# Naive Bayes Algorithm for Classification

AcadView

June 5, 2018

---

## 1. Overview

Naive Bayes classifier is a straightforward and powerful algorithm for the classification task. Even if we are working on a data set with millions of records with some attributes, it is suggested to try Naive Bayes approach.

Naive Bayes classifier gives great results when we use it for textual data analysis. Such as Natural Language Processing.

To understand the naive Bayes classifier we need to understand the Bayes theorem. So let's first discuss the Bayes Theorem.

Note: Recall Bayes theorem that we have studied while we discussed probability in this course.

## 2. Bayes' Theorem

Bayes theorem named after Rev. Thomas Bayes. It works on conditional probability. Conditional probability is the probability that something will happen, given that something else has already occurred. Using the conditional probability, we can calculate the probability of an event using its prior knowledge.

Below is the formula for calculating the conditional probability.

$$P(H \mid E) = \frac{P(E \mid H) * P(H)}{P(E)}$$

where

• P(H) is the probability of hypothesis H being true. This is known as the prior probability.

• P(E) is the probability of the evidence(regardless of the hypothesis).

• P(E|H) is the probability of the evidence given that hypothesis is true.

• P(H|E) is the probability of the hypothesis given that the evidence is there.

Let's consider an example to understand how the above formula of Bayes theorem works. A Path Lab is performing a Test of disease say D with two results Positive & Negative. They guarantee that their test result is 99% accurate: if you have the disease, they will give test positive 99% of the time. If you don't have the disease, they will test negative 99% of the time. If 3% of all the people have this disease and test gives positive result, what is the probability that you actually have the disease?

For solving the above problem, we will have to use conditional probability. Probability of people suffering from Disease D,

P(D)=0.03=3%

Probability that test gives positive result and patient have the disease,

P(P os|D)=0.99 = 99%

Probability of people not suffering from Disease D,

P(~ D)=0.97 = 97%

Probability that test gives positive result and patient does have the disease,

P(P os| ~ D)=0.01 = 1%

For calculating the probability that the patient actually have the disease i.e, P(D|P os) we will use Bayes theorem:

$$P(D \mid Pos) = \frac{P(Pos \mid D) * P(D)}{P(Pos)}$$

We have all the values of numerator but we need to calculate P(Pos):

P(Pos)= P(D, pos) + P(~ D, pos)

= P(pos|D) ∗ P(D) + P(pos| ~ D) ∗ P(~ D)

= 0.99 ∗ 0.03 + 0.01 ∗ 0.97 = 0.0297 + 0.0097

= 0.0394

Let's calculate, P(D|P os)=(P(P os|D) ∗ P(D))/P(P os)

= (0.99 ∗ 0.03)/0.0394

= 0.753807107

So, Approximately 75% chances are there that the patient is actually suffering from disease.

# 3. Naive Bayes Classifier

Naive Bayes is a kind of classifier which uses the Bayes Theorem. It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class. This is also known as **Maximum A Posteriori (MAP).**

The MAP for a hypothesis is:

**MAP(H)** = max(P(H|E))

= max((P(E|H) ∗ P(H))/P(E))

= max(P(E|H) ∗ P(H))

P(E) is evidence probability, and it is used to normalize the result. It remains same so, removing it won't affect.

Naive Bayes classifier assumes that all the features are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature. We can use Wikipedia example for explaining the logic i.e.,

A fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

In real datasets, we test a hypothesis given multiple evidence(feature). So, calculations become complicated. To simplify the work, the feature independence approach is used to uncouple' multiple evidence and treat each as an independent one.

*P(H|MultipleEvidences) = P(E1|H)∗P(E2|H)∗P(En|H)∗P(H)/P(MultipleEvidences)*

# 4. Example of Naive Bayes Classifier

Let's consider a training dataset with 1500 records and 3 classes. We presume that there are no missing values in our data.

We have 3 classes associated with Animal Types:

• Parrot,

• Dog,

• Fish.

The Predictor features set consists of 4 features as

• Swim

• Wings

• Green Color

• Dangerous Teeth.

All the features are categorical variables with either of the 2 values:T(True) or F( False).

| Swim | Wings | Green Color | Dangerous Teeth | Animal Type |
|---|---|---|---|---|
| 50 | 500/500 | 400/500 | 0 | Parrot |
| 450/500 | 0 | 0 | 500/500 | Dog |
| 500/500 | 0 | 100/500 | 50/500 | Fish |

The above table shows a frequency table of our data. In our training data:

• Parrots have 50(10%) value for Swim, i.e., 10% parrot can swim according to our data, 500 out of 500(100%) parrots have wings, 400 out of 500(80%) parrots are Green and 0(0%) parrots have Dangerous Teeth.

• Classes with Animal type Dogs shows that 450 out of 500(90%) can swim, 0(0%) dogs have wings, 0(0%) dogs are of Green color and 500 out of 500(100%) dogs have Dangerous Teeth.

