

Naive Bayes Classifiers

This article explores Naive Bayes classifiers, a family of algorithms based on Bayes' Theorem. Despite the "naive" assumption of feature independence, these classifiers are widely utilized for their simplicity and efficiency in machine learning. The article delves into theory, implementation, and applications, shedding light on their practical utility despite oversimplified assumptions.

What is Naive Bayes classifiers?

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. To start with, let us consider a dataset.

One of the most simple and effective classification algorithms, the Naïve Bayes classifier aids in the rapid development of machine learning models with rapid prediction capabilities.

Why it is called Naive Bayes?

The "Naive" part of the name indicates the simplifying assumption made by the Naïve Bayes classifier. The classifier assumes that the features used to describe an observation are conditionally independent, given the class label. The "Bayes" part of the name refers to Reverend Thomas Bayes, an 18th-century statistician and theologian who formulated Bayes' theorem.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit("Yes") or unfit("No") for playing golf. Here is a tabular representation of our dataset.

	Outlook	Temperature	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes

[Open In App](#)

	Outlook	Temperature	Humidity	Windy	Play Golf
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are 'Outlook', 'Temperature', 'Humidity' and 'Windy'.
- Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is 'Play golf'.

Assumption of Naive Bayes

The fundamental Naive Bayes assumption is that each feature makes an:

- **Feature independence:** The features of the data are conditionally independent of each other, given the class label.
- **Continuous features are normally distributed:** If a feature is continuous, then it is assumed to be normally distributed within each class.
- **Discrete features have multinomial distributions:** If a feature is discrete, then it is assumed to have a multinomial distribution within each class.
- **Features are equally important:** All features are assumed to contribute equally to the prediction of the class label.

Open In App

- **No missing data:** The data should not contain any missing values.

With relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity or the outlook being 'Rainy' has no effect on the winds. Hence, the features are assumed to be **independent**.
- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can't predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

The assumptions made by Naive Bayes are not generally correct in real-world situations. In fact, the independence assumption is never correct but often works well in practice. Now, before moving to the formula for Naive Bayes, it is important to know about Bayes' theorem.

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

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

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

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
- $P(A)$ is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- $P(B)$ is Marginal Probability: Probability of Evidence.
- $P(A|B)$ is a posteriori probability of B, i.e. probability of event after evidence is seen.
- $P(B|A)$ is Likelihood probability i.e the likelihood that a hypothesis will come true based on the evidence.

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where, y is class variable and X is a dependent feature vector (of size n) where:

Open In App

$$X = (x_1, x_2, x_3, \dots, x_n)$$

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

X = (Rainy, Hot, High, False)

y = No

So basically, $P(y|X)$ here means, the probability of “Not playing golf” given that the weather conditions are “Rainy outlook”, “Temperature is hot”, “high humidity” and “no wind”.

With relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent. For example, the temperature being ‘Hot’ has nothing to do with the humidity or the outlook being ‘Rainy’ has no effect on the winds. Hence, the features are assumed to be **independent**.
- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can’t predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

Now, its time to put a naive assumption to the Bayes’ theorem, which is, **independence** among the features. So now, we split **evidence** into the independent parts.

Now, if any two events A and B are independent, then,

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

Hence, we reach to the result:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)}$$

which can be expressed as:

$$P(y|x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i|y)}{P(x_1)P(x_2)\dots P(x_n)}$$

Now, as the denominator remains constant for a given input, we can remove that term:

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

Open In App

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability. This can be expressed mathematically as:

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y)$$

So, finally, we are left with the task of calculating $P(y)$ and $P(x_i|y)$.

Please note that $P(y)$ is also called class probability and $P(x_i|y)$ is called conditional probability.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$.

Let us try to apply the above formula manually on our weather dataset. For this, we need to do some precomputations on our dataset.

We need to find $P(x_i|y_j)$ for each x_i in X and y_j in y . All these calculations have been demonstrated in the tables below:

Outlook

	Yes	No	P(yes)	P(no)
Sunny	3	2	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%


So, in the figure above, we have calculated $P(x_i|y_j)$ for each x_i in X and y_j in y manually in the tables 1-4. For example, probability of playing golf given that the temperature is cool, i.e $P(\text{temp.} = \text{cool} \mid \text{play golf} = \text{Yes}) = 3/9$.

Also, we need to find class probabilities $P(y)$ which has been calculated in the table 5. For example, $P(\text{play golf} = \text{Yes}) = 9/14$.

So now, we are done with our pre-computations and the classifier is ready!

Let us test it on a new set of features (let us call it today):

```
today = (Sunny, Hot, Normal, False)
```


$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

and probability to not play golf is given by:

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

Since, $P(today)$ is common in both probabilities, we can ignore $P(today)$ and find proportional probabilities as:

$$P(Yes|today) \propto \frac{3}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} \approx 0.02116$$

and

$$P(No|today) \propto \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \cdot \frac{5}{14} \approx 0.0068$$

Now, since

$$P(Yes|today) + P(No|today) = 1$$

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.02116}{0.02116+0.0068} \approx 0.0237$$

and

$$P(No|today) = \frac{0.0068}{0.0141+0.0068} \approx 0.33$$

Since

$$P(Yes|today) > P(No|today)$$

So, prediction that golf would be played is 'Yes'.

