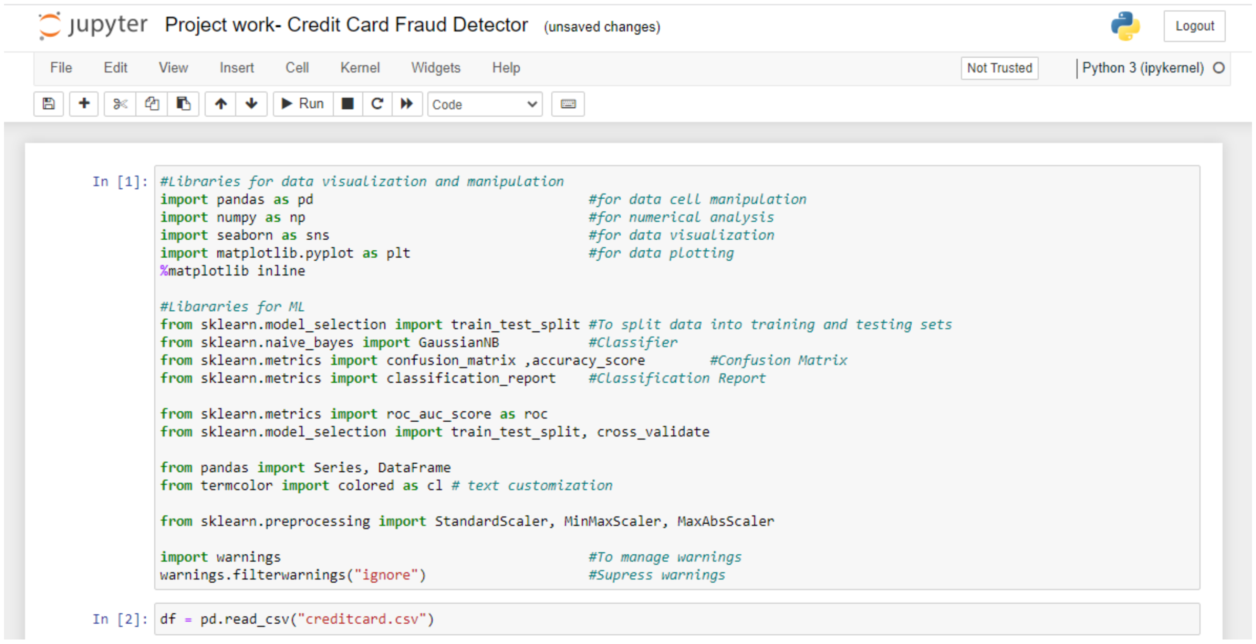
Pandas: Used for data structures and operations for manipulating numerical tables

Numpy: Used for numerical analysis

Matplotlib.pyplot: Used for plotting data

Seaborn: Used for data visualization (used on top of matplotlib library)

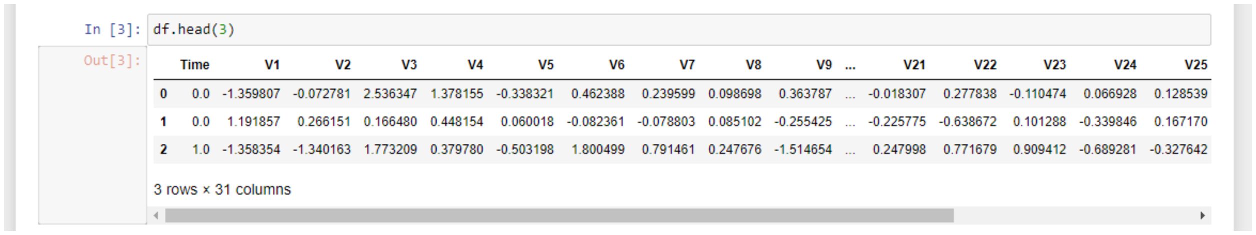
Sklearn: Used in our kernel to split the set into a training and a testing set, to create the model, and to visualize the results.


