Naïve Bayes Classiﬁer Algorithm in R

▷ Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classiﬁcation problems.

▷ It is mainly used in text classiﬁcation that includes a high-dimensional training dataset.

▷ Naïve Bayes Classiﬁer is one of the simple and most effective Classiﬁcation algorithms which helps in building the fast machine learning models that can make quick predictions.

▷ It is a probabilistic classiﬁer, which means it predicts on the basis of the probability of an object.

▷ Some popular examples of Naïve Bayes Algorithm are spam ﬁltration, Sentimental analysis, and classifying articles.

### Why is it called Naïve Bayes?

▷ The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

* Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identiﬁed on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
* Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

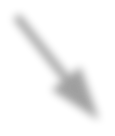
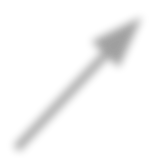
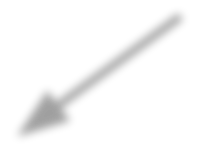
### Bayes' Theorem:

▷ Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

▷ The formula for Bayes' theorem is given as:

Likelihood Class Prior Probability

Posterior Probability



Predictor prior probability

### Working of Naïve Bayes' Classiﬁer:

▷ Working of Naïve Bayes' Classiﬁer can be understood with the help of the below example:

▷ Suppose we have a dataset of weather conditions and corresponding target variable "Play".

▷ So, using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions.

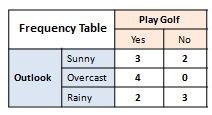
▷ So, to solve this problem, we need to follow the below steps:

* Construct a frequency table for each attribute against the target.
* Transform the frequency tables to likelihood tables
* Finally use the Naive Bayesian equation to calculate the posterior probability for each class.
* The class with the highest posterior probability is the outcome of prediction.



|  |  |
| --- | --- |
| ▷ | *Example 1*: |
| ▷ | We use the same simple Weather dataset here. |
| ▷ | There are **total 14 tuples** out of which- |
| ▷ | **9 tuples** are of **“yes”** class |
| ▷ | **5 tuples** are of **“no”** class |
| ▷ | **Outlook, Temp, Humidity, Windy are attributes**. |
| ▷ | **Play Golf - target or class attribute** |
| ▷ | **Outlook attribute has values – Rainy, Overcast, Sunny** |
| ▷ | **Temp attribute has values – Hot, Mild, Cool** |
| ▷ | **Humidity attribute has values – High, Normal** |
| ▷ | **Windy attribute has values – False, True** |

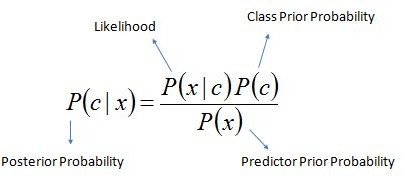
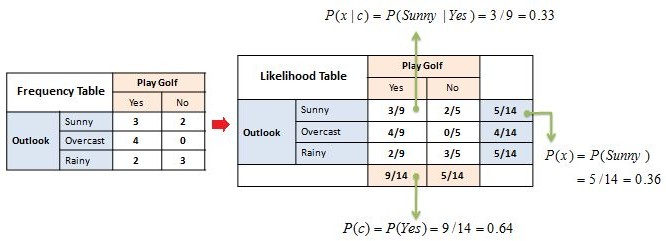
▷ Construct a frequency table for each attribute against the target.



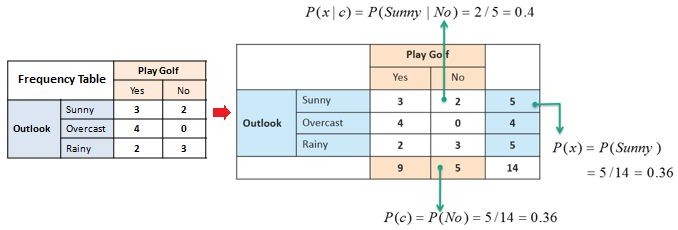
▷ Transform the frequency tables to likelihood tables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Likelihood Table** | | **Play Golf** | | **Total** |
| **Yes** | **No** |
| **Outlook** | **Sunny** | **3/9** | **2/5** | **5/14** |
| **Overcast** | **4/9** | **0/5** | **4/14** |
| **Rainy** | **2/9** | **3/5** | **5/14** |
|  | | **9/14** | **5/14** |  |

* Finally use the Naive Bayesian equation to calculate the posterior probability for each class.
* **probability that tuple X belongs to class C = yes**, given that we know the **attribute Outlook=“Sunny”**



###### probability that tuple X belongs to class C = no, given that we know the attribute Outlook=“Sunny”





▷ Problem: If the weather is sunny, then the Player should play or not?

