***NAÏVE BAYSE ALGORITHM***

It is a probability based classification algorithm.

Naive Bayes is one of the simplest machine learning algorithms for classification. We'll cover an introduction to Naive Bayes, and implement it in Python.

Envision this: a power administrator might want to supply explicit units of electric flow to different plant divisions in view of their past patterns of force utilization. To improve on the cycle, she/he intends to sort the manufacturing plant divisions into three gatherings low, medium, and high-power buyers in view of which he knows how much power to supply. Issues of this sort normally fall under prescient grouping demonstrating, or are basically known as characterization type issues. Innocent Bayes is one of the least difficult AI calculations that has forever been a #1 for characterizing information.

Naïve Bayes depends on Bayes theorem, which was proposed by Reverend Thomas Bayes, thinking back to the 1760's. Its fame has soar somewhat recently and the calculation is broadly being utilized to handle issues across the scholarly world, government, and business. A Naive Bayes classifier is a blend of various positive characteristics in commonsense AI. We'll reveal insight into the instincts behind this further on. We should begin by first grasping the working of a Guileless Bayes calculation, and afterward carrying out it in Python utilizing the scikit-learn library.

In this blog we'll learn about the following topics:

* Introduction to Naive Bayes Algorithm
* Conditional Probability and Bayes Theorem
* Working of Naive Bayes Algorithm
* Applications of Naive Bayes
* Implementing Naive Bayes with Scikit-Learn
* Pros and Cons
* Summary

INTRODUCTION TO NAÏVE BAYES ALGORITHM:

Naive Bayes falls under the umbrella of administered AI calculations that are fundamentally utilized for order. In this specific situation, "administered" lets us know that the calculation is prepared with both info highlights and straight out yields (i.e., the information incorporates the right wanted yield for each point, which the calculation ought to anticipate).

Yet, for what reason is the calculation called "guileless"? This is on the grounds that the classifier accepts that the information includes that go into the model are free of one another. Subsequently, transforming one info highlight won't influence any of the others. It's thusly guileless as in this supposition might be valid, and it most presumably isn't.We'll examine the guilelessness of this calculation exhaustively in the Working of Naive Bayes Calculation area. Before that, we should momentarily take a gander at why this calculation is straightforward, yet strong, and simple to execute. One of the huge benefits of Guileless Bayes is that it utilizes a probabilistic methodology; every one of the calculations are finished on the fly continuously, and yields are created promptly. While dealing with a lot of information, this gives Credulous Bayes a high ground over conventional grouping calculations like SVMs and Troupe strategies.

We should get everything rolling by getting a hang of the hypothesis fundamental for figuring out Innocent Bayes.

PROBABILITY, CONDITIONAL PROBABILITY, AND BAYES THEOREM:

Probability is the foundation upon which Naive Bayes has been built. Let’s get down to the nitty-gritty of what probability is all about.

**What Is Probability?**

Probability is one of the pivotal parts of math that assists us with foreseeing how likely an occasion X is to happen thinking about the absolute of expected results. To make sense of this in a more exact manner, consider an instance of making a forecast on whether you would head off to college on a particular day. Here there are two potential results: join in or skip. Consequently, the probability of you joining in or skipping school is ½. Numerically, likelihood can be addressed by the accompanying condition:

***Probability of an Event = Number of Favorable Events / Total Number of Outcomes***

***0 <= Probability of an Event <= 1***

The "ideal occasions" mean the event(s) for which you need the likelihood of their happening. Likelihood generally lies in the scope of 0 to 1, with 0 importance there's no chance of that occasion occurring, and 1 significance there's a 100 percent plausibility it will work out.

Conditional Probability is a subset of Probability, which restricts the idea of probability to create a dependency on a specific event; let’s understand it in the next section.

CONDITIONAL PROBABILITY :

Conditional Probability is registered for at least two occasions. Take two occasions, An and B. The restrictive likelihood of occasion B is characterized as the likelihood that occasion B will happen given the information that occasion A has previously occurred. It is addressed as P(B|A), and numerically by the formula:

**P(B|A) = P(A and B)/P(A)**

Let’s look at an example to understand the concept clearly. Consider you have two containers, Crate A and Crate B, which are loaded up with both Apples and Mangoes. The organization subtleties are displayed underneath:



