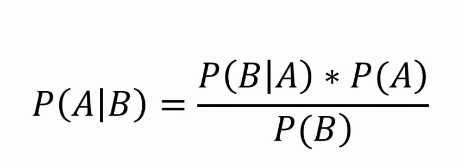
**Naive Bayes** is a classification algorithm that works based on the Bayes theorem. Bayes theorem is used to find the probability of a hypothesis with given evidence.



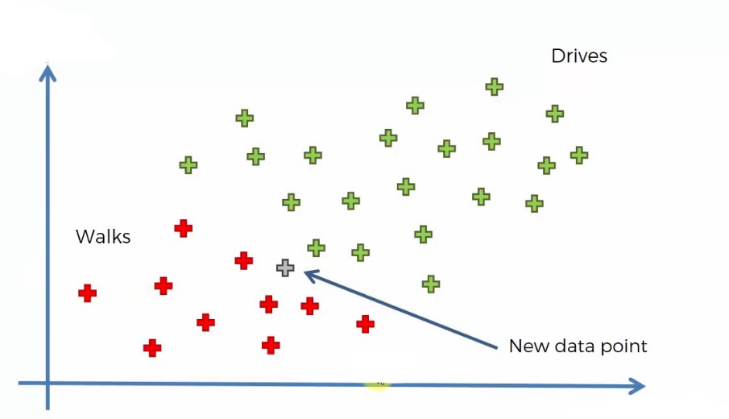
Using Bayes theorem we can find the probability of A, given that B occurred.

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) is the independent probabilities of A and B.

A dataset of employees, our aim is to create a model to find whether a person is going to the office by driving or walking using **salary and age** of the person.

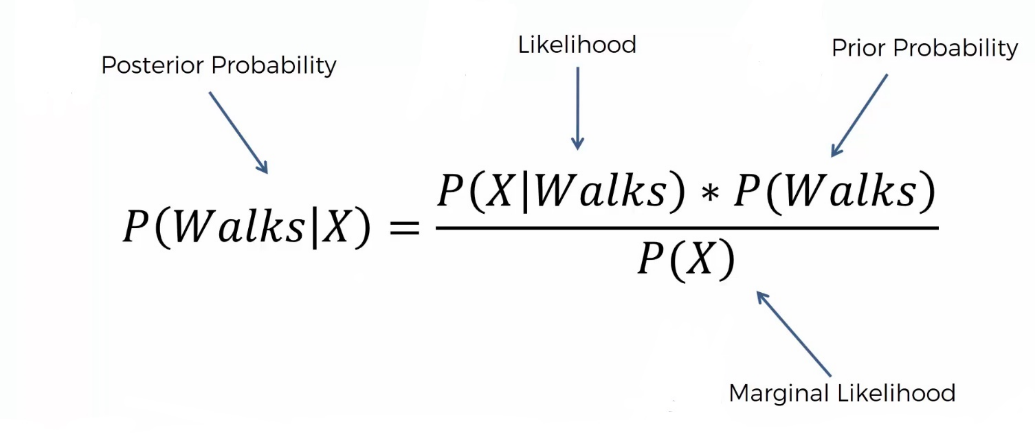


Our aim is to find the category that the new point belongs to.

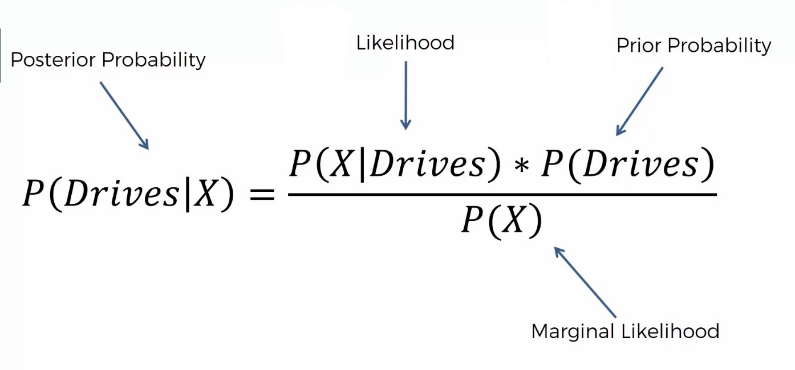
Age on the X-axis and Salary on the Y-axis.

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.

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



also for the driving is :



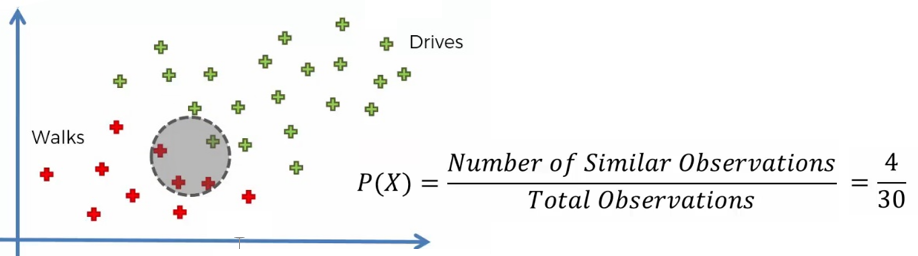
**Steps involved in Naive Bayes algorithm**

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

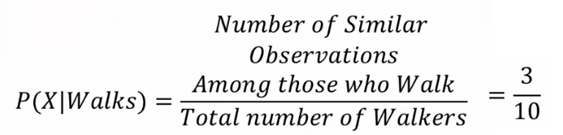
P(Walks) is simply the probability of those who walk among all



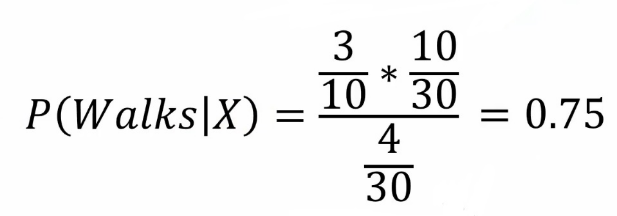
 In order to find the marginal likelihood, P(X), we have to consider a circle around the new data point of any radii including some red and green points.



P(X|Walks) can be found by :



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

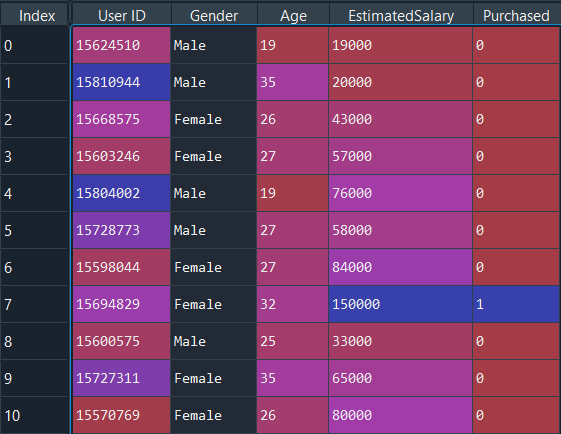


**Step 2**: Similarly we can find the posterior probability of Driving, and it is 0.25

**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.

Implementation of Naive Bayes

We are using the Social network ad dataset. The dataset contains the details of users in 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.



import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import sklearn

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

X = dataset.iloc[:, [1, 2, 3]].values

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

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

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

**Split**

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)

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)

**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)

|  |  |
| --- | --- |
| y\_pred predicted values | y\_test actual values to test (check) |
| train | test data |

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

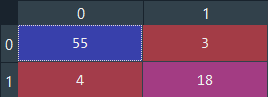
from sklearn.metrics import confusion\_matrix,accuracy\_score

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

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

**confusion matrix** –

ac – **0.9125**



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)