▷ Solution: The class with the highest posterior probability is the outcome of prediction.

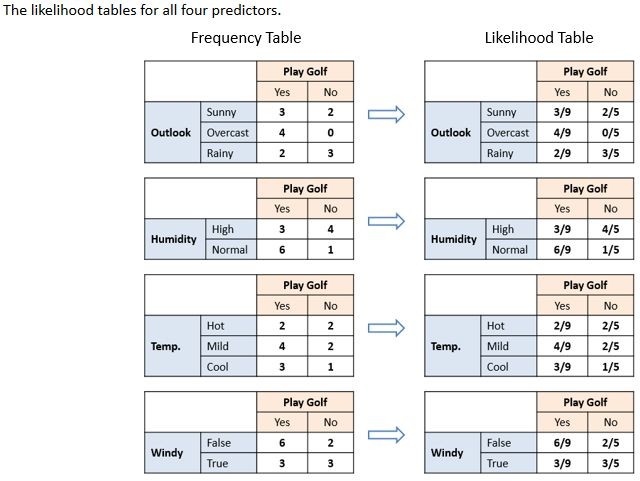
▷ We got,





So, as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.



###### Exercise

* Calculate Posterior probability that tuple X belongs to class C = yes, given that we know the attribute Humidity=“High”
* Calculate Posterior probability that tuple X belongs to class C = No, given that we know the attribute Humidity=“High”
* Calculate Posterior probability that tuple X belongs to class C = yes, given that we know the attribute Humidity=“Normal”
* Calculate Posterior probability that tuple X belongs to class C = No, given that we know the attribute Humidity=“Normal”
* Calculate Posterior probability that tuple X belongs to class C = yes, given that we know the attribute Temp=“Hot”
* Calculate Posterior probability that tuple X belongs to class C = No, given that we know the attribute Temp=“Hot”
* Calculate Posterior probability that tuple X belongs to class C = yes, given that we know the attribute Temp=“Mild”
* Calculate Posterior probability that tuple X belongs to class C = No, given that we know the attribute Temp=“Mild”
* Calculate Posterior probability that tuple X belongs to class C = yes, given that we know the attribute Temp=“Cool”
* Calculate Posterior probability that tuple X belongs to class C = No, given that we know the attribute Temp=“Cool”
* Calculate Posterior probability that tuple X belongs to class C = yes, given that we know the attribute Windy=“True”
* Calculate Posterior probability that tuple X belongs to class C = No, given that we know the attribute Windy=“False”

▷ 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):

▷



Where P(c|x) is the posterior probability of class (target) given predictor (attribute)

P(Yes|X) = P(Rainy|Yes) × P(Cool|Yes) × P(High|Yes) × P(True|Yes) × P(yes)

= (2/9) × (3/9) × (3/9) × (3/9) × (9/14) =0.005291

P(No|X) = P(Rainy|No) × P(Cool|No) × P(High|No) × P(True|No) × P(No)

= (3/5) × (1/5) × (4/5) × (3/5) × (5/14) =0.02057

### ▷ So we have,

* + P(Yes|X) = 0.005291
  + P(No|X) = 0.02057

## ▷ Since, P(No | X) > P(Yes | X)

▷ **So, prediction that golf would be played is ‘No’**

## Implementation of Naïve Bayes Classiﬁer in R

▷ **Step 1 – Install and Load required packages – caTools, e1071, caret**

library(caTools) library(e1071) library(caret)

## ▷ Step 2 - Load dataset iris

Iris

▷ **Step 3 - Gather information about dataset and if necessary, perform data preprocessing**

##### dim(iris) table(iris$Species)

▷ Table function in R - table(), performs categorical tabulation of data with the variable and its frequency.

▷ Table() function is also helpful in creating Frequency tables with condition and cross tabulations.

#### Syntax:

##### table(…, exclude = if (useNA == "no") c(NA, NaN), useNA = c("no", "ifany", "always"), dnn = list.names(…), deparse.level = 1)

###### Arguments

* + … - one or more objects which can be interpreted as factors (including character strings), or a list (or data frame) whose components can be so interpreted.
  + exclude - levels to remove for all factors in …. If it does not contain NA and useNA is not speciﬁed, it implies useNA = "ifany".
  + useNA - whether to include NA values in the table.
  + dnn - the names to be given to the dimensions in the result (the dimnames names).
  + deparse.level - controls how the default dnn is constructed.

