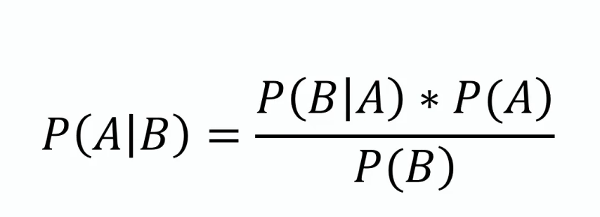
Introduction

**Naive Bayes** is a machine learning algorithm that is used by data scientists for classification. The naive Bayes algorithm works based on the Bayes theorem. Before explaining Naive Bayes, first, we should discuss Bayes Theorem. Bayes theorem is used to find the probability of a hypothesis with given evidence. This beginner-level article intends to introduce you to the Naive Bayes algorithm and explain its underlying concept and implementation.



In this equation, using Bayes theorem, we can find the probability of A, given that B occurred. A is the hypothesis, and B is the evidence.

P(B|A) is the probability of B given that A is True.

P(A) and P(B) are the independent probabilities of A and B.

**Learning Objectives**

* Learn the concept behind the Naive Bayes algorithm.
* See the steps involved in the naive Bayes algorithm
* Practice the step-by-step implementation of the algorithm.

*This article was published as a part of the*[*Data Science Blogathon*](https://datahack.analyticsvidhya.com/contest/data-science-blogathon-4/)*.*

[](https://www.analyticsvidhya.com/events/datahour/datahour-setup-data-warehousing-in-microsoft-fabric)

Table of contents

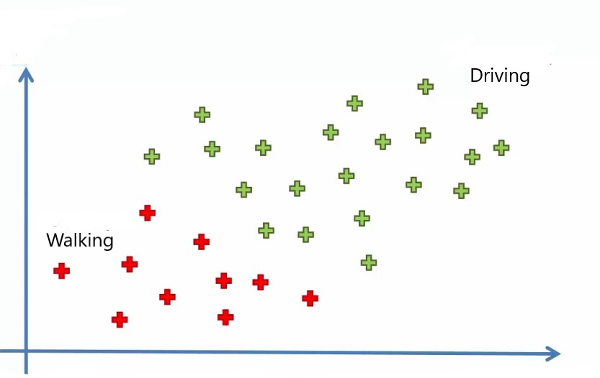
What Is the Naive Bayes Classifier Algorithm?

The Naive Bayes classifier algorithm is a machine learning technique used for classification tasks. It is based on Bayes’ theorem and assumes that features are conditionally independent of each other given the class label. The algorithm calculates the probability of a data point belonging to each class and assigns it to the class with the highest probability.

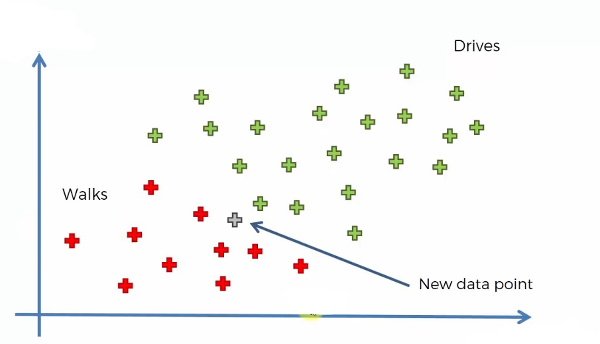
Naive Bayes is known for its simplicity, efficiency, and effectiveness in handling high-dimensional data. It is commonly used in various applications, including text classification, spam detection, and sentiment analysis.

Naive Bayes Theorem: The Concept Behind the Algorithm

Let’s understand the concept of the Naive Bayes Theorem and how it works through an example. We are taking a case study in which we have the dataset of employees in a company, our aim is to create a model to find whether a person is going to the office by driving or walking using the salary and age of the person.



