Overview

- Understand one of the most popular and simple machine learning classification algorithms, the Naive Bayes algorithm
- It is based on the Bayes Theorem for calculating probabilities and conditional probabilities
- Learn how to implement the Naive Bayes Classifier in R and Python

Introduction

Here's a situation you've got into in your data science project:

You are working on a classification problem and have generated your set of hypothesis, created features and discussed the importance of variables. Within an hour, stakeholders want to see the first cut of the model.

What will you do? You have hundreds of thousands of data points and quite a few variables in your training data set. In such a situation, if I were in your place, I would have used '**Naive Bayes**', which can be extremely fast relative to other classification algorithms. It works on Bayes theorem of probability to predict the class of unknown data sets.

In this article, I'll explain the basics of this algorithm, so that next time when you come across large data sets, you can bring this algorithm to action. In addition, if you are a newbie in Python or R, you should not be overwhelmed by the presence of available codes in this article.

If you prefer to learn Naive Bayes theorem from the basics concepts to the implementation in a structured manner, you can enroll in this free course:

- Naive Bayes from Scratch

Are you a beginner in Machine Learning? Do you want to master the machine learning algorithms like Naive Bayes? Here is a comprehensive course covering the machine learning and deep learning algorithms in detail –

Table of Contents

6. Tips to improve the power of Naive Bayes Model

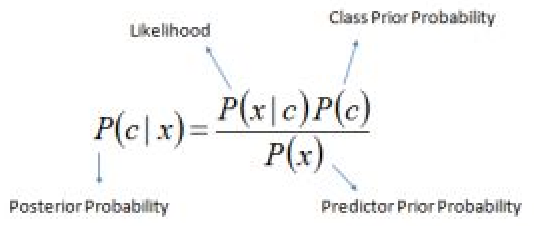
What is Naive Bayes algorithm?

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Above,

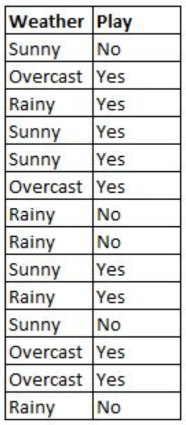
- *P*(*c*|*x*) is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- *P*(*c*) is the prior probability of *class*.
- *P*(*x*|*c*) is the likelihood which is the probability of *predictor* given *class*.
- *P*(*x*) is the prior probability of *predictor*.

How Naive Bayes algorithm works?

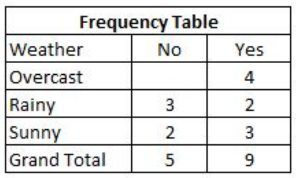
Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let’s follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

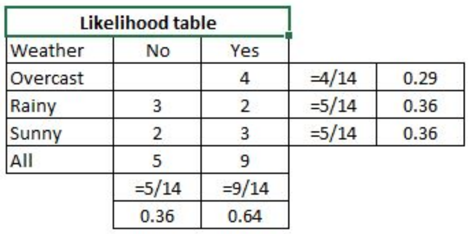
Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.



Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No



Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9



Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

What are the Pros and Cons of Naive Bayes?

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from `predict_proba` are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

4 Applications of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

How to build a basic model using Naive Bayes in R?

Again, scikit learn (python library) will help here to build a Naive Bayes model in Python. There are three types of Naive Bayes model under the scikit-learn library:

- **Gaussian:** It is used in classification and it assumes that features follow a normal distribution.

- **Multinomial:** It is used for discrete counts. For example, let’s say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of “word occurring in the document”, we have “count how often word occurs in the document”, you can think of it as “number of times outcome number x_i is observed over the n trials”.

- **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with ‘bag of words’ model where the 1s & 0s are “word occurs in the document” and “word does not occur in the document” respectively.

```
require(e1071) #Holds the Naive Bayes Classifier

Train <- read.csv(file.choose())

Test <- read.csv(file.choose())



#Make sure the target variable is of a two-class classification problem only



levels(Train$Item_Fat_Content)



model <- naiveBayes(Item_Fat_Content~., data = Train)

class(model)

pred <- predict(model,Test)

table(pred)
```

Above, we looked at the basic Naive Bayes model, you can improve the power of this basic model by tuning parameters and handle assumption intelligently. Let’s look at the methods to improve the performance of Naive Bayes Model. I’d recommend you to go through this document for more details on Text classification using Naive Bayes.