jupyter Project work- Credit Card Fraud Detector (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [1]: #Libraries for data visualization and manipulation
        import pandas as pd                                    #for data cell manipulation
        import numpy as np                                     #for numerical analysis
        import seaborn as sns                                  #for data visualization
        import matplotlib.pyplot as plt                        #for data plotting
        %matplotlib inline

        #Libararies for ML
        from sklearn.model_selection import train_test_split #To split data into training and testing sets
        from sklearn.naive_bayes import GaussianNB             #Classifier
        from sklearn.metrics import confusion_matrix ,accuracy_score         #Confusion Matrix
        from sklearn.metrics import classification_report    #Classification Report

        from sklearn.metrics import roc_auc_score as roc
        from sklearn.model_selection import train_test_split, cross_validate

        from pandas import Series, DataFrame
        from termcolor import colored as cl # text customization

        from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

        import warnings                                        #To manage warnings
        warnings.filterwarnings("ignore")                      #Supress warnings

In [2]: df = pd.read_csv("creditcard.csv")
```

Output:

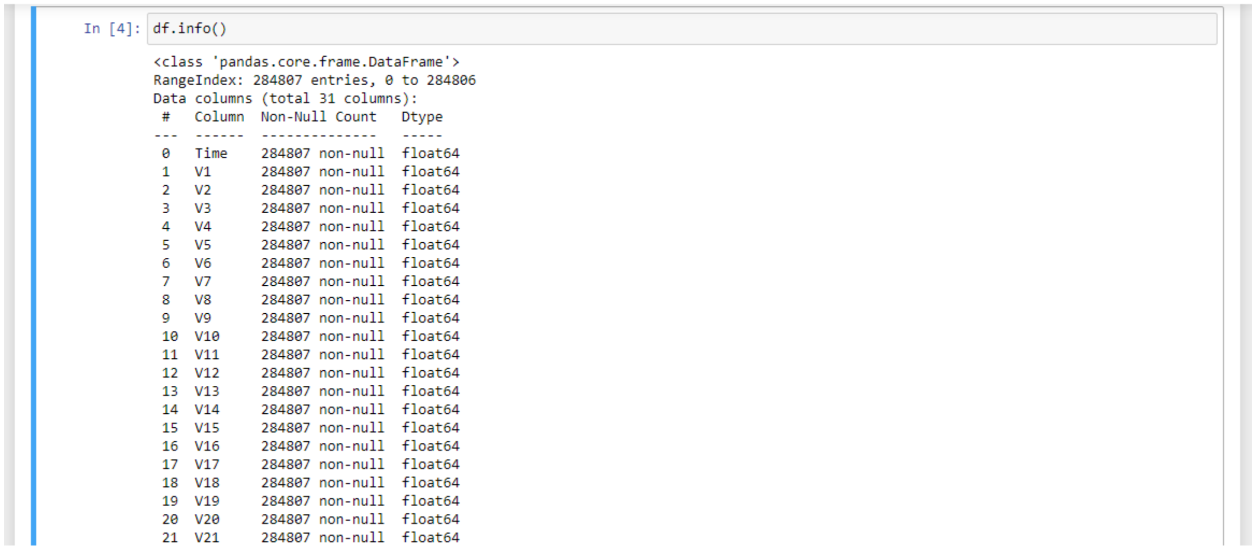
We import the data from a csv file, and we load it into a pandas Data Frame object. The output for the dataset is given:



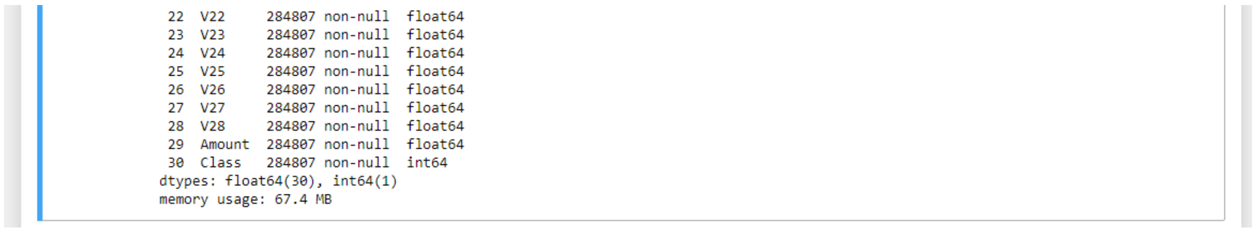
```
In [3]: df.head(3)
Out[3]:
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642

