Tutorial: Naïve Bayes (important)

Q1: Our very simple naïve Bayes problem. In this case, we have only one input variable/attribute. Normally we have more. We have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. The question is are players will play if weather is sunny?

Solution:

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)
Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64
Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.
Let's see the opposite:
P(No | Sunny) = P(Sunny |No) * P(No) / P (Sunny)
Here we have P (Sunny |No) = 2/5 = 0.4, P(Sunny) = 5/14 = 0.36, P(No)= 5/14 = 0.36
Now, P (No | Sunny) = 0.4 * 0.36 / 0.36 = 0.40, which has lower probability.
Thus, the answer is yes, play.

Q2: Given the training data in the table below (Buy Computer data) predict the class (yes or no) of the following example using Naïve Bayes classification: age<=30, income=medium, student=yes, credit-rating=fair (this is our target)

Solution:

RID	age	income	student	credit_rating	Class: buys_computer
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31 . . . 40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31 . . . 40	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	31 . . . 40	medium	no	excellent	yes
13	31 . . . 40	high	yes	fair	yes
14	>40	medium	no	excellent	no

E= age<=30, income=medium, student=yes, credit-rating=fair
E_1 is age<=30, E2 is income=medium, student=yes, E4 is credit-rating=fair
We need to compute P(yes|E) and P(no|E) and compare them.

$$P(yes \mid E) = \frac{P(E_1 \mid yes)\,P(E_2 \mid yes)\,P(E_3 \mid yes)\,P(E_4 \mid yes)\,P(yes)}{P(E)}$$

P(yes)=9/14=0.643 P(no)=5/14=0.357

P(E1|yes)=2/9=0.222 P(E1|no)=3/5=0.6
P(E2|yes)=4/9=0.444 P(E2|no)=2/5=0.4
P(E3|yes)=6/9=0.667 P(E3|no)=1/5=0.2
P(E4|yes)=6/9=0.667 P(E4|no)=2/5=0.4

$$P(yes \mid E) = \frac{0.222\ 0.444\ 0.667\ 0.668\ 0.443}{P(E)} = \frac{0.028}{P(E)} \qquad P(no \mid E) = \frac{0.6\ 0.4\ 0.2\ 0.4\ 0.357}{P(E)} = \frac{0.007}{P(E)}$$

Hence, the Naïve Bayes classifier predicts buys_computer=yes for the new example.