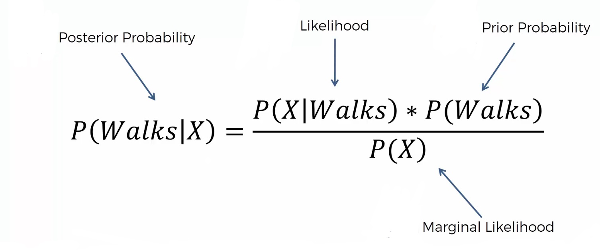
In the above image, we can see 30 data points in which red points belong to those who are walking and green belong to those who are driving. Now let’s add a new data point to it. Our aim is to find the category that the new point belongs to



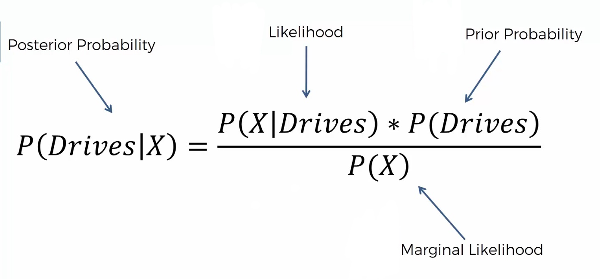
Note that we are taking age on the X-axis and Salary on the Y-axis. We are using the Naive Bayes algorithm to find the category of the new data point. For this, we have to find the posterior probability of walking and driving for this data point. After comparing, the point belongs to the category having a higher probability.

In the above image, we can see 30 data points in which red points belong to those who are walking and green belong to those who are driving. Now let’s add a new data point to it. Our aim is to find the category that the new point belongs to

The posterior probability of walking for the new data point is:



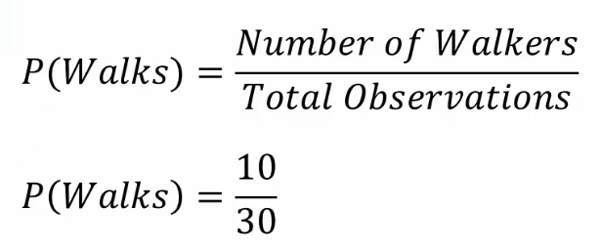
and that for the driving is:



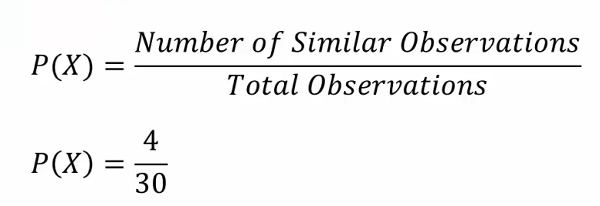
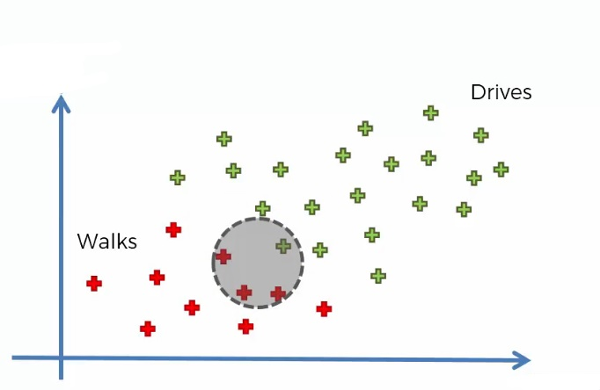
Steps Involved in the Naive Bayes Classifier Algorithm

**Step 1**: We have to find all the probabilities required for the Bayes theorem for the calculation of posterior probability.

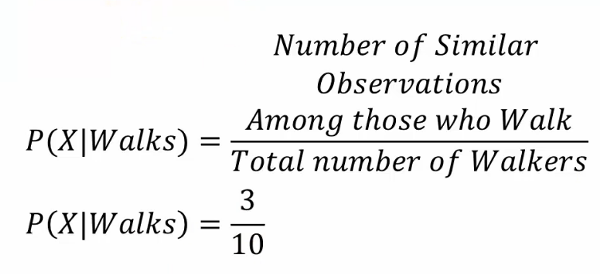
P(Walks) is simply the probability of those who walk among all.



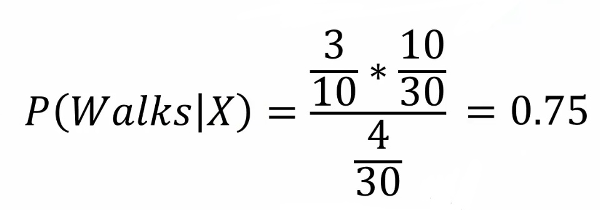
In order to find the marginal likelihood, P(X), we have to consider a circle around the new data point of any radii, including some red and green points.