3 rows × 31 columns



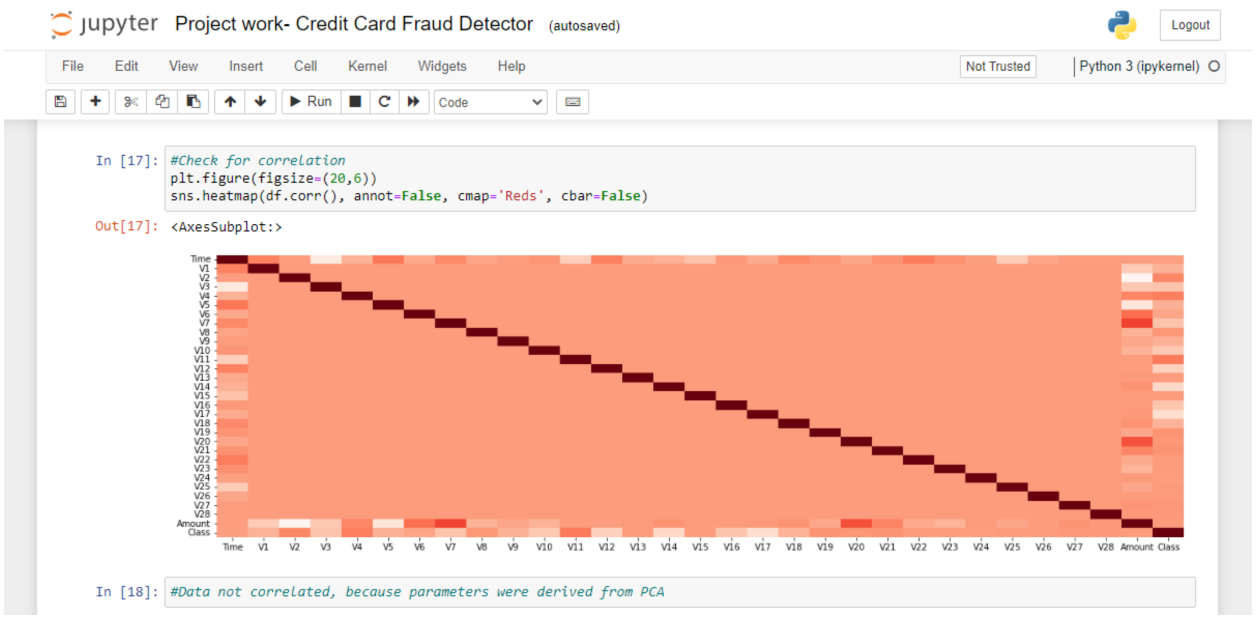
```
In [4]: df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 284807 entries, 0 to 284806
        Data columns (total 31 columns):
         #   Column  Non-Null Count   Dtype
        ---  ------  --------------   -----
         0   Time    284807 non-null  float64
         1   V1      284807 non-null  float64
         2   V2      284807 non-null  float64
         3   V3      284807 non-null  float64
         4   V4      284807 non-null  float64
         5   V5      284807 non-null  float64
         6   V6      284807 non-null  float64
         7   V7      284807 non-null  float64
         8   V8      284807 non-null  float64
         9   V9      284807 non-null  float64
         10  V10     284807 non-null  float64
         11  V11     284807 non-null  float64
         12  V12     284807 non-null  float64
         13  V13     284807 non-null  float64
         14  V14     284807 non-null  float64
