



Log ind på GeeksforGeeks med Google



Chris Kramer

chris.mondrup.kramer@gmail.com

Fortsæt som Chris

Når du opretter en konto, deler Google dit navn, din mailadresse og dit profilbillede med GeeksforGeeks. Se [privatlivspolitikken](#) og servicevilkårene for GeeksforGeeks.

Naive Bayes Classifiers

[Read](#)[Discuss\(20+\)](#)[Courses](#)[Practice](#)[Vid](#)

This article discusses the theory behind the Naive Bayes classifiers and their implementation.

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.

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.

AD

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
4	Sunny	Cool	Normal	False	Yes

	Outlook	Temperature	Humidity	Windy	Play Golf
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

Data Types Control Flow Functions List String Set Tuple Dictionary Oops Exception Handling Py

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:

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

- independent
- equal

contribution to the outcome.

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.

Note: 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).
- $P(A|B)$ is a posteriori probability of B, i.e. probability of event after evidence is seen.

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:

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

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".

Naive assumption

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:

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

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	2	3	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 $P(\text{temp.} = \text{cool} | \text{play golf} = \text{Yes}) = 3/9$.

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)

So, probability of playing golf is given by:

$$P(Yes|today) = \frac{P(Sunny|Outlook|Yes)P(Hot|Temperature|Yes)P(Normal|Humidity|Yes)P(No|Wind|Yes)P(Yes)}{P(today)}$$

and probability to not play golf is given by:

$$P(No|today) = \frac{P(Sunny|Outlook|No)P(Hot|Temperature|No)P(Normal|Humidity|No)P(No|Wind|No)P(No)}{P(today)}$$

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

$$P(Yes|today) \propto \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} \approx 0.0141$$

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):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

and

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

Since

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

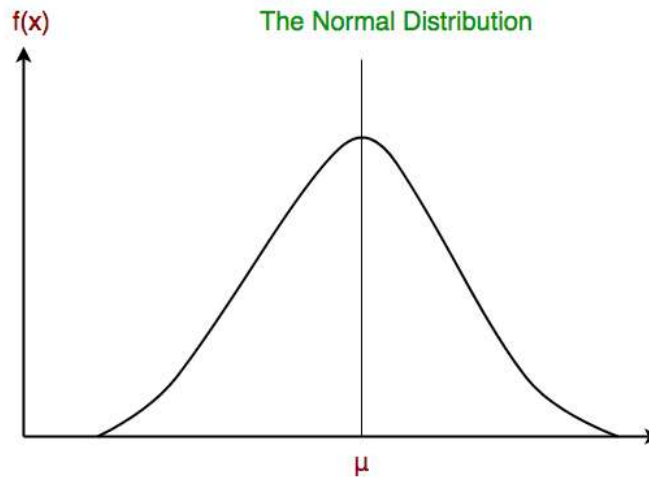
So, prediction that golf would be played is 'Yes'.

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 $P(x_i | y)$.

Now, we discuss one of such classifiers here.

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
Overcast	4	0	4/9	0/5
Rainy	2	3	2/9	3/5
Total	9	5	100%	100%

```
# 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, random_state=1)

# 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 values (y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test,
y_pred)*100)
```

Output:

```
Gaussian Naive Bayes model accuracy(in %): 95.0
```

Other popular Naive Bayes classifiers are:

- **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).

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. 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.

References:

- https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- http://gerardnico.com/wiki/data_mining/naive_bayes
- http://scikit-learn.org/stable/modules/naive_bayes.html

This blog is contributed by [Nikhil Kumar](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 28 Mar, 2023

153

Similar Reads

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

Related Tutorials

1. Computer Science and Programming For Kids
2. Pandas AI: The Generative AI Python Library
3. OpenAI Python API - Complete Guide
4. Top Computer Vision Projects (2023)
5. Python for Kids - Fun Tutorial to Learn Python Programming

Like

153

[Previous](#)[Python Input Methods for Competitive Programming](#)[Next](#)[Amazing hacks of Python](#)

Article Contributed By :

**GeeksforGeeks**

Vote for difficulty

Current difficulty : **Easy**

Easy

Normal

Medium

Hard

Expert

Improved By :

[Sambhav_Khurana](#)[shivamraj74](#)[raunaksrivastava22](#)[rkbhola5](#)[germanshepherd48](#)[ishanshekharsigning](#)

Article Tags :

[Computer Subject](#)[Machine Learning](#)[Python](#)

Practice Tags :

[Machine Learning](#)[python](#)

Improve Article

Report Issue

**GeeksforGeeks**

A-143, 9th Floor, Sovereign Corporate
Tower, Sector-136, Noida, Uttar Pradesh -
201305

feedback@geeksforgeeks.org

Company

[About Us](#)[Legal](#)[Careers](#)[In Media](#)

Explore

[Job-A-Thon For Freshers](#)[Job-A-Thon For Experienced](#)[GfG Weekly Contest](#)[Offline Classes \(Delhi/NCR\)](#)

[Contact Us](#)[DSA in JAVA/C++](#)[Advertise with us](#)[Master System Design](#)[Master CP](#)

Languages

[Python](#)[Java](#)[C++](#)[PHP](#)[GoLang](#)[SQL](#)[R Language](#)[Android Tutorial](#)

Algorithms

[Sorting](#)[Searching](#)[Greedy](#)[Dynamic Programming](#)[Pattern Searching](#)[Recursion](#)[Backtracking](#)

Computer Science

[GATE CS Notes](#)[Operating Systems](#)[Computer Network](#)[Database Management System](#)[Software Engineering](#)[Digital Logic Design](#)[Engineering Maths](#)

Data Science & ML

[Data Science With Python](#)[Data Science For Beginner](#)[Machine Learning Tutorial](#)[Maths For Machine Learning](#)[Pandas Tutorial](#)[NumPy Tutorial](#)

Data Structures

[Array](#)[String](#)[Linked List](#)[Stack](#)[Queue](#)[Tree](#)[Graph](#)

Web Development

[HTML](#)[CSS](#)[JavaScript](#)[Bootstrap](#)[ReactJS](#)[AngularJS](#)[NodeJS](#)

Python

[Python Programming Examples](#)[Django Tutorial](#)[Python Projects](#)[Python Tkinter](#)[OpenCV Python Tutorial](#)[Python Interview Question](#)

DevOps

[Git](#)[AWS](#)[Docker](#)[Kubernetes](#)[Azure](#)[GCP](#)

NLP Tutorial

Deep Learning Tutorial

Competitive Programming

Top DSA for CP

Top 50 Tree Problems

Top 50 Graph Problems

Top 50 Array Problems

Top 50 String Problems

Top 50 DP Problems

Top 15 Websites for CP

Interview Corner

Company Wise Preparation

Preparation for SDE

Experienced Interviews

Internship Interviews

Competitive Programming

Aptitude Preparation

Commerce

Accountancy

Business Studies

Economics

Management

Income Tax

Finance

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

Aptitude Questions

SSC CGL Practice Papers

System Design

What is System Design

Monolithic and Distributed SD

Scalability in SD

Databases in SD

High Level Design or HLD

Low Level Design or LLD

Top SD Interview Questions

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

UPSC

Polity Notes

Geography Notes

History Notes

Science and Technology Notes

Economics Notes

Important Topics in Ethics

UPSC Previous Year Papers

Write & Earn

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship