# 1.8. Naïve Bayes Classifier

Naïve Bayes is a probabilistic algorithm that's typically used for classification problems hence the name Naïve Bayes Classifier. Naïve Bayes is simple, intuitive, and yet performs surprisingly well in many cases. For example, spam filters Email app uses are built on Naïve Bayes.

Let's have a recap of some basic concepts on probability and Bayes' Theorem which is the core of Naïve Bayes Classifier.

- Probability simply means the likelihood of an event to occur and always takes a value between 0 and 1 (0 and 1 inclusive). The probability of event A is denoted as p(A) and calculated as the number of the desired outcome divided by the number of all outcomes.

- Conditional probability is the likelihood of an event A to occur given that another event B that has a relation with event A has already occurred.

  Conditional probability,

$$P(A \mid B) = \frac{P(A \text{ and } B)}{P(B)}$$

- A joint probability is the probability of two events occurring together and denoted as $P(A \text{ and } B)$ or $P(A \cap B)$. *For independent events*, joint probability can be written as:

$$P(A \cap B) = P(A)\,P(B)$$

- Bayes Theorem gives us the probability of an event (say A), based on prior knowledge of conditions that might be related to the event.

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

  Where $A$ and $B$ are the events and $P(B) \neq 0$

  - ✓ $P(A \mid B)$ is a conditional probability, the likelihood of event A occurring given that B is true.

  - ✓ $P(B \mid A)$ is a conditional probability, the likelihood of event B occurring given that A is true.

  - ✓ $P(A)$ and $P(B)$ are the probabilities of observing A and B respectively; they are known as the marginal probability.

A joint probability is the probability of two events occurring together and denoted $as\ p(A\ and\ B)$. For independent events, joint probability can be written as:

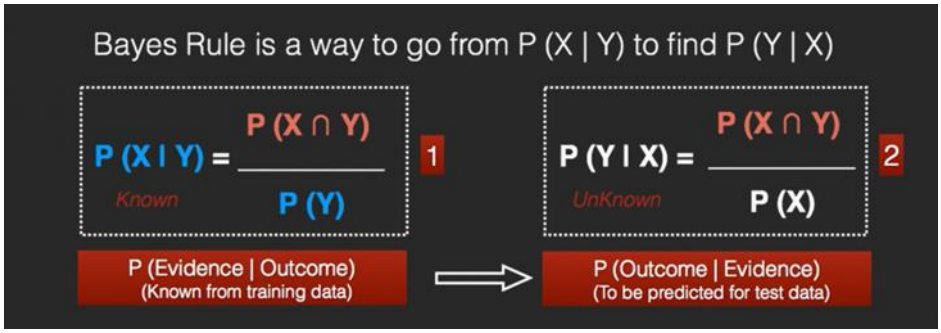
$$P(A\ and\ B)\ =\ P(A).P(B)$$

## Naïve Bayes

Consider, $X = (X_1, X_2, \dots, X_n)$ represent n features. $y$ is the label with $c$ possible values (classes). . The value of $X_i$ is $x_i$ . To make classifications, we need to use $X$ to predict $Y$. In other words, given a data point $X = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n})$, what is the probability $Y$ being $c$. Now, According to the Bayes Theorem:

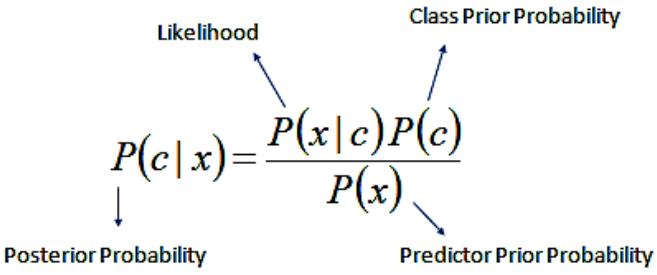
$$\boldsymbol{P(y \mid X) = \frac{P(X \mid y)\, P(y)}{P(X)}}$$

So, The Bayes Rule is a way of going from $P(X \mid y)$, known from the training dataset, to find $P(y \mid X)$.




In more expressive form,

$$\boldsymbol{P(y = c \mid x_1, x_2, \dots, x_n) = \frac{P(x_1, x_2, \dots, x_n \mid y = c)\, P(y = c)}{P(x_1, x_2, \dots, x_n)}}$$




In simple words this can be written as ,

$$posterior = \frac{prior\ \times likelihood}{evidence}$$

$P(y = c \mid x_1, x_2, \dots, x_n)$ is called our posterior probability. By combining our observed information, we are updating our a priori information on probabilities to compute a posterior probability that an observation has class $c$.

$P(y = c)$ is the prior probability of the outcome as class $c$, which is based on training data.

$P(x_1, x_2, \dots, x_n)$ is the probability of the predictor variables. Essentially, based on the training data, what is the probability of each observed combination of predictor variables. When new data comes in, this becomes our *evidence*.