[Open In App](#)

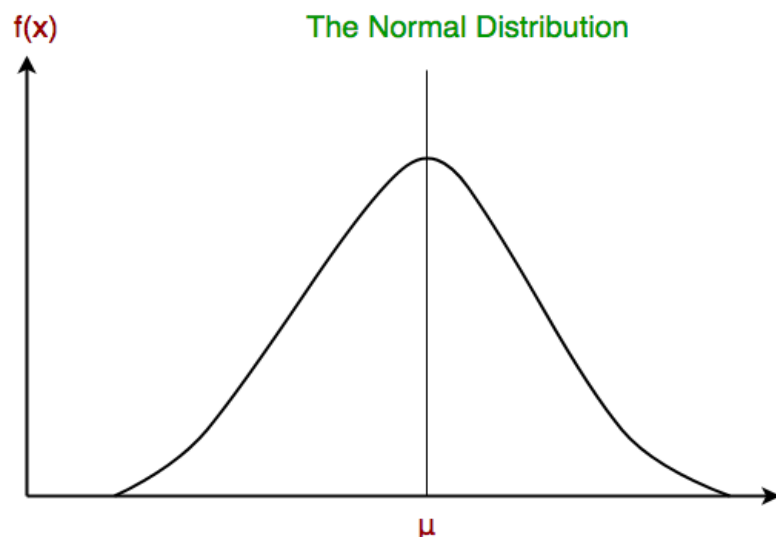
The method that we discussed above is applicable for discrete data. In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$.

Types of Naive Bayes Model

There are three types of Naive Bayes Model:

Gaussian Naive Bayes classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called [Normal distribution](#). When plotted, it gives a bell shaped curve which is symmetric about the mean of the



feature values as shown below:

Updated table of prior probabilities for outlook feature is as following:

The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

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

Now, we look at an implementation of Gaussian Naive Bayes classifier using scikit-learn.

	Yes	No	P(Yes)	P(No)
Sunny	3	2	3/9	2/5
Rainy	4	0	4/9	0/5
Overcast	2	3	2/9	3/5
Total	9	5	100%	100%

Python

```
# load the iris dataset
from sklearn.datasets import load_iris
iris = load_iris()

# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target

# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,

# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)

# comparing actual response values (y_test) with predicted response value
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_scor
```

Output:

Open In App

Multinomial Naive Bayes

Feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution. This is the event model typically used for document classification.

Bernoulli Naive Bayes

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence(i.e. a word occurs in a document or not) features are used rather than term frequencies(i.e. frequency of a word in the document).

Advantages of Naive Bayes Classifier

- Easy to implement and computationally efficient.
- Effective in cases with a large number of features.
- Performs well even with limited training data.

Disadvantages of Naive Bayes Classifier

- Assumes that features are independent, which may not always hold in real-world data.
- Can be influenced by irrelevant attributes.
- May assign zero probability to unseen events, leading to poor generalization.

Applications of Naive Bayes Classifier

- **Spam Email Filtering:** Classifies emails as spam or non-spam based on features.
- **Text Classification:** Used in sentiment analysis, document categorization, and topic classification.
- **Medical Diagnosis:** Helps in predicting the likelihood of a disease based on symptoms.
- **Credit Scoring:** Evaluates creditworthiness of individuals for loan approval.
- **Weather Prediction:** Classifies weather conditions based on various factors.

As we reach to the end of this article, here are some important points to ponder upon:

- In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering.

Open In App

They require a small amount of training data to estimate the necessary parameters.

- Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

Conclusion

In conclusion, Naive Bayes classifiers, despite their simplified assumptions, prove effective in various applications, showcasing notable performance in document classification and spam filtering. Their efficiency, speed, and ability to work with limited data make them valuable in real-world scenarios, compensating for their naive independence assumption.

Frequently Asked Questions(FAQs)

1. What is Naive Bayes real example?

Naive Bayes is a simple probabilistic classifier based on Bayes' theorem. It assumes that the features of a given data point are independent of each other, which is often not the case in reality. However, despite this simplifying assumption, Naive Bayes has been shown to be surprisingly effective in a wide range of applications.

2. Why is it called Naive Bayes?

Naive Bayes is called "naive" because it assumes that the features of a data point are independent of each other. This assumption is often not true in reality, but it does make the algorithm much simpler to compute.

3. What is an example of a Bayes classifier?

A Bayes classifier is a type of classifier that uses Bayes' theorem to compute the probability of a given class for a given data point. Naive Bayes is one of the most common types of Bayes classifiers.

4. What is better than Naive Bayes?

Open In App

There are several classifiers that are better than Naive Bayes in some situations. For example, logistic regression is often more accurate than Naive Bayes, especially when the features of a data point are correlated with each other.

5. Can Naive Bayes probability be greater than 1?

No, the probability of an event cannot be greater than 1. The probability of an event is a number between 0 and 1, where 0 indicates that the event is impossible and 1 indicates that the event is certain.



Article Tags : [Computer Subject](#) [Machine Learning](#) [Python](#)

Recommended Articles

1. [Applying Multinomial Naive Bayes to NLP Problems](#)
2. [Complement Naive Bayes \(CNB\) Algorithm](#)
3. [Classification of Text Documents using the approach of Naïve Bayes](#)
4. [Bernoulli Naive Bayes](#)
5. [Gaussian Naive Bayes](#)
6. [Multinomial Naive Bayes](#)
7. [ML | Naive Bayes Scratch Implementation using Python](#)
8. [Gaussian Naive Bayes using Sklearn](#)
9. [ML | Dummy classifiers using sklearn](#)
10. [How to use built-in image classifiers of visual recognition module using IBM watson?](#)
11. [How to build custom image classifiers using IBM Watson?](#)
12. [Passive Aggressive Classifiers](#)
13. [Decision Tree Classifiers in R Programming](#)
14. [Probability Calibration of Classifiers in Scikit Learn](#)
15. [Building Naive Bayesian classifier with WEKA](#)

Open In App