## ▷ Step 4 - Split the dataset into the train and test parts.

set.seed(123)

split = sample.split(iris$Species, SplitRatio = 0.7)# returns true if observation goes to the Training set and false if observation goes to the test set.

#Creating the training set and test set separately training\_set = subset(iris, split == TRUE)

test\_set = subset(iris, split == FALSE) training\_set

test\_set

table(test\_set$Species)

###### ▷ Step 5 – Apply naïve bayes classiﬁer to the training data set using naiveBayes()

▷ **Syntax -**

naiveBayes(formula, data, laplace = 0, ..., subset, na.action = na.pass)

* Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

▷ Arguments

* formula - A formula of the form class ~ x1 + x2 + …. Interactions are not allowed.
* data - Either a data frame of predictors (categorical and/or numeric) or a contingency table.
* laplace - positive double controlling Laplace smoothing. The default (0) disables Laplace smoothing.
* subset - For data given in a data frame, an index vector specifying the cases to be used in the training sample.
* na.action - A function to specify the action to be taken if NAs are found. The default action is not to count them for the computation of the probability factors. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable.

iris\_classiﬁer=naiveBayes(Species ~ ., data = training\_set)

▷ **Step 5 - Display the Model**

iris\_classiﬁer

#### ▷ Step 6 - Predict test data

iris\_test\_pred=predict(iris\_classiﬁer,test\_set) iris\_test\_pred

#### ▷ Step 7 – Compare actual test data and predicted results

▷ Table function also helpful in creating 2 way cross table in R table(test\_set$Species)

table(iris\_test\_pred)

table(iris\_test\_pred, test\_set$Species,dnn=c("Prediction","Actual"))

#### ▷ Step 8- Check the accuracy of prediction with confusion matrix function.

##### cm = confusionMatrix(test\_set$Species, iris\_test\_pred) print(cm)

▷ If test data set has zero frequency issue, apply smoothing techniques “Laplace Correction” to predict the class of test data set.

iris\_classiﬁer\_lap=naiveBayes(Species ~ ., data = training\_set,laplace=1) iris\_classiﬁer\_lap

iris\_test\_pred\_lap=predict(iris\_classiﬁer\_lap,test\_set) table(iris\_test\_pred\_lap)

cmlap = confusionMatrix(test\_set$Species, iris\_test\_pred\_lap) print(cmlap)

## Implementing Naïve Bayes Classiﬁer to the Titanic dataset

▷ /\*Program 2\*/

###### ▷ Display dataset Titanic

Titanic

###### ▷ Display datatype of Titanic dataset

class(Titanic)

###### ▷ Display ﬁrst few rows of dataset

head(Titanic)

###### ▷ Describe dataset

str(Titanic)

▷ **Convert Titanic dataset into dataframe**

dfdata<-as.data.frame(Titanic)

#### ▷ Display datatype of converted dataset

class(dfdata)

#### ▷ Gather information about dataset and perform preprocessing if necessary.

names(dfdata) dim(dfdata) dfdata

## ▷ Split dataset into training and test sets

* set.seed(123)
* t\_split = sample.split(dfdata$Survived, SplitRatio = 0.8)
* #Creating the training set and test set separately
* training\_set1 = subset(dfdata, t\_split == TRUE)
* test\_set1 = subset(dfdata, t\_split == FALSE)
* training\_set1
* test\_set1
* table(test\_set1$Survived)

▷ **Apply naïve bayes classiﬁer to the training dataset** titanic\_classiﬁer=naiveBayes(Survived ~ ., data = training\_set1) titanic\_classiﬁer

## ▷ Predict test data

titanic\_test\_pred=predict(titanic\_classiﬁer,test\_set1) titanic\_test\_pred

▷ **Compare actual test data and predicted results**

table(titanic\_test\_pred)

table(titanic\_test\_pred, test\_set1$Survived,dnn=c("Prediction","Actual"))

▷ **Check the accuracy of prediction with confusion matrix function.** cm\_titanic = confusionMatrix(test\_set1$Survived, titanic\_test\_pred) print(cm\_titanic)

# Naïve Bayes Classiﬁer Algorithm

**Any Doubts?**

# Hierarchical clustering

**Agglomerative**

### #hierarchical Clustering #agglomerative install.packages("dplyr") library(dplyr)

df<-USArrests df

### #pre-processing #remove na values df<-na.omit(df)

#scale (normalizing or Standardizing) d<-scale(df)

### head(d)

d<-dist(d,method = "euclidean") hc<-hclust(d,method = "complete") plot(hc)

### plot(hc,cex=0.1,hang = -1)