         15  V15     284807 non-null  float64
         16  V16     284807 non-null  float64
         17  V17     284807 non-null  float64
         18  V18     284807 non-null  float64
         19  V19     284807 non-null  float64
         20  V20     284807 non-null  float64
         21  V21     284807 non-null  float64
```



```
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

❖ Step 2 - Visualize Data:

We check our data for null values, and we check if there is any correlation between the variables. The data is largely not correlated, stemming from the fact that the parameters are a result of PCA.

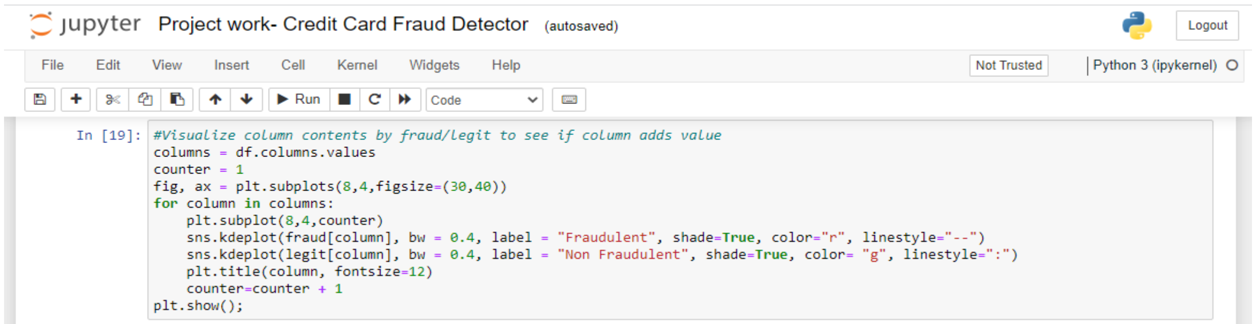


jupyter Project work- Credit Card Fraud Detector (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 Amount Class

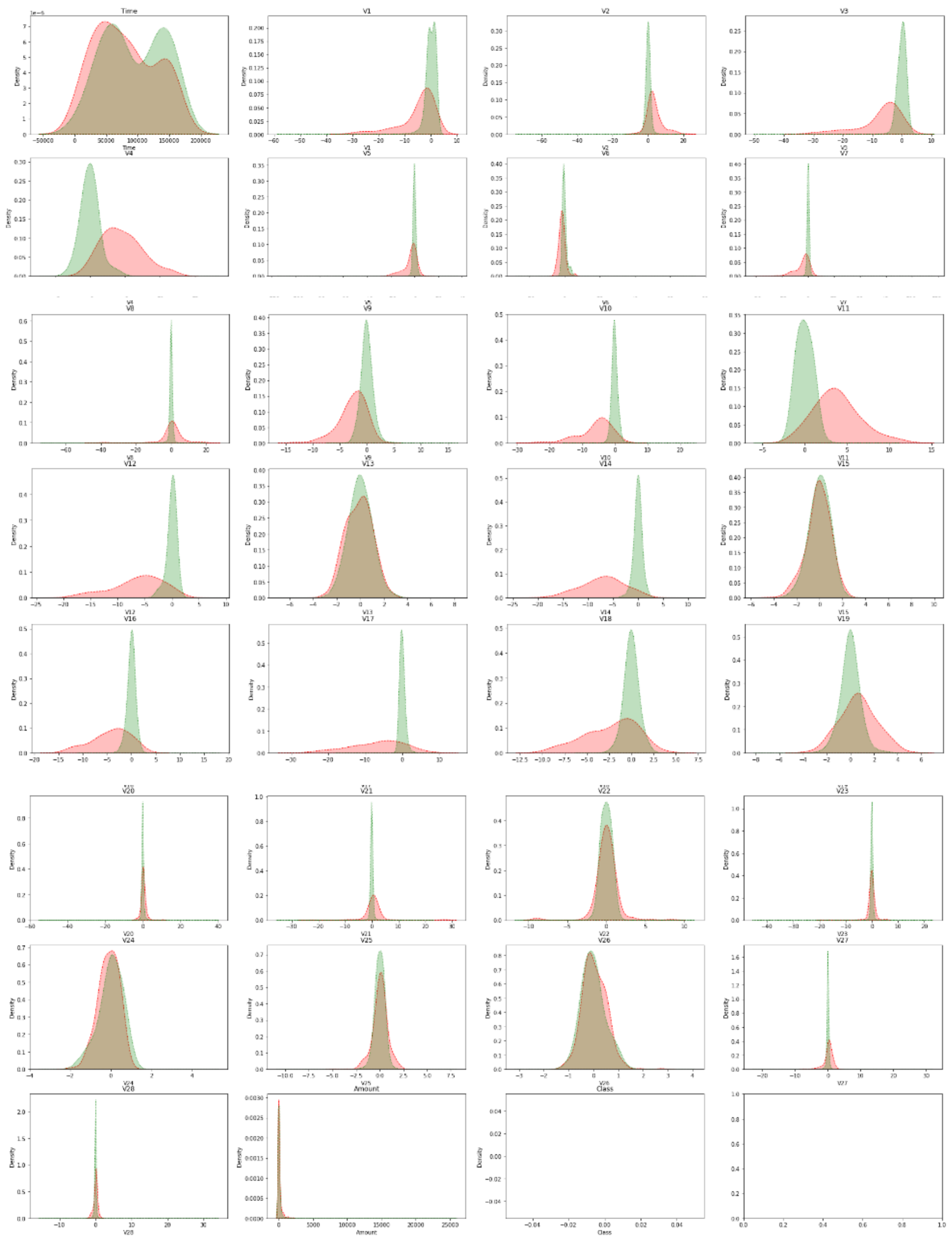
```
In [17]: #Check for correlation
         plt.figure(figsize=(20,6))
         sns.heatmap(df.corr(), annot=False, cmap='Reds', cbar=False)
Out[17]: <AxesSubplot:>
```