Fig 1:Image from Adobe Stock

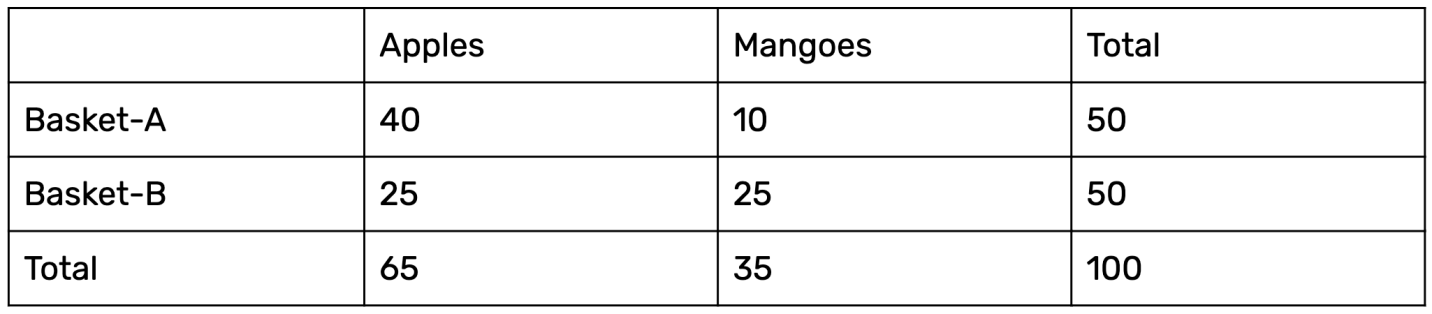


Fig 2:Image from Paperspace

P(Apple/Basket-B) = P(Apple and Basket-B)/ P(Basket-B)

= 25/50

This probability reflects first choosing Basket-B and then picking an apple from it.In the next section we'll look at Bayes theorem, which follows in the footsteps of Conditional probability.

BAYES RULE:

Bayes Rule revolves around the concept of deriving a hypothesis (H) from the given evidence (E). It relates two notions: the probability of the hypothesis before getting the evidence, P(H), and the probability of the hypothesis after getting the evidence, P(H|E). In general, it’s given by the following equation:

P(H|E) = (P(E|H) \* P(H)) / P(E)

Which tells us:

How often H happens given that E happens, written as P(H|E)

When we know:

How often E happens given that H happens, written as P(E|H)

How likely A is on its own, written as P(H)

How likely B is on its own, written as P(E)

The Bayes Rule is a way of going from P(E|H) to finding P(H|E). In simple terms, it provides a way to calculate the probability of a hypothesis given the evidence.

BAYES RULE FROM A MACHINE LEARNING PERSPECTIVE:

We usually have training data to teach our model, and validation data to evaluate the model and make new predictions. Let’s call our input features as evidence, and labels as outcomes in the training data. Using conditional probability, we calculate the probability of the evidence given the outcomes, denoted as P(Evidence|Outcome). Our goal now is to find the probability of an outcome with respect to the evidence, denoted as P(Outcome|Evidence). Let’s define Bayes Rule for both, P(Evidence|Outcome) and P(Outcome|Evidence).

Consider X to denote Evidence and Y to denote Outcome.

P(Evidence|Outcome) is thus P(X|Y), and is represented as follows:

P(X|Y) = (P(Y|X) \* P(X)) / P(Y) (To be estimated from the training data.)

P(Outcome|Evidence) is P(Y|X), and is represented as follows:

P(Y|X) = (P(X|Y) \* P(Y)) / P(X) (To be predicted from the test data.)

If the problem at hand has two outcomes, then we calculate the probability of each outcome and say the highest one wins. But what if we have multiple input features? This is when Naive Bayes comes into the picture; let’s discuss this algorithm in the next section.

The Bayes Rule provides the formula to compute the probability of output (Y) given the input (X). In real-world problems, unlike the hypothetical assumption of having a single input feature, we have multiple X variables. When we can assume the features are independent of each other, we extend the Bayes Rule to what is called Naive Bayes.

WORKING OF NAÏVE BAYES ALGORITHM:

Consider a case where there are multiple inputs (X1, X2, X3,... Xn). We predict the outcome (Y) using the Naive Bayes equation as follows:

P(Y=k | X1...Xn) = ( P(X1 | Y=k) \* P(X2 | Y=k) \* P(X3 | Y=k) \* ....\* P(Xn | Y=k) ) \* P(Y=k) / P(X1)\*P(X2)\*P(X3)\*P(Xn)

In the above formula:

P(Y=k | X1...Xn) is called the Posterior Probability, which is the probability of an outcome given the evidence.

P(X1 | Y=k) \* P(X2 | Y=k) \* ... P(Xn | Y=k) is the probability of the likelihood of evidence.

P(Y=k) is the Prior Probability.

P(X1)\*P(X2)\*P(Xn) is the probability of the evidence.