• Classes with Animal type Fishes shows that 500 out of 500(100%) can swim, 0(0%) fishes have wings, 100(20%) fishes are of Green color and 50 out of 500(10%) dogs have Dangerous Teeth.

Now, it's time to work on predict classes using the Naive Bayes model. We have taken 2 records that have values in their feature set, but the target variable needs to predicted

| | Swim | Wings | Green | Teeth |
|---|---|---|---|---|
| 1. | True | False | True | False |
| 2. | True | False | True | True |

We have to predict animal type using the feature values.

We will use the Naive Bayes approach

**P(H|Multiple Evidences)** = P(E1| H)* P(E2|H) ……*P(En|H) * P(H) / P(Multiple Evidences)

Let's consider the first record.

The Evidence here is **Swim & Green**. The Hypothesis can be an animal type to be Dog, Parrot, Fish.

**For Hypothesis testing for the animal to be a Dog:**

P(Dog | Swim, Green) = P(Swim|Dog) * P(Green|Dog) * P(Dog) / P(Swim, Green)

= 0.9 * 0 * 0.333 / P(Swim, Green)

= 0

**For Hypothesis testing for the animal to be a Parrot:**

P(Parrot| Swim, Green) = P(Swim|Parrot) * P(Green|Parrot) * P(Parrot) / P(Swim, Green)

= 0.1 * 0.80 * 0.333 / P(Swim, Green)

= 0.0264/ P(Swim, Green)

**For Hypothesis testing for the animal to be a Fish:**

P(Fish| Swim, Green) = P(Swim|Fish) * P(Green|Fish) * P(Fish) / P(Swim, Green)

= 1 * 0.2 * 0.333 / P(Swim, Green)

= 0.0666/ P(Swim, Green)

The denominator of all the above calculations is same i.e, P(Swim, Green). The value of P(Fish| Swim, Green) is greater that P(Parrot| Swim, Green).

Using Naive Bayes, we can predict that the class of this record is Fish.

Let's consider the second record.

The Evidence here is Swim, Green & Teeth. The Hypothesis can be an animal type to be Dog, Parrot, Fish.

**For Hypothesis testing for the animal to be a Dog:**

P(Dog | Swim, Green, Teeth) = P(Swim|Dog) * P(Green|Dog) * P(Teeth|Dog) * P(Dog) / P(Swim, Green, Teeth)

= 0.9 * 0 * 1 * 0.333 / P(Swim, Green, Teeth)

= 0

**For Hypothesis testing for the animal to be a Parrot:**

P(Parrot| Swim, Green, Teeth) = P(Swim|Parrot) * P(Green|Parrot)* P(Teeth|Parrot) * P(Parrot) / P(Swim, Green, Teeth)

= 0.1 * 0.80 * 0 *0.333 / P(Swim, Green, Teeth)

= 0

**For Hypothesis testing for the animal to be a Fish:**

P(Fish|Swim, Green, Teeth) = P(Swim|Fish) * P(Green|Fish) * P(Teeth|Fish) *P(Fish) / P(Swim, Green, Teeth)

= 1 * 0.2 * 0.1 * 0.333 / P(Swim, Green, Teeth)

= 0.00666 / P(Swim, Green, Teeth)

The denominator of all the above calculations is same i.e, P(Swim, Green, Teeth). The value of P(Fish| Swim, Green, Teeth) is the only positive value greater than 0. Using Naive Bayes, we can predict that the class of this record is Fish.

As the calculated value of probabilities is very less. To normalize these values, we need to use denominators.

Let's proceed to learn the various type of **Naive Bayes Methods**.

# 5. Types of Naive Bayes Algorithm

## 5.1. Gaussian Naive Bayes

When attribute values are continuous, an assumption is made that the values associated with each class are distributed according to Gaussian i.e., Normal Distribution.

If in our data, an attribute say "x" contains continuous data. We first segment the data by the class and then compute mean $\mu_y$ & Variance $\sigma_y^2$ of each class.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

## 5.2. MultiNomial Naive Bayes

MultiNomial Naive Bayes is preferred to use on data that is multinomially distributed. It is one of the standard classic algorithms. Which is used in text categorization (classification). Each event in text classification represents the occurrence of a word in a document.

## 5.3. Bernoulli Naive Bayes

Bernoulli Naive Bayes is used on the data that is distributed according to multivariate Bernoulli distributions.i.e., multiple features can be there, but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. So, it requires features to be binary valued.

# 6 Simple Python Implementation

In this part we will apply what we have learned in Python using the scikit-learn library.