$P(x_1, x_2, \dots, x_n \mid y = c)$ is the conditional probability or *likelihood*, means the probability of a specific combination of features given a class label. *To be able to calculate this, we need extremely large datasets to have an estimate on the probability distribution for all different combinations of feature values. To overcome this issue, Naive Bayes algorithm assumes that all features are independent of each other*.

Under the assumption of features being independent, $P(x_1, x_2, \dots, x_n \mid y = c)$ can be written as $P(x_1 \mid y = c) * P(x_2 \mid y = c) * \dots * P(x_n \mid y = c)$

$$\boldsymbol{P(y = c \mid x_1, x_2, \dots, x_n) = \frac{P(x_1 \mid y = c) * P(x_2 \mid y = c) * \dots * P(x_n \mid y = c) * P(y = c)}{P(x_1) * P(x_2) * \dots * P(x_n)}}$$

The conditional probability for a single feature given the class label (i.e. $\boldsymbol{P(x_i \mid y = c)}$ ) can be more easily estimated from the data.

The probability of a class $\boldsymbol{P(y = c)}$ is very simple to calculate:

$$P(y = c) = \frac{\#\ of\ observations\ with\ label\ c}{\#\ of\ all\ observations}$$

Since, the denominator $P(x_1, x_2, \dots, x_n)$ will be a constant value.

$$\boldsymbol{P(y = c \mid x_1, x_2, \dots, x_n) \propto P(y = c) \prod_{i=1}^{n} P(x_i \mid y = c)}$$

$$\boldsymbol{y = argmax\ P(y = c) \prod_{i=1}^{n} P(x_i \mid y = c)}$$

The denominator $\boldsymbol{P(x_1, x_2, \dots, x_n)}$ can be removed to simplify the equation because it is only normalizing constant i.e., a constant that makes the posterior probability integrate to one.

The algorithm needs to store probability distributions of features for each class independently. For example, if there are 5 classes and 10 features, 50 different probability distributions need to be stored.

As other supervised learning algorithms, Naive Bayes uses features to make a prediction on a target variable. The key difference is that Naive Bayes assumes that features are independent of each other and there is no correlation between features. However, this is not the case in real life. This naive assumption of features being uncorrelated is the reason why this algorithm is called “Naive”.

### **Example:** Playing game (Yes or No) based on weather condition

Let’s build a Naïve Bayes classifier that predicts whether we should play golf given the weather forecast that has been collected over 14 days. It takes four attributes to describe the weather forecast: outlook, temperature, humidity, and the presence or absence of wind.

Furthermore, the label `Play` takes the values yes or no.

| | Predictors | | | | Response |
|---|---|---|---|---|---|
| | **Outlook** | **Temperature** | **Humidity** | **Wind** | **Class Play=Yes Play=No** |
| Day1 | Sunny | Hot | High | Weak | No |
| Day2 | Sunny | Hot | High | Strong | No |
| Day3 | Overcast | Hot | High | Weak | Yes |
| Day4 | Rain | Mild | High | Weak | Yes |
| Day5 | Rain | Cool | Normal | Weak | Yes |
| Day6 | Rain | Cool | Normal | Strong | No |
| Day7 | Overcast | Cool | Normal | Strong | Yes |
| Day8 | Sunny | Mild | High | Weak | No |
| Day9 | Sunny | Cool | Normal | Weak | Yes |
| Day10 | Rain | Mild | Normal | Weak | Yes |
| Day11 | Sunny | Mild | Normal | Strong | Yes |
| Day12 | Overcast | Mild | High | Strong | Yes |
| Day13 | Overcast | Hot | Normal | Weak | Yes |
| Day14 | Rain | Mild | High | Strong | No |

In the ***learning phase***, we compute the table of likelihoods from the training data.

$$P(Outlook = o \mid Play = c\ ), where\ o\ \in\ [Sunny, Overcast, Rainy]\ and\ c\ \in\ [yes, no]$$

| P(Outlook=o\|Class $_{Play=Yes\|No}$) | Frequency | | Probability in Class | |
|---|---|---|---|---|
| **Outlook =** | Play=Yes | Play=No | Play=Yes | Play=No |
| Sunny | 2 | 3 | 2/9 | 3/5 |
| Overcast | 4 | 0 | 4/9 | 0/5 |
| Rain | 3 | 2 | 3/9 | 2/5 |
| | total= 9 | total=5 | | |

