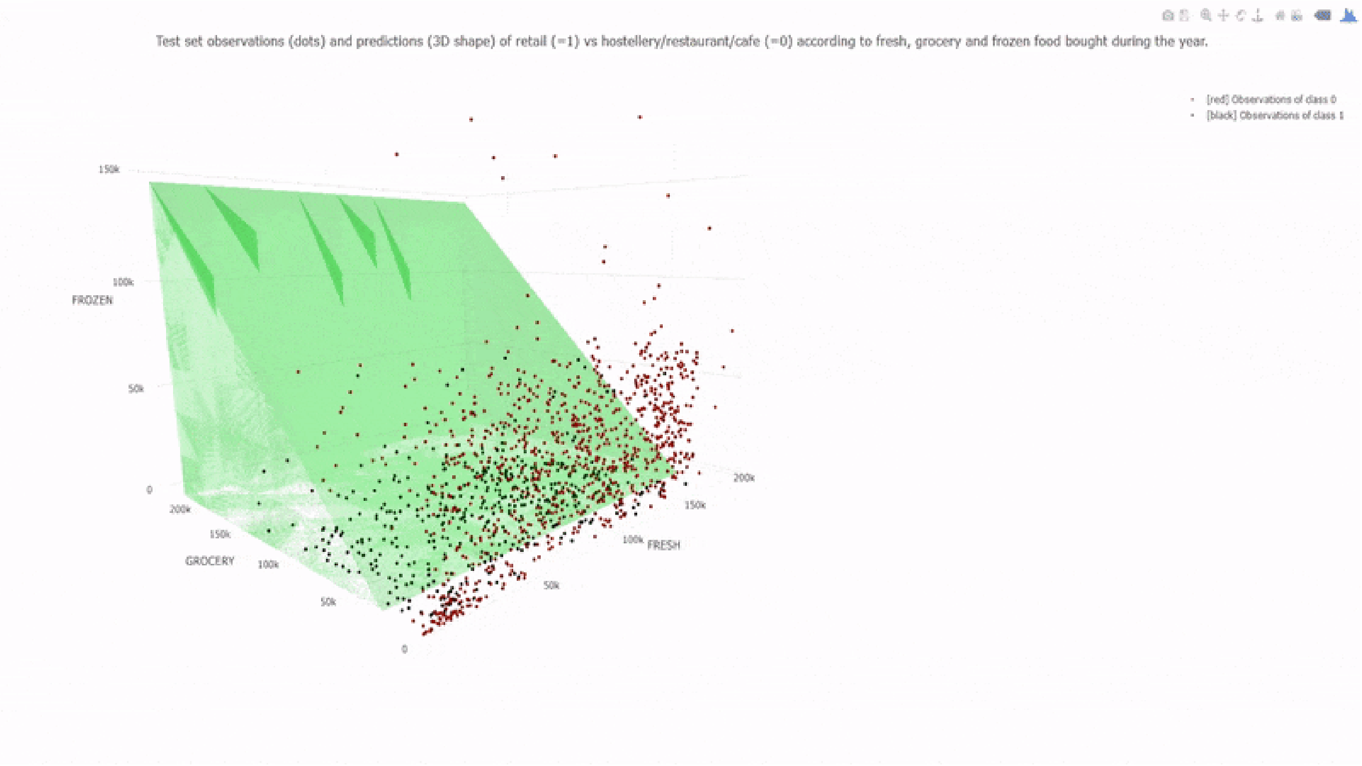
towards
data science

All about Naive Bayes

Gaurav Chauhan Oct 8, 2018 · 6 min read



· · ·

> Learning a Naive Bayes classifier is just a matter of counting how many times each attribute co-occurs with each class

· · ·

Naive Bayes is the most simple algorithm that you can apply to your data. As the name suggests, here this algorithm makes an assumption as all the variables in the dataset is “Naive” i.e not correlated to each other.