P(X|Walks) can be found by:



Now we can find the posterior probability using the Bayes theorem,



**Step 2**: Similarly, we can find the posterior probability of Driving, and it is 0.25

**Step 3**: Compare both posterior probabilities. When comparing the posterior probability, we can find that P(walks|X) has greater values, and the new point belongs to the walking category.

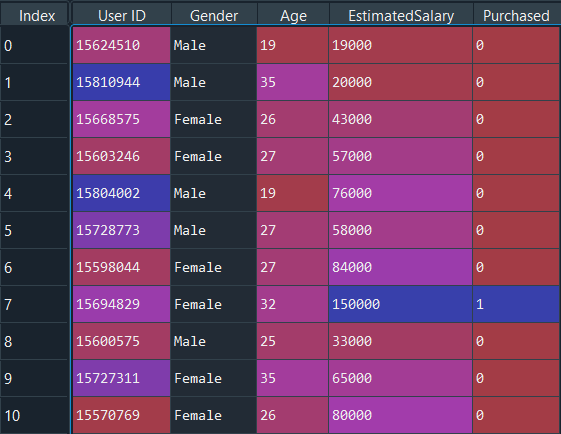


Source: Unsplash

Implementation of Naive Bayes in Python Programming

Now let’s implement Naive Bayes step by step using the python programming language

We are using the Social network ad dataset. The dataset contains the details of users on a social networking site to find whether a user buys a product by clicking the ad on the site based on their salary, age, and gender.



**Step 1: Importing the libraries**

Let’s start the programming by importing the essential libraries required.

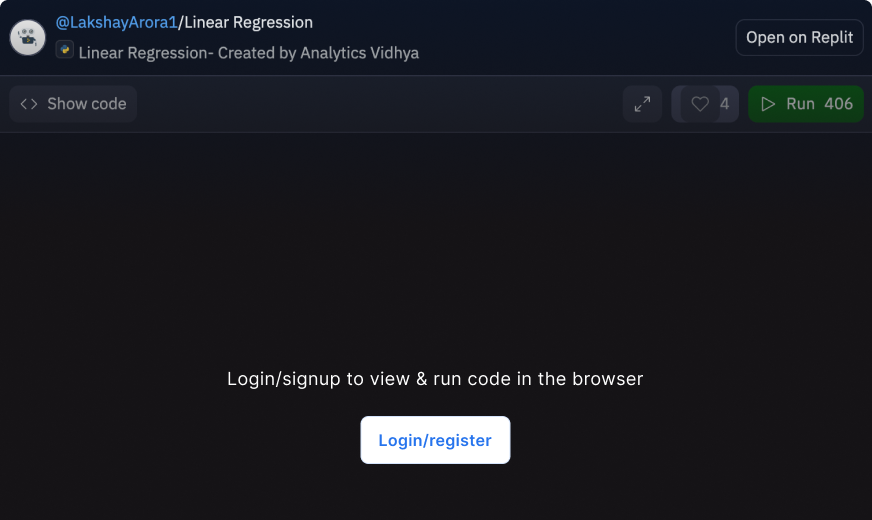
**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

**import** sklearn

**Step 2: Importing the dataset**  
**Python Code:**

[](https://id.analyticsvidhya.com/auth/login/?next=https://www.analyticsvidhya.com/blog/2021/01/a-guide-to-the-naive-bayes-algorithm/?&utm_source=coding-window-blog&source=coding-window-blog)

Since our dataset contains character variables, we have to encode it using LabelEncoder.

from sklearn.preprocessing **import** LabelEncoder

le = LabelEncoder()

X[:,0] = le.fit\_transform(X[:,0])

**Step 3: Train test splitting**

We are splitting our data into train and test datasets using the scikit-learn library. We are providing the test size as 0.20, which means our training data contains 320 training sets, and the test sample contains 80 test sets.

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = **train\_test\_split**(X, y, test\_size = 0.20, random\_state = 0)

**Step 4: Feature scaling**

Next, we are doing **feature scaling** to the training and test set of independent variables.