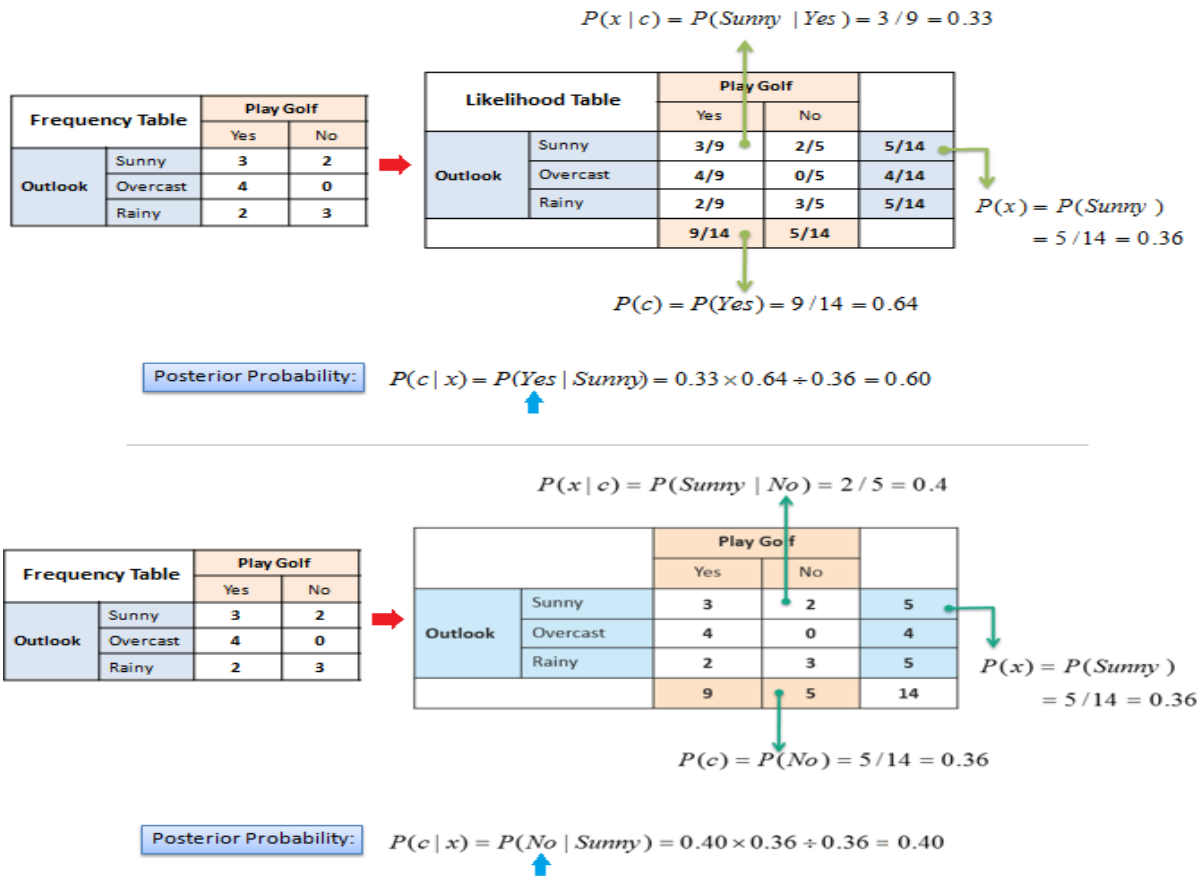
Check: Here the denominator P(E) is not calculated, as it is the same in both cases, so does not have a practical effect in the result. How much is P(E) however? Consult the slides to find it out!

Q3: This weather dataset has 14 instances and five numbers of attributes. Here, first four attributes are predictors and the last attribute is the target attribute (if we have to play golf).

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

The posterior probability can be calculated by first, constructing a frequency table for each attribute against the target. Then, transforming the frequency tables to likelihood tables and 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.

See the following schematic, as an example:



Now, back to our problem. The likelihood tables for all four predictors are:

Frequency Table

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3

Likelihood Table

		Play Golf	
		Yes	No
Outlook	Sunny	3/9	2/5
	Overcast	4/9	0/5
	Rainy	2/9	3/5

		Play Golf	
		Yes	No
Humidity	High	3/9	4/5
	Normal	6/9	1/5

		Play Golf	
		Yes	No
Temp.	Hot	2/9	2/5
	Mild	4/9	2/5
	Cool	3/9	1/5

		Play Golf	
		Yes	No
Windy	False	6/9	2/5
	True	3/9	3/5

In this example we have 4 inputs (predictors). The final posterior probabilities can be standardized between 0 and 1. **Our Target:** We need to classify the following new instance:
Outlook: Rainy, Temp: Cool, Humidity: High, Windy: True, Play:?

P(Yes|X) = P(Rainy|Yes) x P(Cool|Yes) x P(High|Yes) x P(True|Yes) x P(Yes)
P(Yes|X) = 2/9 x 3/9 x 3/9 x 3/9 x 9/14 = 0.00529
or 0.00529/ (0.00529+ 0.02057) = 0.20 (this is an option)

P(No|X) = P(Rainy|No) x P(Cool|No) x P(High|No) x P(True|No) x P(No)
P(No|X) = 3/5 x 1/5 x 4/5 x 3/5 x 5/14 = 0.02057
or 0.02057/ (0.00529+ 0.02057) = 0.80

So the probability for no is highest as compare to yes, so it is more likely to watch a movie instead of playing golf!

Q4: (Homework) This weather dataset has 14 instances and five numbers of attributes. It is slightly different from the table in Q3 (see the outlook attribute). **Our Target:** We need to classify the following new instance:
Outlook: Sunny, Temp: Mild, Humidity: Normal, Windy: False, Play:?