Tips to improve the power of Naive Bayes Model

Here are some tips for improving power of Naive Bayes Model:

- If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.
- If test data set has zero frequency issue, apply smoothing techniques “Laplace Correction” to predict the class of test data set.
- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.
- Naive Bayes classifiers has limited options for parameter tuning like alpha=1 for smoothing, fit_prior=[True|False] to learn class prior probabilities or not and some other options (look at detail here). I would recommend to focus on your pre-processing of data and the feature selection.
- You might think to apply some *classifier combination technique like* ensembling, bagging and boosting but these methods would not help. Actually, “ensembling, boosting, bagging” won’t help since their purpose is to reduce variance. Naive Bayes has no variance to minimize.

Bayes’ Theorem - Example Solution

A simple approach to Bayes’ Theorem with example

Conditional probability is the sine qua non of data science and statistics. There are many useful explanations and examples of conditional probability and Bayes’ Theorem. In this article, I will explain the background of the Bayes’ Theorem with example by using simple math.

Bayes’ Theorem looks simple in mathematical expressions such as;

P(A|B) = P(B|A)P(A)/P(B)

The important point in data science is not the equation itself, the application of this equation to the verbal problem is more important than remembering the equation. So, I will solve a simple conditional probability problem with Bayes theorem and logic.

Problem 1:

Let’s work on a simple NLP problem with Bayes Theorem. By using NLP, I can detect spam e-mails in my inbox. Assume that the word ‘offer’ occurs in 80% of the spam messages in my account. Also, let’s assume ‘offer’ occurs in 10% of my desired e-mails. If 30% of the received e-mails are considered as a scam, and I will receive a new message which contains ‘offer’, what is the probability that it is spam?

Now, I assume that I received 100 e-mails. The percentage of spams in the whole e-mail is 30%. So, I have 30 spam e-mails and 70 desired e-mails in 100 e-mails. The percentage of the word ‘offer’ that occurs in spam e-mails is 80%. It means 80% of 30 e-mail and it makes 24. Now, I know that 30 e-mails of 100 are spam and 24 of them contain ‘offer’ where 6 of them not contains ‘offer’.

The percentage of the word ‘offer’ that occurs in the desired e-mails is 10%. It means 7 of them (10% of 70 desired e-mails) contain the word ‘offer’ and 63 of them not.

Now, we can see this logic in a simple chart.