```
In [18]: #Data not correlated, because parameters were derived from PCA
```

To perform feature selection, we create a series of graphs for each feature from **V1 to V28** in addition to **Time and Amount**, and we use the target as the hue in order to determine if the feature adds value to the model.



In [19]:
```python
#Visualize column contents by fraud/legit to see if column adds value
columns = df.columns.values
counter = 1
fig, ax = plt.subplots(8,4,figsize=(30,40))
for column in columns:
    plt.subplot(8,4,counter)
    sns.kdeplot(fraud[column], bw = 0.4, label = "Fraudulent", shade=True, color="r", linestyle="--")
    sns.kdeplot(legit[column], bw = 0.4, label = "Non Fraudulent", shade=True, color= "g", linestyle=":")
    plt.title(column, fontsize=12)
    counter=counter + 1
plt.show();
```



❖ Step 3 - Create Training and Testing Set:

We choose the most valuable features, and we split our data set into a training and testing sets. We use **20% for testing** the classifier:

- ✓ **X_train:** Contains the independent variables used for training
- ✓ **y_train:** Contains the dependent variables used for training
- ✓ **X_test:** Contains the independent variables used for testing
- ✓ **y_test:** Contains Dependent variables used for testing

```
In [22]: #Split data into training and testing sets
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2) #20% for testing
```

❖ Step 4 - Train Model:

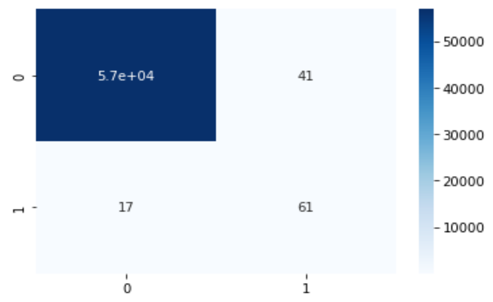
To create the Naive Bayes Model, we instantiate an object from the sklearn GaussianNB module and we fit it to our data.