Table 1: Dataset

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

R-Implementations

Our first simple attempt in Naïve Bayes classification, using Iris Dataset

The Iris dataset is pre-installed in R and it includes 150 data points and 5 variables. Each data point concerns a particular iris flower and gives 4 measurements of the flower: Sepal.Length, Sepal.Width, Petal.Length and Petal.Width together with the flower's Species. The goal is to build a classifier that predicts species from the 4 measurements, so species is the class variable.

```
library(e1071)
library(caret)
data(iris)

str(iris)
'data.frame':           150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500

classifier <- naiveBayes(iris[,1:4],iris[,5])
table(predict(classifier, iris[,-5]), iris[,5], dnn=list('predicted','actual'))

            actual
predicted    setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         47         3
  virginica       0          3        47
```

As you should see the classifier does a pretty good job of classifying. Why is this not surprising? Because, we used all samples in training! (Not correct for a classification case study!)
To see what's going on 'behind-the-scenes', first do:

```
classifier$apriori
```

This gives the class distribution in the data: the prior distribution of the classes ("A priori" is the Latin for "from before")

```
Y
    setosa versicolor  virginica
        50         50         50
```

Since the predictor variables here are all continuous, the Naïve Bayes classifier generates three Gaussian (Normal) distributions for each predictor variable: one for each value of the class variable Species. If we type:

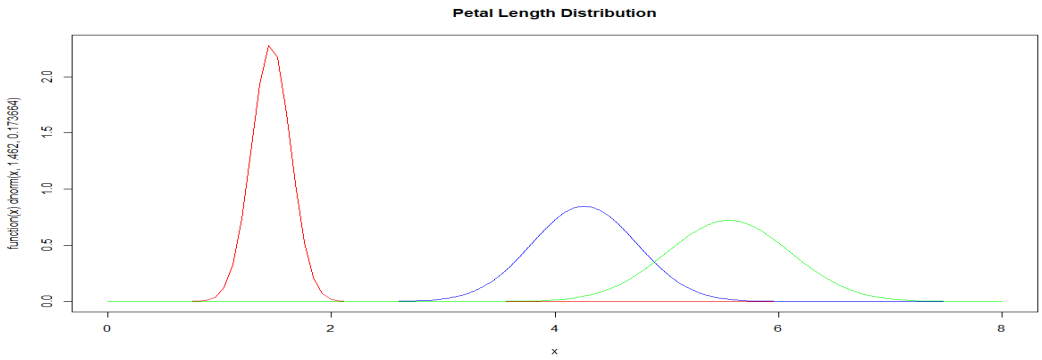
```
classifier$tables
```

We will see the mean (first column) and standard deviation (second column) for the 3 class-dependent Gaussian distributions for all input variables.

```
$Sepal.Length
           Sepal.Length
Y             [,1]      [,2]
  setosa     5.006 0.3524897
  versicolor 5.936 0.5161711
  virginica  6.588 0.6358796

$Sepal.Width
           Sepal.Width
Y             [,1]      [,2]
  setosa     3.428 0.3790644
  versicolor 2.770 0.3137983
  virginica  2.974 0.3224966

$Petal.Length
           Petal.Length
Y             [,1]      [,2]
  setosa     1.462 0.1736640
  versicolor 4.260 0.4699110
  virginica  5.552 0.5518947

$Petal.Width
           Petal.Width
Y             [,1]      [,2]
  setosa     0.246 0.1053856
  versicolor 1.326 0.1977527
  virginica  2.026 0.2746501
```

We can plot these 3 distributions against each other with the following three R commands, see the case for petal length:

```
plot(function(x) dnorm(x, 1.462, 0.1736640), 0, 8, col="red", main = "Petal Length Distribution")
curve(dnorm(x, 4.260, 0.4699110), add=TRUE, col="blue")
curve(dnorm(x, 5.552, 0.5518947), add=TRUE, col="green")
```



Note that setosa iris samples (the red curve) tend to have smaller petals (mean value = 1.462) and there is less variation in petal length (standard deviation is only 0.1736640).