Naive Bayes is a very popular classification algorithm that is mostly used to get the base accuracy of the dataset.

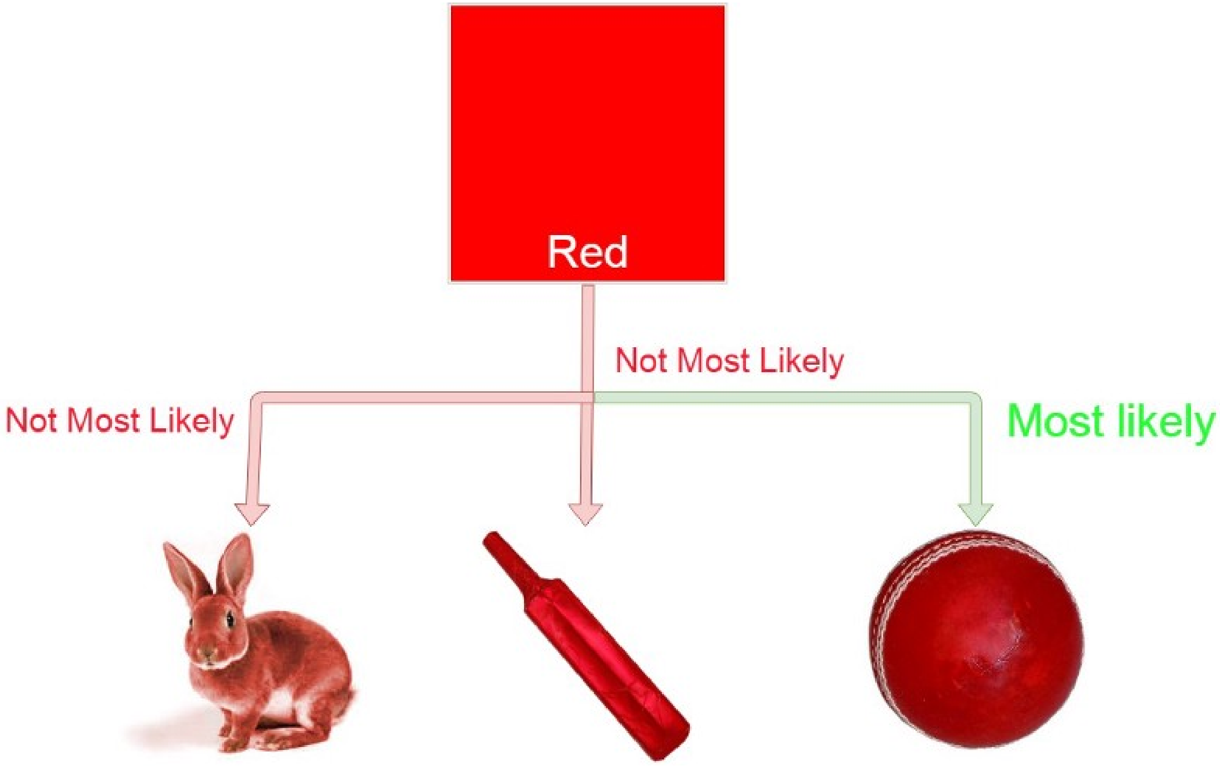
Explain like I am five

Let's assume that you are walking on the playground. Now you see some red object in front of you. This red object can be a bat or a cat or a ball. You will definitely assume that it will be a ball. But why so?

Let's us think you are making a machine and you have given the task as above to classify an object in between bat, ball and a cat. At first you will think of creating a machine that will identify the characters of the object and then map it with your classification objects such that if an object is a circle then it will be a ball or if the object is living-being then it will be a cat or in our case, if our object is red then it is most probable that it will be a ball.

Why so? because from our childhood we have seen a red ball but a red cat or a red bat is very unlikely to our eyes.