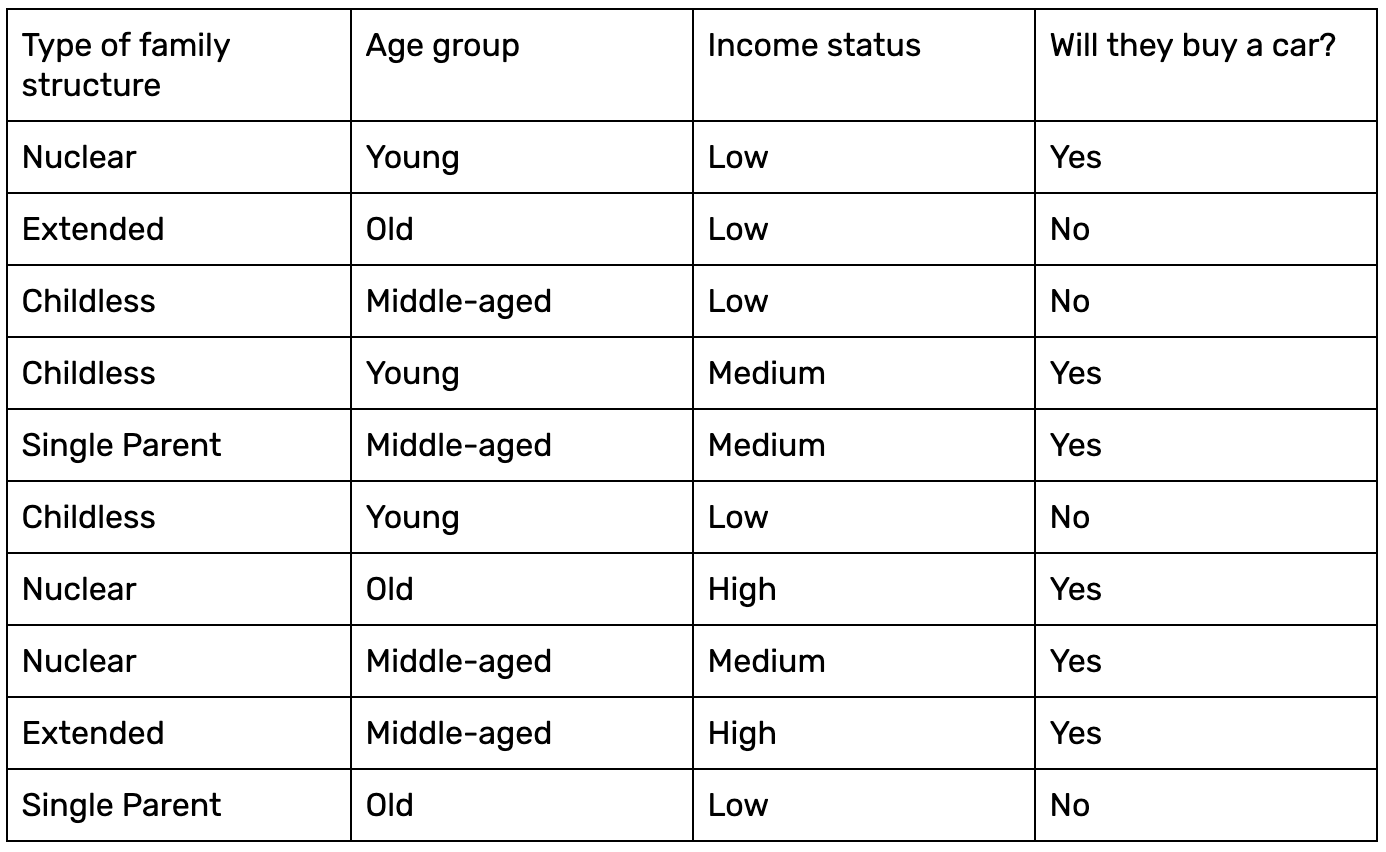
A clear-cut example should give you insight into how the above equation is put into practice. Let’s consider a simple dataset comprised of 10 data samples:

Fig 3: Image from Paperspace

Given three inputs–for example, Single Parent, Young, and Low–we want to compute the probability of these people buying a car. Let’s use Naive Bayes.

Firstly, let’s compute the probability of the output labels (P(Y)) given the data.

P(No) = 4/10

P(Yes) = 6/10

Now let’s calculate the probability of the likelihood of the evidence. Given the inputs Childless, Young, and Low, we'll calculate the probability with respect to both class labels as follows:

P(Single Parent|Yes) = 1/6

P(Single Parent|No) = 1/4

P(Young|Yes) = 2/6

P(Young|No) = 1/4

P(Low|Yes) = 1/6

P(Low|No) = 4/4

Since P(X1) \* P(X2) \* ... \* P(Xn) remains the same when calculating the probability for both Yes and No output labels, we can eliminate that value.