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

**Step 5: Training the Naive Bayes model on the training set**

from sklearn.naive\_bayes **import** GaussianNB

classifier = GaussianNB()

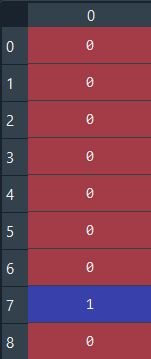
classifier.fit(X\_train, y\_train)

Let’s predict the test results

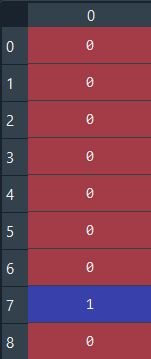
y\_pred = classifier.predict(X\_test)

Predicted and actual value

y\_pred



y\_test



For the first 8 values, both are the same. We can evaluate our matrix using the confusion matrix and accuracy score by comparing the predicted and actual test values

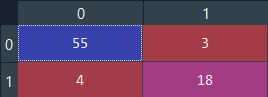
from sklearn.metrics **import** confusion\_matrix,accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

ac = accuracy\_score(y\_test,y\_pred)

**confusion matrix**

ac – **0.9125**



Accuracy is good. Note that you can achieve better results for this problem using different algorithms.

Full Python Tutorial

# Importing the libraries

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

# Importing the dataset

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 0)

# Feature Scaling

**from** sklearn.preprocessing **import** StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Training the Naive Bayes model on the Training set

**from** sklearn.naive\_bayes **import** GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score

ac = accuracy\_score(y\_test,y\_pred)

cm = confusion\_matrix(y\_test, y\_pred)

What Are the Assumptions Made by the Naive Bayes Algorithm?

There are several variants of Naive Bayes, such as Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes. Each variant has its own assumptions and is suited for different types of data. Here are some assumptions that the Naive Bayers algorithm makes:

1. The main assumption is that it assumes that the features are conditionally independent of each other.
2. Each of the features is equal in terms of weightage and importance.
3. The algorithm assumes that the features follow a normal distribution.
4. The algorithm also assumes that there is no or almost no correlation among features.

Conclusion

The naive Bayes algorithm is a powerful and widely-used machine learning algorithm that is particularly useful for classification tasks. This article explains the basic math behind the Naive Bayes algorithm and how it works for binary classification problems. Its simplicity and efficiency make it a popular choice for many data science applications. we have covered most concepts of the algorithm and how to implement it in Python. Hope you liked the article, and do not forget to practice algorithms.

**Key Takeaways**

* Naive Bayes is a probabilistic classification algorithm(binary o multi-class) that is based on Bayes’ theorem.
* There are different variants of Naive Bayes, which can be used for different tasks and can even be used for regression problems.
* Naive Bayes can be used for a variety of applications, such as spam filtering, sentiment analysis, and recommendation systems.

Frequently Asked Questions

***The media shown in this article are not owned by Analytics Vidhya and is used at the Author’s discretion.***

[blogathon](https://www.analyticsvidhya.com/blog/tag/blogathon/)[Naive Bayes](https://www.analyticsvidhya.com/blog/tag/naive-bayes/)

[](https://www.analyticsvidhya.com/blog/author/surbhi_22/)

[Surabhi S](https://www.analyticsvidhya.com/blog/author/surbhi_22/)14 Jul 2023

[Algorithm](https://www.analyticsvidhya.com/blog/category/algorithm/)[Beginner](https://www.analyticsvidhya.com/blog/category/beginner/)[Classification](https://www.analyticsvidhya.com/blog/category/classification/)[Machine Learning](https://www.analyticsvidhya.com/blog/category/machine-learning/)[Maths](https://www.analyticsvidhya.com/blog/category/maths/)

Frequently Asked Questions

Q1. When should we use a naive Bayes classifier?

A. The naive Bayes classifier is a good choice when you want to solve a binary or multi-class classification problem when the dataset is relatively small and the features are conditionally independent. It is a fast and efficient algorithm that can often perform well, even when the assumptions of conditional independence do not strictly hold. Due to its high speed, it is well-suited for real-time applications. However, it may not be the best choice when the features are highly correlated or when the data is highly imbalanced.

Q2. What is the difference between Bayes Theorem and Naive Bayes Algorithm?

Q3. Is Naive Bayes a regression technique or classification technique?

Responses From Readers

Top of Form

Submit reply

Bottom of Form



Onkar16 Jan, 2021

Awesome explanation  This will clear all the doubts and its very helful for newbies. Keep up the good work



Ryan19 Jan, 2021

Hi, Great post;) I would like to ask when estimating the marginal likelihood P(X), we need to draw a circle around the new data, how should we choose the radius in order to increase accuracy of the estimation? And how is the radius or metric used going too affect the accuracy? Is there any book you can recommend for this topic? Thank you so much.



Sania26 Apr, 2021

Thanks Surbhi! Easy to understand.



Karim