So in our case, we can classify an object by mapping its features with our classifier individually. As in our case, this red color was mapped with a bat, a cat, and a ball, but eventually, we get the most probability of red object with a ball and therefore we classified that object with a ball.

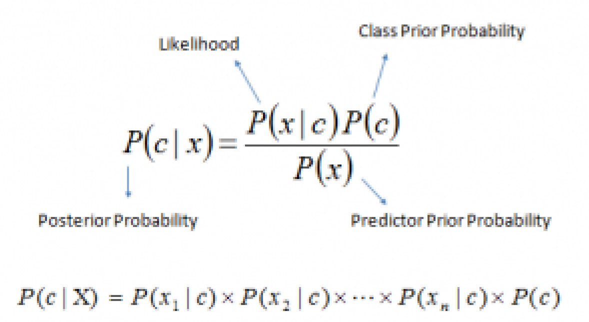


. . .

Formula

Here c represents the class eg. ball, cat, bat.

x represents features calculated individually.



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

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

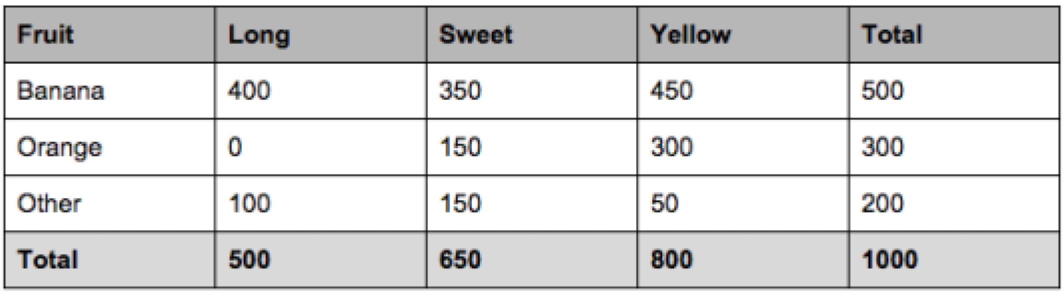
Where,

- $P(c|x)$ is the posterior probability of *class* c given *predictor* (*features*).
- $P(c)$ is the 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*.

. . .

Example

let's say we have data on 1000 pieces of fruit. The fruit being a Banana, Orange or some other fruit and imagine we know 3 features of each fruit, whether it's long or not, sweet or not and yellow or not, as displayed in the table below.



Fruit	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	**500**	**650**	**800**	**1000**

So from the table what do we already know?

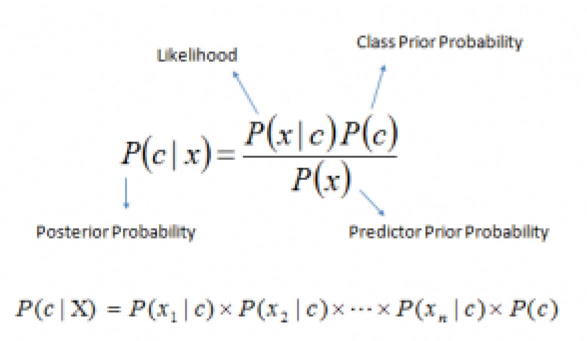
- 50% of the fruits are bananas
- 30% are oranges
- 20% are other fruits

Based on our training set we can also say the following:

- From 500 bananas 400 (0.8) are Long, 350 (0.7) are Sweet and 450 (0.9) are Yellow
- Out of 300 oranges, 0 are Long, 150 (0.5) are Sweet and 300 (1) are Yellow
- From the remaining 200 fruits, 100 (0.5) are Long, 150 (0.75) are Sweet and 50 (0.25) are Yellow

Which should provide enough evidence to predict the class of another fruit as it's introduced.

So let's say we're given the features of a piece of fruit and we need to predict the class. If we're told that the additional fruit is Long, Sweet and Yellow, we can classify it using the following formula and subbing in the values for each outcome, whether it's a Banana, an Orange or Other Fruit. The one with the highest probability (score) being the winner.