```
In [23]: NB_classifier = GaussianNB()
         NB_classifier.fit(X_train, y_train)

Out[23]: GaussianNB()
```

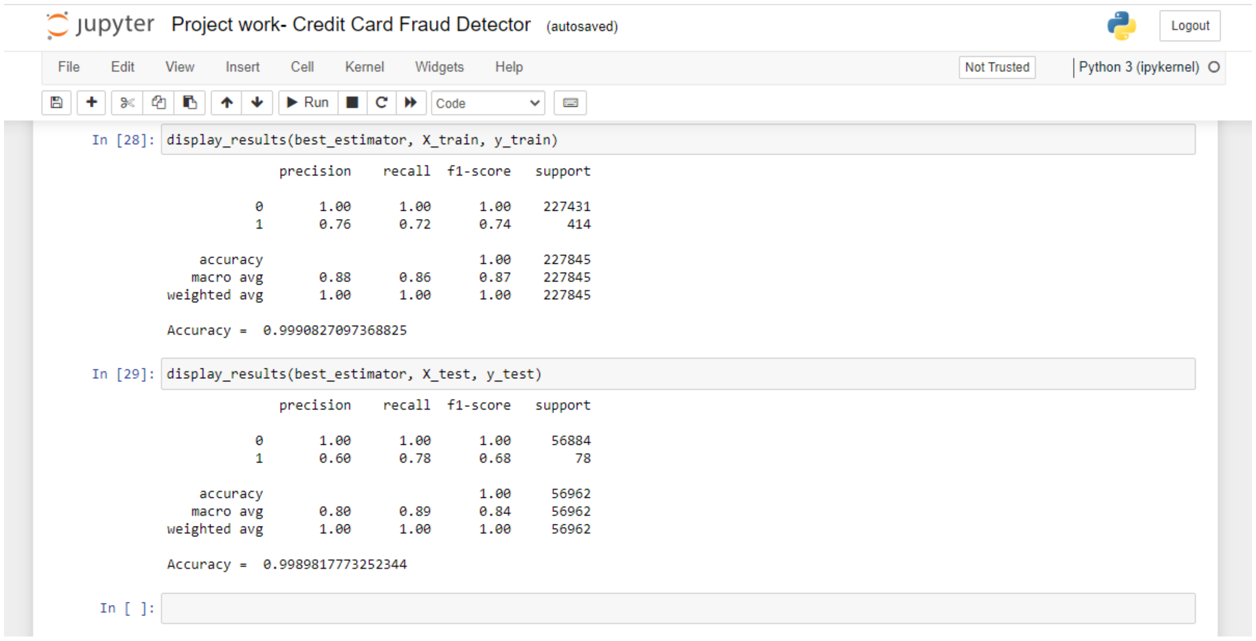
❖ Step 5 - Evaluate Model:

We classify the data and we use a confusion matrix to visualize results. We also generate a classification report:

```
In [26]: y_predict = NB_classifier.predict(X_test)
         cm = confusion_matrix(y_test, y_predict)
         sns.heatmap(cm, annot=True, cmap='Blues')

Out[26]: <AxesSubplot:>
```



Training and Testing Data Performance:


jupyter
Project work- Credit Card Fraud Detector (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [28]: display_results(best_estimator, X_train, y_train)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00    227431
           1       0.76      0.72      0.74       414

    accuracy                           1.00    227845
   macro avg       0.88      0.86      0.87    227845
weighted avg       1.00      1.00      1.00    227845

Accuracy =  0.9990827097368825

In [29]: display_results(best_estimator, X_test, y_test)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56884
           1       0.60      0.78      0.68        78

    accuracy                           1.00     56962
   macro avg       0.80      0.89      0.84     56962
weighted avg       1.00      1.00      1.00     56962

Accuracy =  0.9989817773252344

In [ ]:
```

6. Advantages and Disadvantages of Naive Bayes Classifier

Advantages:

- ✓ Naive Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- ✓ It doesn’t require as much training data
- ✓ It handles both continuous and discrete data
- ✓ It is highly scalable with the number of predictors and data points
- ✓ It is fast and can be used to make real-time predictions
- ✓ It is not sensitive to irrelevant features

Disadvantages:

- ✓ If your test data set has a categorical variable of a category that wasn't present in the training data set, the Naive Bayes model will assign it zero probability and won't be able to make any predictions in this regard. This phenomenon is called 'Zero Frequency,' and you'll have to use a smoothing technique to solve this problem.

- ✓ This algorithm is also notorious as a lousy estimator. So, you shouldn't take the probability outputs of 'predict proba' too seriously.

- ✓ It assumes that all the features are independent. While it might sound great in theory, in real life, you'll hardly find a set of independent features.

7. Conclusion

Naive Bayes Classification is a type of Supervised Machine learning algorithm that is used to classify based on probability calculations. It has three main types; Gaussian classifier, Bernoulli Classifier, and Multinomial Classifier and is used by various applications from different industries, including Business, Health, Technology, Environment, etc. Training is very easy and fast; just requiring considering each attribute in each class separately. Test is straightforward; just looking up tables or calculating conditional probabilities with normal distributions. Performance competitive to most of state-of-the-art classifiers even in presence of violating independence assumption. If the data contains text information as the features, then Naive Bayes is the best option. Besides keeping drawbacks away, Naive Bayes also performs well in data containing numeric and binary values but there are many other algorithms to give a try. Also, before going for text analysis, it is advisable to go through text preprocessing concepts such as lemmatization, stemming, vectorization, POS tagging, stop word removal, etc. which play a crucial role in text mining.

References:

[1] T.M. Cover and J.A. Thomas. Elements of information theory.

New York:John Wiley & Sons, 1991.

[2] P. Domingos and M. Pazzani. On the optimality of the simple

Bayesian classifier under zero-one loss. Machine Learning, 29:103–130, 1997.

[3] R.O. Duda and P.E. Hart. Pattern classification and scene analysis. New York: John Wiley and Sons, 1973.

[4] N. Friedman, D. Geiger, and Goldszmidt M. Bayesian network

classifiers. Machine Learning, 29:131–163, 1997.

[5] J. Hellerstein, Jayram Thathachar, and I. Rish. Recognizing

end-user transactions in performance management. In Proceedings of AAAI-2000, pages 596–602, Austin, Texas, 2000.

[6] J. Hilden. Statistical diagnosis based on conditional independence does not require it. Comput. Biol. Med., 14(4):429–435,

1984.

[7] R. Kohavi. Wrappers for performance enhancement and oblivious decision graphs. Technical report, PhD thesis, Department of Computer Science, Stanford, CA, 1995.

[8] Hands on Machine Learning with Scikit-Learn and Tensorflow by Aurélién Géron.

[9] Introduction to Machine Learning with Python by Andreas C. Müller and Sarah Guido.

[10] Udemy course – Machine Learning – A Z by Kirill Eremenko and Hadelin de Ponteves.

[11] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[12] http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/

[13] https://hands-on.cloud/implementing-naive-bayes-classification-using-python/#h-naive-bayes-classification-implementation

[14] https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn

[15] https://stackabuse.com/the-naive-bayes-algorithm-in-python-with-scikit-learn/

[16] https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.