But, let's try to do a "proper" classification, by creating separate training and testing subsets.

```
#Split iris data into train and test
train.indices <- sample(1:nrow(iris), 100)     # 100 train and 50 test samples
iris.train <- iris[train.indices, ]
iris.test <- iris[-train.indices, ]

classifier1 <- naiveBayes(iris.train[,1:4],iris.train[,5])
#alternatively, we could have
#classifier1 <- naiveBayes(Species~., data=iris.train)  # check the differences inside the bracket
classifier1
```

```
Naive Bayes Classifier for Discrete Predictors
Call:
naiveBayes.default(x = iris.train[, 1:4], y = iris.train[, 5])

A-priori probabilities:
iris.train[, 5]
    setosa versicolor  virginica
      0.30       0.35       0.35

Conditional probabilities:
             Sepal.Length
iris.train[, 5]     [,1]     [,2]
    setosa     5.030000 0.3869866
    versicolor 5.997143 0.5215845
    virginica  6.560000 0.6348691

             Sepal.Width
iris.train[, 5]     [,1]     [,2]
    setosa     3.473333 0.3600128
    versicolor 2.820000 0.2958537
    virginica  2.922857 0.3163075

             Petal.Length
iris.train[, 5]     [,1]     [,2]
    setosa     1.426667 0.1818171
    versicolor 4.282857 0.4448964
    virginica  5.560000 0.5796551

             Petal.Width
iris.train[, 5]      [,1]      [,2]
    setosa     0.2466667 0.1041661
```

```
    versicolor 1.3457143 0.1852548
    virginica  2.0000000 0.2940588
```

```
table(predict(classifier1, iris.test[,-5]), iris.test[,5], dnn=list('predicted','actual'))
```

```
            actual
predicted    setosa versicolor virginica
  setosa         20          0         0
  versicolor      0         15         1
  virginica       0          0        14
```

```
accuracy_iris <-predict(classifier1,iris.test[,-5])
testTable <- table(accuracy_iris, iris.test[,5])
testAcc<- sum(diag(testTable))/sum(testTable)
testAcc
```

```
 [1] 0.98
```

```
See also the following equivalent ways of showing the performance
nb_mod.C <- naiveBayes(Species~., data=iris.train)
nb.test.C <- predict(nb_mod.C, iris.test[,1:4], type="class")
nb.test.acc <- confusionMatrix(nb.test.C, iris.test$Species, mode="prec_recall")
nb.test.acc
nb.test.acc1 <- confusionMatrix(nb.test.C, iris.test$Species, mode="everything")
nb.test.acc1                    # see the difference from everything to prec_recall
```

```
Confusion Matrix and Statistics
            Reference
Prediction   setosa versicolor virginica
  setosa         20          0         0
  versicolor      0         15         1
  virginica       0          0        14

Overall Statistics
               Accuracy : 0.98
                 95% CI : (0.8935, 0.9995)
    No Information Rate : 0.4
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9697
Mcnemar's Test P-Value : NA

Statistics by Class:
                     Class: setosa Class: versicolor Class: virginica
Sensitivity                    1.0            1.0000           0.9333
Specificity                    1.0            0.9714           1.0000
Pos Pred Value                 1.0            0.9375           1.0000
Neg Pred Value                 1.0            1.0000           0.9722
Precision                      1.0            0.9375           1.0000
Recall                         1.0            1.0000           0.9333
F1                             1.0            0.9677           0.9655
Prevalence                     0.4            0.3000           0.3000
Detection Rate                 0.4            0.3000           0.2800
Detection Prevalence           0.4            0.3200           0.2800
Balanced Accuracy              1.0            0.9857           0.9667
```