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

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

formula for reference

Banana:

$$P\left(\frac{Banana}{Long, Sweet, Yellow}\right) = \frac{P\left(\frac{Long}{Banana}\right) \times P\left(\frac{Sweet}{Banana}\right) \times P\left(\frac{Yellow}{Banana}\right) \times P(Banana)}{P(Long)\,P(Sweet)\,P(Yellow)}$$

$$P\left(\frac{Banana}{Long, Sweet, Yellow}\right) = \frac{(0.8) \times (0.7) \times (0.9) \times (0.5)}{0.25 \times 0.33 \times 0.41}$$

$$P\left(\frac{Banana}{Long, Sweet, Yellow}\right) = 0.252$$

Orange:

$$P\left(\frac{Orange}{Long, Sweet, Yellow}\right) = 0$$

Other Fruit:

$$P\left(\frac{Other}{Long, Sweet, Yellow}\right) = \frac{P\left(\frac{Long}{Other}\right) \times P\left(\frac{Sweet}{Other}\right) \times P\left(\frac{Yellow}{Other}\right) \times P(Other)}{P(Long)\,P(Sweet)\,P(Yellow)}$$

$$P\left(\frac{Other}{Long, Sweet, Yellow}\right) = \frac{(0.5) \times (0.75) \times (0.25) \times (0.2)}{0.25 \times 0.33 \times 0.41}$$

$$P(\frac{Other}{Long, Sweet, Yellow}) = 0.01875$$

In this case, based on the higher score (0.252 for banana) we can assume this Long, Sweet and Yellow fruit is in fact, a Banana.

. . .

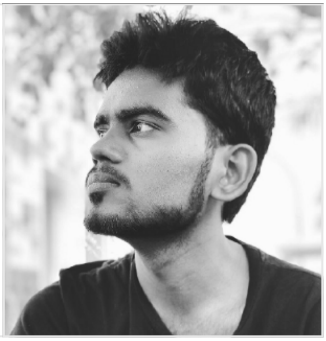
Implementation in python

- Implementation of Naive Bayes algorithm **from scratch** in python with explanation in each step is uploaded to my Github repository.
- Implementation of Naive Bayes with **help of Scikit learn** is also added to my Github repository.

2796gaurav/Naive-bayes-explained

This is a very in depth explination of naive bayes w.r.t implementation in python which can be used in Machine Learning…

github.com



. . .

Advantages

- It is easy and fast to predict the class of the test data set. It also performs 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).

. . .

Disadvantages

- 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 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.

• • •

Applications

- **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.

• • •

When to use

- Text Classification
- when dataset is huge
- When you have small training set

• • •

Further i will add other machine learning algorithms. The main motto of this article was to give an in depth knowledge of Naive Bayes without using any hard word and explain it from scratch. Further if you want to implement Naive bayes, start from these datasets and you can comment your predicted score with code in the comments section.

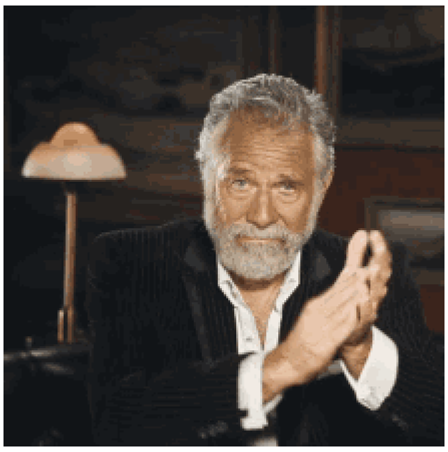
- Iris dataset

- Wine dataset
- Adult dataset

Till then,

Happy coding :)

And Don’t forget to clap clap clap…



Machine Learning · Artificial Intelligence · Data Science · Algorithms · Data