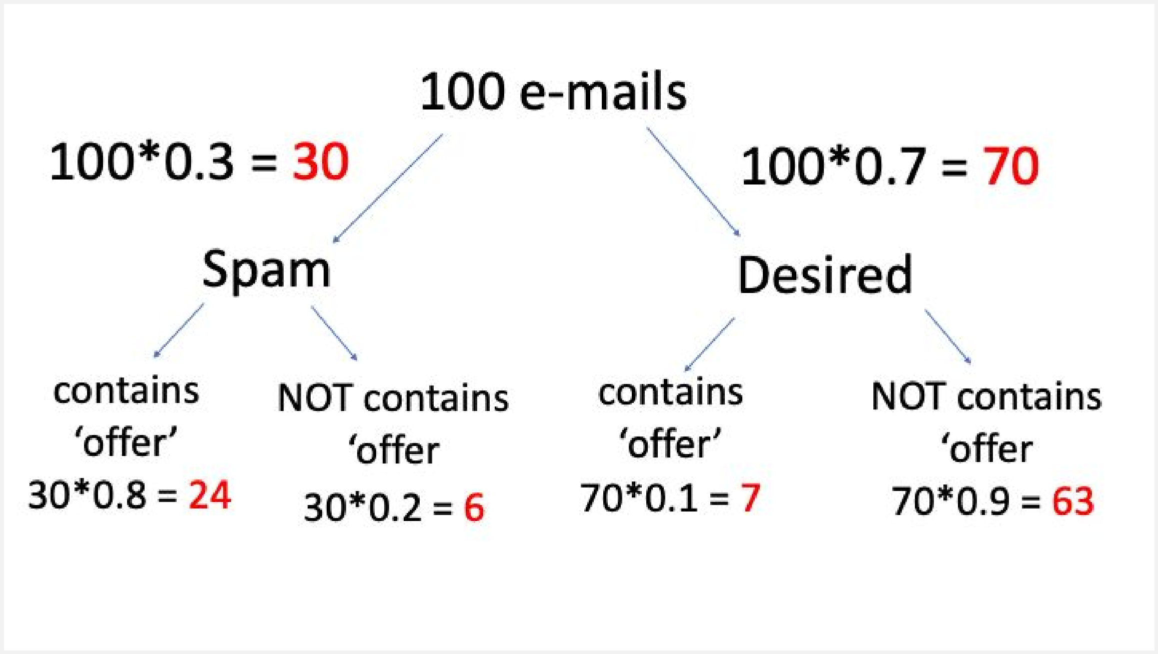

100 e-mails
100*0.3 = 30
100*0.7 = 70
Spam
Desired
contains ‘offer’ 30*0.8 = 24
NOT contains ‘offer 30*0.2 = 6
contains ‘offer’ 70*0.1 = 7
NOT contains ‘offer 70*0.9 = 63

image by author

The question was what is the probability of spam where the mail contains the word ‘offer’:

1. We need to find the total number of mails which contains ‘offer’ ;

24 +7 = 31 mail contain the word ‘offer’

2. Find the probability of spam if the mail contains 'offer' ;

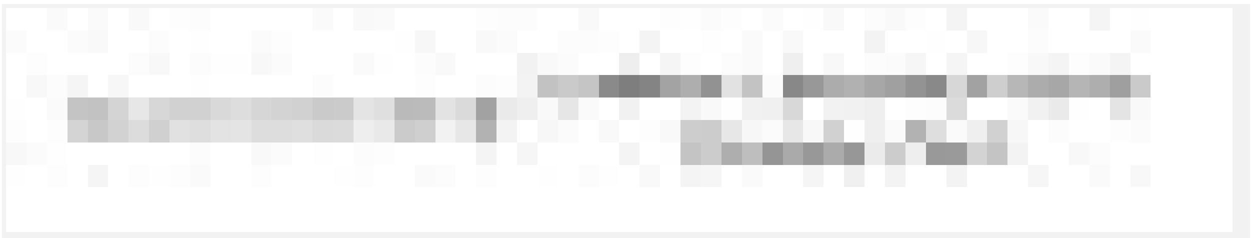
In 31 mails 24 contains 'offer' means 77.4% = 0.774 (probability)

NOTE: In this example, I choose the percentages which give integers after calculation. As a general approach, you can think that we have 100 units at the beginning so if the results are not an integer, it will not create a problem. Such that, we cannot say 15.3 e-mails but we can say 15.3 units.

Solution with Bayes' Equation:

A = Spam

B = Contains the word 'offer'



$$P(spam|contains\ offer) = \frac{P(contains\ offer|spam) * P(spam)}{P(contains\ offer)}$$

image by author

P(contains offer|spam) = 0.8 (given in the question)

P(spam) = 0.3 (given in the question)

Now we will find the probability of e-mail with the word ‘offer’. We can compute that by adding ‘offer’ in spam and desired e-mails. Such that;

P(contains offer) = 0.3*0.8 + 0.7*0.1 = 0.31

[illegible]

$$P(spam|contains\ offer) = \frac{0.8 * 0.3}{0.31} = 0.774$$

image by author

As it is seen in both ways the results are the same. In the first part, I solved the same question with a simple chart and for the second part, I solved the same question with Bayes’ theorem.

Problem 2:

I want to solve one more example from a popular topic as Covid-19. As you know, Covid-19 tests are common nowadays, but some results of tests are not true. Let’s assume; a diagnostic test has 99% accuracy and 60% of all people have Covid-19. If a patient tests positive, what is the probability that they actually have the disease?