Naïve Bayes Classification using Students Dataset

Suppose we wish to understand the choice a student makes between different types of high-school programmes (e.g. an academic, general or vocational course). To explore this, we have data available on

the academic choices made by 200 US students. This dataset is available from the UCLA Institute for digital research and education using the following link: https://stats.idre.ucla.edu/stat/data/hsbdemo.dta. In this tutorial, it is provided as an xlsx file. In addition to the actual programme choice made by the student, the dataset also contains information on other factors that could potentially influence their choices such as each student's socio-economic status, the type of school attended (public or private), gender and their prior reading, writing, maths and science scores. For tutorial purposes, we will make a Naive Bayes classifier here.

```
library(e1071)
library(caret)
library(readxl)   # this is the package for reading xlsx files

students<-read_excel("C:/R_codes/hsbdemo.xlsx")
head(students)
```

```
# A tibble: 6 x 13
     id female ses    schtyp prog    read write  math science socst honors  awards   cid
  <dbl> <chr>  <chr>  <chr>  <chr>  <dbl> <dbl> <dbl>   <dbl> <dbl> <chr>    <dbl> <dbl>
1    45 female low    public vocat~    34    35    41      29    26 not en~      0     1
2   108 male   middle public gener~    34    33    41      36    36 not en~      0     1
3    15 male   high   public vocat~    39    39    44      26    42 not en~      0     1
4    67 male   low    public vocat~    37    37    42      33    32 not en~      0     1
5   153 male   middle public vocat~    39    31    40      39    51 not en~      0     1
6    51 female high   public gener~    42    36    42      31    39 not en~      0     1
# the tibble is  the new style in dataframes
```

```
Let's see the structure of students' dataset
str(students)
```

```
tibble [200 x 13] (S3: tbl_df/tbl/data.frame)
 $ id     : num [1:200] 45 108 15 67 153 51 164 133 2 53 ...
 $ female : chr [1:200] "female" "male" "male" "male" ...
 $ ses    : chr [1:200] "low" "middle" "high" "low" ...
 $ schtyp : chr [1:200] "public" "public" "public" "public" ...
 $ prog   : chr [1:200] "vocation" "general" "vocation" "vocation" ...
 $ read   : num [1:200] 34 34 39 37 39 42 31 50 39 34 ...
 $ write  : num [1:200] 35 33 39 37 31 36 36 31 41 37 ...
 $ math   : num [1:200] 41 41 44 42 40 42 46 40 33 46 ...
 $ science: num [1:200] 29 36 26 33 39 31 39 34 42 39 ...
 $ socst  : num [1:200] 26 36 42 32 51 39 46 31 41 31 ...
 $ honors : chr [1:200] "not enrolled" "not enrolled" "not enrolled" "not enrolled" ...
 $ awards : num [1:200] 0 0 0 0 0 0 0 0 0 0 ...
 $ cid    : num [1:200] 1 1 1 1 1 1 1 1 1 1 ..
```

```
detailed view of the data set
students$prog <- as.factor(students$prog)          # need to be as factor type, as this is our target
str(students)                                      # see the difference from the above str command
summary(students)
```

```
       id              female              ses                schtyp
 Min.   :  1.00   Length:200         Length:200         Length:200
 1st Qu.: 50.75   Class :character   Class :character   Class :character
 Median :100.50   Mode  :character   Mode  :character   Mode  :character
 Mean   :100.50
 3rd Qu.:150.25
 Max.   :200.00
    prog                read            write           math            science
 Length:200         Min.   :28.00   Min.   :31.00   Min.   :33.00   Min.   :26.00
 Class :character   1st Qu.:44.00   1st Qu.:45.75   1st Qu.:45.00   1st Qu.:44.00
 Mode  :character   Median :50.00   Median :54.00   Median :52.00   Median :53.00
```

```
                  Mean   :52.23   Mean   :52.77   Mean   :52.65   Mean   :51.85
                  3rd Qu.:60.00   3rd Qu.:60.00   3rd Qu.:59.00   3rd Qu.:58.00
                  Max.   :76.00   Max.   :67.00   Max.   :75.00   Max.   :74.00
     socst              honors             awards             cid
 Min.   :26.00   Length:200         Min.   :0.00   Min.   : 1.00
 1st Qu.:46.00   Class :character   1st Qu.:0.00   1st Qu.: 5.00
 Median :52.00   Mode  :character   Median :1.00   Median :10.50
 Mean   :52.41                      Mean   :1.67   Mean   :10.43
 3rd Qu.:61.00                      3rd Qu.:2.00   3rd Qu.:15.00
 Max.   :71.00                      Max.   :7.00   Max.   :20.00
```