Thus, the posterior probability is computed as follows (note that X is the test data):

P(Yes|X) = P(Single Parent|Yes) \* P(Young|Yes) \* P(Low|Yes) = 1/6 \* 2/6 \* 1/6 = 0.0063

P(No|X) = P(Single Parent|No) \* P(Young|No) \* P(Low|No) = 1/4 \* 1/4 \* 4/4 = 0.0625

The final probabilities are:

P(Yes|X) = 0.0063/(0.0063 + 0.0625) = 0.09

P(No|X) = 0.0625/(0.0063 + 0.0625) = 0.91

Thus, the results clearly show that the car probably will not be purchased.

We previously mentioned that the "naiveness" of the algorithm is that it assumes each feature is independent of the others. We calculated the probabilities with respect to the output label with this assumption, so that each feature has an equal contribution and is independent of all the other features. This indeed is the naive assumption.

APPLICATIONS:

Below are a few use cases that employ Naive Bayes:

* **Real-time prediction:** Naive Bayes is an eager learning classifier and is quite fast in its execution. Thus, it could be used for making predictions in real-time.
* **Multi-class prediction:** The Naive Bayes algorithm is also well-known for multi-class prediction, or classifying instances into one of several different classes.
* **Text classification/spam filtering/sentiment analysis:** When used to classify text, a Naive Bayes classifier often achieves a higher success rate than other algorithms due to its ability to perform well on multi-class problems while assuming independence. As a result, it is widely used in spam filtering (identifying spam email) and sentiment analysis (e.g. in social media, to identify positive and negative customer sentiments).
* **Recommendation Systems:** A Naive Bayes Classifier can be used together with Collaborative Filtering to build a Recommendation System which could filter through new information and predict whether a user would like a given resource or not.

IMPLEMENTATION OF NAÏVE BAYES:

Code:

|  |
| --- |
| import pandas as pd  df=pd.read\_csv("/content/drive/MyDrive/csv/Iris - Iris.csv")  df  print(df.head)  print(df.tail)  x=df.loc[:,["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm"]]  x  y=df.loc[:,"Species"]  y  from sklearn.model\_selection import train\_test\_split  x\_train,x\_test,y\_train,y\_test= train\_test\_split(x,y,test\_size=0.30,random\_state=0)    from sklearn.naive\_bayes import GaussianNB  gnb = GaussianNB() #for Gaussain  gnb.fit(x\_train, y\_train) #for Gaussain  y\_predict=gnb.predict(x\_test) #for Gaussain  x\_train,y\_train  print(len(x\_train))  print(len(y\_train))  x\_test,y\_test  Output:    from sklearn.metrics import accuracy\_score  #for accuracy  acc=accuracy\_score(y\_test,y\_predict)  print(acc) |

EXPLANATION:

We have used scikit learn library to build naïve bayes model.

Firstly for visualization we have imported pandas library then we have obtained the dataset in csv file and then distributed the features in x variable and label in y variable. After this we have imported train\_test\_split model and splited the whole dataset into training and tested portion with 70/30. 70% training data and 30% testing data. Now we have imported naïve bayes model to apply it onto the dataset and then fitted the model in it to obtain accuracy.

**We have accuracy of 97% for this data set which is quite a satisfactory result** :)

ADVANTAGES OF NAÏVE BAYES :

* Naive Bayes is easy to grasp and works quickly to predict class labels. It also performs well on multi-class prediction.
* When the assumption of independence holds, a Naive Bayes classifier performs better compared to other models like logistic regression, and you would also need less training data.
* It performs well when the input values are categorical rather than numeric. In the case of numerics, a normal distribution is assumed to compute the probabilities (a bell curve, which is a strong assumption).

DISADVANTAGES OF NAÏVE BAYES :

* If a categorical variable has a category in the test data set, which is not observed by the model in the training data, it will assign a 0 (zero) probability and not be able to make a prediction. This is often referred to as “Zero Frequency”. To solve this problem, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
* The other limitation of Naive Bayes is the assumption of independence among the features. In real life, it is almost impossible that we get a set of features that are completely independent.

SUMMARY:

Though Naive Bayes has a handful of limitations, it’s still a go-to algorithm to classify data, primarily due to its simplicity. It has worked particularly well for document classification and spam filtering. For a more hands-on understanding of Naive Bayes, I recommend you try using what we implemented on various other datasets to gain a deeper insight into how Naive Bayes analyzes and classifies the data.