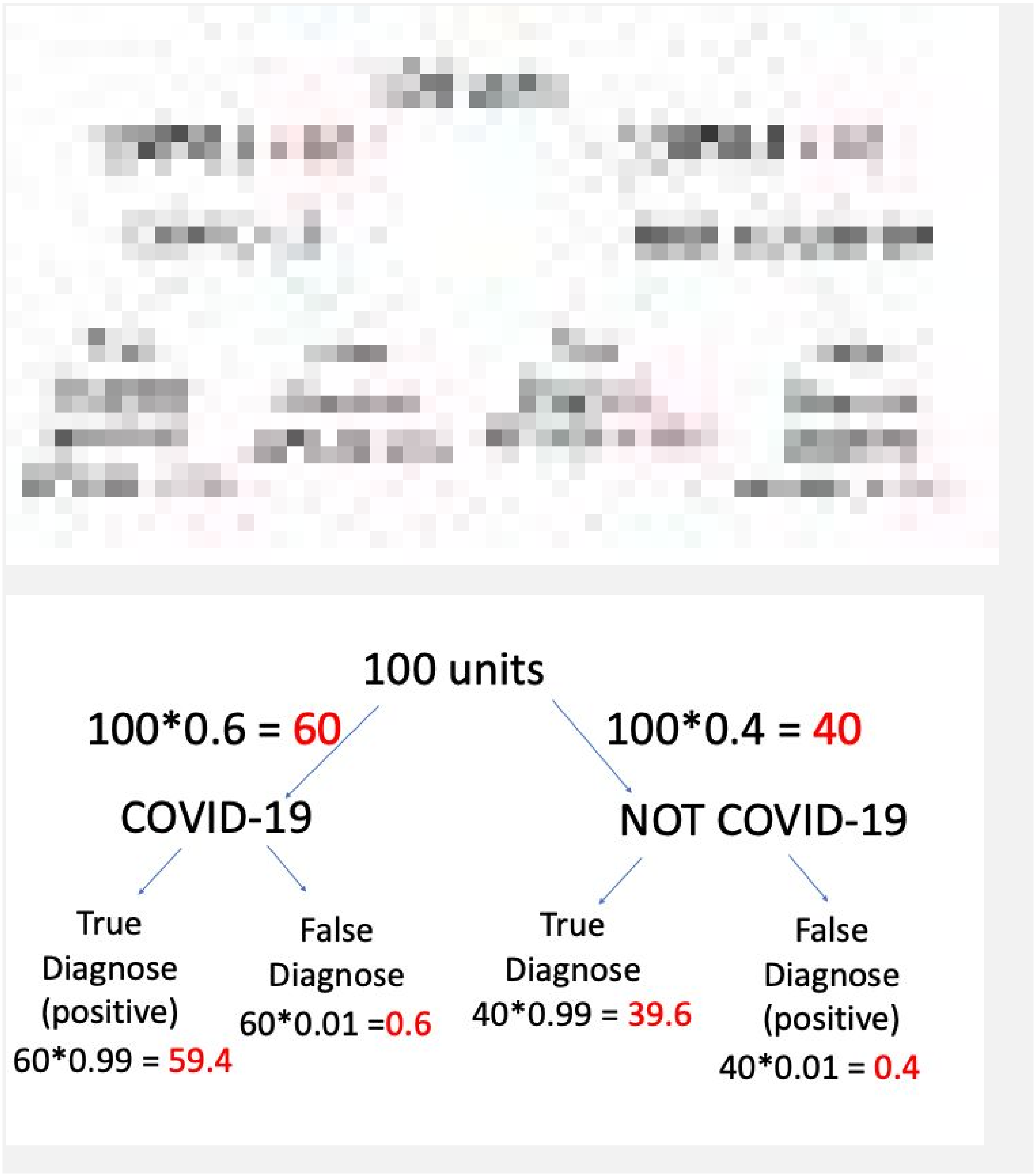


image by author

The total units which have positive results= 59.4 + 0.4 = 59.8

59.4 units (true positive) is 59.8 units means 99.3% = 0.993 probability

With Bayes';

[illegible]

$$P(covid19|positive) = \frac{P(positive|covid19) * P(covid19)}{P(positive)}$$

image by author

P(positive|covid19) = 0.99

P(covid19) = 0.6

P(positive) = 0.6*0.99+0.4*0.01=0.598

[illegible]

$$P(covid19|positive) = \frac{0.99 * 0.6}{0.598} = 0.993$$

image by author

Again, we find the same answer with the chart. There are many examples to learn Bayes' Theorem's applications such as the Monty Hall problem which is a little puzzle that you have 3 doors. Behind the doors, there are 2 goats and 1 car. You are asked to select one door to find the car. After selecting one door, the host opens one of the not selected doors and revealing goat. Then, you are asked to switch the doors or stick with your first choice. With running this process a thousand times and simulating it, you can find the probability of winning and figure out the idea of Bayes' theorem and Bayesian statistics in general through the Monty Hall problem.

When we think Bayes' Theorem in the machine learning concept, it provides a way to calculate the probability of the hypothesis based on conditions by using the relationship between data and hypothesis. Also, it is the first step for understanding True Positive, False Positive, True Negative, and False Negative concepts in data science classification problems and Naive Bayes classifier.