Now we will make a Naive Bayes classifier for our data. We will make a 70/30 partitioning for the training and testing set. We used the createDataPartition from caret package to make a balanced partitioning. This is very important in this context because we are going to calculate the prior probabilities from the count of the training data. So, it should follow the proportion of the parent dataset.

```
# creation of Training and Testing Set
trainIndex <- createDataPartition(students$prog, p=0.7)$Resample1
train <- students[trainIndex, ]
test<- students[-trainIndex, ]

Classifier
NBclassifier <- naiveBayes(prog~., data=train)
NBclassifier

Naive Bayes Classifier for Discrete Predictors
Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
 academic   general  vocation
0.5248227 0.2269504 0.2482270

Conditional probabilities:
          id
Y            [,1]     [,2]
  academic 110.31081 58.21522
  general   95.34375 46.13767
  vocation  93.34286 58.03191

          female
Y           female      male
  academic 0.5810811 0.4189189
  general  0.5937500 0.4062500
  vocation 0.5142857 0.4857143

          ses
Y            high       low    middle
  academic 0.3783784 0.1621622 0.4594595
  general  0.1875000 0.3750000 0.4375000
  vocation 0.1142857 0.2857143 0.6000000

          schtyp
Y           private     public
  academic 0.21621622 0.78378378
  general  0.12500000 0.87500000
  vocation 0.05714286 0.94285714
```

```
        read
Y            [,1]     [,2]
  academic 56.89189 9.886434
  general  48.90625 9.613027
  vocation 44.54286 8.168601

        write
Y            [,1]     [,2]
  academic 56.97297 7.331622
  general  50.71875 9.632721
  vocation 46.37143 9.650010

        math
Y            [,1]     [,2]
  academic 56.43243 8.724208
  general  49.28125 6.721340
  vocation 46.00000 8.443584

        science
Y            [,1]     [,2]
  academic 53.83784  8.722553
  general  51.00000 10.207524
  vocation 47.71429 10.923723

        socst
Y            [,1]     [,2]
  academic 57.39189  8.759836
  general  50.40625  9.345896
  vocation 43.22857 11.499580

        honors
Y          enrolled not enrolled
  academic 0.4054054    0.5945946
  general  0.1250000    0.8750000
  vocation 0.1142857    0.8857143

        awards
Y            [,1]     [,2]
  academic 2.391892 1.885907
  general  1.218750 1.475267
  vocation 0.800000 1.510746

        cid
Y             [,1]     [,2]
  academic 12.972973 5.125742
  general   8.750000 5.297595
  vocation  6.028571 4.823263
```

The naiveBayes function assumed Gaussian distributions for numeric variables. For numeric variable "science", the Y values are the means and standard deviations of the predictors within each class. For categorical variable "honors", the Y values are the conditional probability of the predictors within each class. Now let us make prediction for the training data and for the test data.

```
NBclassifier$apriori
Y
academic  general vocation
      74       32       35

NBclassifier$tables$science
        science
```

```
Y            [,1]     [,2]
 academic 54.01351  9.010638
 general  52.71875  9.805313
 vocation 47.97143 10.489450
```

```
trainPred <- predict(NBclassifier, newdata = train, type = "class")
trainTable <- table(train$prog, trainPred)
trainTable
```

```
          trainPred
           academic general vocation
  academic       52       8       14
  general        12       7       13
  vocation        7       6       22
```

```
trainAcc <- sum(diag(trainTable))/sum(trainTable)*100
trainAcc
```

```
[1] 57.44681
```

```
testPred <- predict(NBclassifier, newdata = test, type = "class")
testTable <- table(test$prog, testPred)
testTable
```

```
          testPred
           academic general vocation
  academic       21       2        8
  general         4       1        8
  vocation        1       1       13
```

```
testAcc<- sum(diag(testTable))/sum(testTable)*100
testAcc
```

```
[1] 59.32203
```