Similarly,

$$P(Temperature = t|Play = c), where\ t \in [Hot, Mild, Cool]\ and\ c \in [yes, no]$$

| P(Temperature=t \| Class Play=Yes\|No) | Frequency | | Probability in Class | |
|---|---|---|---|---|
| **Temperature =** | Play=Yes | Play=No | Play=Yes | Play=No |
| Hot | 2 | 2 | 2/9 | 2/5 |
| Mild | 4 | 2 | 4/9 | 2/5 |
| Cool | 3 | 1 | 3/9 | 1/5 |
| | total= 9 | total=5 | | |

$$P(Humidity = h|Play = c), where\ h \in [High, Norma]\ and\ c \in [yes, no]$$

| P(Humidity=h \| Class Play=Yes\|No) | Frequency | | Probability in Class | |
|---|---|---|---|---|
| **Humidity =** | Play=Yes | Play=No | Play=Yes | Play=No |
| High | 3 | 4 | 3/9 | 4/5 |
| Normal | 6 | 1 | 6/9 | 1/5 |
| | total= 9 | total=5 | | |

$$P(Wind = w|Play = c), where\ w \in [Weak, Strong]\ and\ c \in [yes, no]$$

| P(Wind=w \| Class Play=Yes\|No) | Frequency | | Probability in Class | |
|---|---|---|---|---|
| **Wind =** | Play=Yes | Play=No | Play=Yes | Play=No |
| strong | 3 | 3 | 3/9 | 3/5 |
| weak | 6 | 2 | 6/9 | 2/5 |
| | total= 9 | total=5 | | |

We also calculate,

$$P(Play = Yes)\ and\ P(Play = No).$$

| P(Class Play=Yes) & P(Class Play=No) | Frequency | | Probability in Class | |
|---|---|---|---|---|
| Play | Play=Yes | Play=No | Play=Yes | Play=No |
| | 9 | 5 | 9/14 | 5/14 |
| | total= 9 | total=5 | | |

***Prediction Phase:***

Let's say, we get a new instance of the weather condition,

- Outlook = Sunny
- Temperature = Cool
- Humidity = High
- Wind = Strong
- Play = ?? (To predict)

Thus,

$$\boldsymbol{X_{new} = (Outlook = Sunny, Temp. = Cool, Humidity = High, Wind = Strong)}$$

that will have to be classified (i.e., are we going to play golf under the conditions specified by $X_{new}$). This is easily done by looking up the tables we built in the learning phase.

$$\boldsymbol{P(Play = Yes | X_{new}) =}$$

$$[\, P(Outlook = Sunny \mid Play = Yes) \times\ P(\, Temp. = Cool |\, Play = Yes) \times$$

$$P(Humidity = High |\, Play = Yes) \times P(\, Wind = Strong |\, Play = Yes)\ \,] \times P(Play = Yes)$$

$$= 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$

$$\boldsymbol{P(Play = No | X_{new}) =}$$

$$[\ \ P(Outlook = Sunny \mid Play = No) \times\ P(\, Temp. = Cool |\, Play = No) \times$$

$$P(\, Temp. = \mid Play = No) \times\ \ P(\, Temp. = Strong |\, Play = No)\ \,] \times\ P(Play = No)$$

$$= 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0205$$

Since, $P(Play = Yes | X_{new}) < P(Play = No | X_{new})$ , our Naïve Bayes model will classify i.e., predict the new instance $X_{new}$ to be "**No**".

After Normalization (making the posterior probability integrate to one), our model's predicted probabilities are:

$$P(Yes) = \frac{0.0053}{0.0053 + 0.0205} = 0.2054$$

$$P(No) = \frac{0.0205}{0.0053 + 0.0205} = 0.7945$$

So far we've seen the computations when the X's are categorical. But how to compute the probabilities when X is a continuous variable? If the independent variables were continuous then the calculation would have been a bit different.

Let us assume that the X follows a particular distribution, then you can plug in the probability density function of that distribution to compute the probability of likelihoods.

The type of distributions depends on the characteristics of features:

- ✓ For binary features (Y/N, True/False, 0/1): Bernoulli distribution
- ✓ For discrete features (i.e. word counts): Multinomial distribution
- ✓ For continuous features: Gaussian (Normal) distribution

So, there are three types of Naive Bayes model under the scikit-learn library:

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

**MultinomialNB**: 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".

**BernoulliNB**: 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.

It is common to name the Naive Bayes with the distribution of features (i.e. Gaussian Naive Bayes Classifier). For mixed type datasets, a different type of distribution may be required for different features.

## Gaussian Naive Bayes

If you assume the X's follow a Normal (aka Gaussian) Distribution, which is fairly common, we substitute the corresponding probability density of a Normal distribution and call it the Gaussian Naive Bayes. You need just the mean and variance of the X to compute this formula.

$$P(X|Y = c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{-(x-\mu_c)^2}{2\sigma_c^2}}$$

To make the features more Gaussian like, you might consider transforming the variable using something like the Box-Cox to achieve this.

## Naïve Bayes Classifier: Pros & Cons

**Pros:**

The assumption that all features are independent makes Naïve Bayes algorithm very fast compared to complicated algorithms. In some cases, speed is preferred over higher accuracy.

It works well with high-dimensional data such as text classification, email spam detection.

**Cons:**

The assumption that all features are independent is not usually the case in real life so it makes Naive Bayes algorithm less accurate than complicated algorithms. Speed comes at a cost!

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*.

---