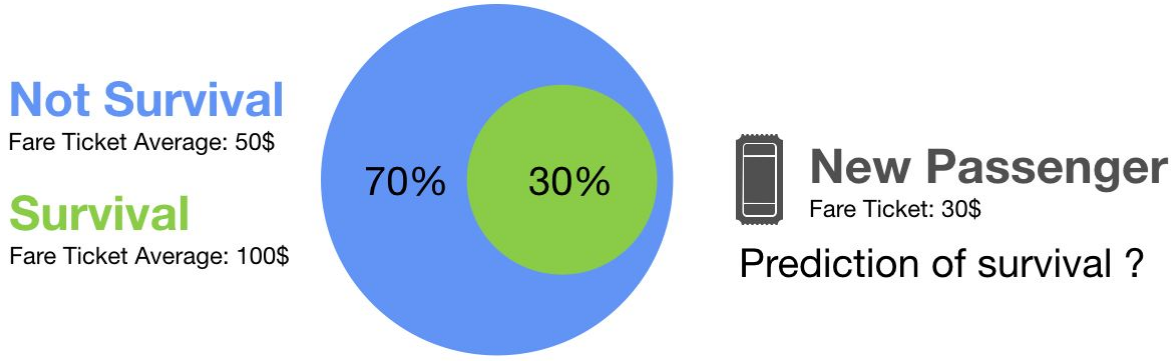
```
# Gaussian Naive Bayes
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
# load the iris datasets
dataset = datasets.load_iris()
# fit a Naive Bayes model to the data
model = GaussianNB()
model.fit(dataset.data, dataset.target)
print(model)
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print(metrics.accuracy_score(expected,predicted))
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

The output is as follows:

```
GaussianNB(priors=None)
0.96
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        50
          1       0.94      0.94      0.94        50
          2       0.94      0.94      0.94        50

avg / total       0.96      0.96      0.96       150

[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
[Finished in 0.7s]
```

# 7. Naive Bayes on Titanic Disaster Dataset

Let's take the famous **Titanic Disaster dataset**. It gathers Titanic passenger personal information and whether or not they survived to the shipwreck. Let's try to make a prediction of survival using passenger ticket fare information.




Imagine you take a random sample of 500 passengers. In this sample, **30% of people survived**. Among passenger who survived, the **fare ticket mean is 100$**. It falls to **50$** in the subset of people who **did not survive**. Now, let's say you have a new passenger. You do not know if he survived or not but you know he bought a **30$ ticket** to cross the Atlantic. What is your prediction of survival for this passenger?

## Python Implementation

Here we implement a classic **Gaussian Naive Bayes** on the Titanic Disaster dataset. We will use Class of the room, Sex, Age, number of siblings/spouses, number of parents/children, passenger fare and port of embarkation information.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB

# Importing dataset
data = pd.read_csv("data/train.csv")

# Convert categorical variable to numeric
data["Sex_cleaned"]=np.where(data["Sex"]=="male",0,1)
data["Embarked_cleaned"]=np.where(data["Embarked"]=="S",0,
                                  np.where(data["Embarked"]=="C",1,
                                           np.where(data["Embarked"]=="Q",2,3)
                                          )
                                 )
```

```
# Cleaning dataset of NaN
data=data[[
    "Survived",
    "Pclass",
    "Sex_cleaned",
    "Age",
    "SibSp",
    "Parch",
    "Fare",
    "Embarked_cleaned"
]].dropna(axis=0, how='any')

# Split dataset in training and test datasets
X_train, X_test = train_test_split(data, test_size=0.5, random_state=int(time.time()))
```

```
# Instantiate the classifier
gnb = GaussianNB()
used_features =[
    "Pclass",
    "Sex_cleaned",
    "Age",
    "SibSp",
    "Parch",
    "Fare",
    "Embarked_cleaned"
]

# Train classifier
gnb.fit(
    X_train[used_features].values,
    X_train["Survived"]
)
y_pred = gnb.predict(X_test[used_features])


# Print results
print("Number of mislabeled points out of a total {} points : {}, performance {:05.2f}%"
      .format(
          X_test.shape[0],
          (X_test["Survived"] != y_pred).sum(),
          100*(1-(X_test["Survived"] != y_pred).sum()/X_test.shape[0])
))
```

```
> Number of mislabeled points out of a total 357 points: 68,
performance 80.95%
```

The performance of our classifier is **80.95%.**

# 8. Advantages and Disadvantage of Naive Bayes classifier

**Advantages:**

• Naive Bayes Algorithm is a fast, highly scalable algorithm.

• Naive Bayes can be use for Binary and Multiclass classification. It provides different types of

• Naive Bayes Algorithms like GaussianNB, MultinomialNB, BernoulliNB.

• It is a simple algorithm that depends on doing a bunch of counts.

• Great choice for Text Classification problems. It's a popular choice for spam email classification.

• It can be easily train on small dataset

**Disadvantages**

• It considers all the features to be unrelated, so it cannot learn the relationship between features. E.g., Lets say Remo is going to a part. While cloth selection for the party, Remo is looking at his cupboard. Remo likes to wear a white color shirt. In Jeans, he likes to wear a brown Jeans, But Remo doesn't like wearing a white shirt with Brown Jeans. Naive Bayes can learn individual features importance but can't determine the relationship among features.