



Let us now see how to implement NB in R.



Naive Bayes Classifier in R

- Due to its popularity and wide applications, there are several packages to apply Naïve Bayes (i.e. e1071, klaR, naivebayes, bnclassify)
- In this presentation, we demonstrate how Naïve Bayes classifier can be implemented using e1071 package (available on CRAN).
- `naiveBayes()` function can be used to train a model. The function computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.



WWW.KENT.EDU


As you might have discovered, R provides many packages to accomplish the same task. Here we will demonstrate the NB model from the e1070 package. Note that the output includes the conditional probabilities that we discussed in the previous module. Please refer to the GitHub account for the data files for this module.

Example

- Let us consider again the credit card default example.
- The goal is to create a classification model that can predict credit card defaults based on the balance on the card as well as the card holder income.
- A Naïve Bayes classifier will assume that income and credit card balance are two independent variables.

Code for this module is available on our github account at <https://github.com/KSU-MSBA/64060.git>

We are interested in predicting credit card default (Y/N) based on two variables, Income and Credit Card Balance. Note that the independent variables are numeric, and while not ideally suited for NB, will suffice to demonstrate our model. It is assumed that the variables are independent, and follow a Normal distribution.


KENT STATE
UNIVERSITY

Example

```
library(caret)
library(ISLR)
library(e1071) #install first

summary(Default)
```

##	default	student	balance	income
##	No :9667	No :7056	Min. : 0.0	Min. : 772
##	Yes: 333	Yes:2944	1st Qu.: 481.7	1st Qu.:21340
##			Median : 823.6	Median :34553
##			Mean : 835.4	Mean :33517
##			3rd Qu.:1166.3	3rd Qu.:43808
##			Max. :2654.3	Max. :73554

WWW.KENT.EDU

Install any libraries that are needed.

Example II

```
#remove student status, which is the second variable
MyData<-Default[,-2]
set.seed(123)

#Divide data into test and train
Index_Train<-createDataPartition(MyData$default, p=0.8, list=FALSE)
Train <-MyData[Index_Train,]
Test  <-MyData[-Index_Train,]

# Build a naïve Bayes classifier
nb_model <-naiveBayes(default~balance+income,data = Train)

# Predict the default status of test dataset
Predicted_Test_labels <-predict(nb_model,Test)

library("gmodels")

# Show the confusion matrix of the classifier
CrossTable(x=Test$default,y=Predicted_Test_labels, prop.chisq = FALSE)
```

The code and output are in your Github account. Here, we first divide the data into training and test, with 80% for training, and the rest for test. The NB classifier is called by the naiveBayes command. We then use the model to predict for the Test Set, and finally output the confusion matrix for the test set. The results are shown in the next slide.

Example II

```
##      Cell Contents
##      -----
##      N / Row Total
##      N / Col Total
##      N / Table Total
##
## Total Observations in Table: 1999
##
## Test$default Predicted Test_labels Row Total
##      No      1928      5      1933
##      0.997      0.003      0.967
##      0.975      0.238
##      0.964      0.003
##
##      Yes      50      16      66
##      0.758      0.242      0.033
##      0.025      0.762
##      0.025      0.008
##
## Column Total      1978      21      1999
##      0.989      0.011
##
##
```

False Positive

False Negative


A total of 55 Misclassification Cases

We have a total of 55 misclassification cases, with 5 cases as False Positive, and 50 as False Negative. Can you calculate Recall and Precision? Recall is the true positive rate, and Precision is the proportion of true positives as a proportion of all predicted positive values.

Probabilities

- Sometimes, it is preferred to have raw prediction probabilities rather than predicted class labels.
- `naiveBayes()` can return probabilities if the 'type' argument in the predict function is set to 'raw'
- The default value of 'type' argument is 'class' which returns the actual predicted class of the input.
- The probabilities can be used to compute the AUC of the model.

The model allows us to predict the raw probabilities, rather than the predicted class labels. To do that, we specify the “raw” argument. See the code for details. Once we have the probabilities, we can then calculate the AUC curve.



Returning Probabilities


```
nb_model <- naiveBayes(default~balance+income,data = Train)

#Make predictions and return probability of each class
Predicted_Test_labels <-predict(nb_model,Test, type = "raw")


#show the first few values
head(Predicted_Test_labels)
```

	No	Yes
## [1,]	0.9999207	7.933836e-05
## [2,]	0.9745238	2.547624e-02
## [3,]	0.9999951	4.907314e-06
## [4,]	0.9998550	1.449972e-04

→ Probability of default


[WWW.KENT.EDU](http://www.kent.edu)

Notice the argument type used in the predict function. This provides the probabilities.



AUC Value and ROC Curves

- We can use the package 'pROC' to compute the Area Under The Curve (AUC) of any classifier (not limited to Naïve Bayes)
- `roc()` is a function to compute the AUC value. The function takes the actual outcomes (i.e. ground truth labels) as the first argument and the predicted probabilities as the second argument and return the AUC.
- `plot.roc()`, takes the same inputs and can be used to plot the ROC curve.
- Note that the plot is shown using sensitivity and reversed ordered specificity which are the same as true positive and false positive rates as we saw before.



WWW.KENT.EDU

A [receiver operating characteristic curve](#), or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, or recall. The false-positive rate is also known as the fall-out or probability of false alarm and can be calculated as $(1 - \text{specificity})$. It can also be thought of as a plot of the power as a function of the Type I Error of the decision rule.

AUC Value and ROC Curves II

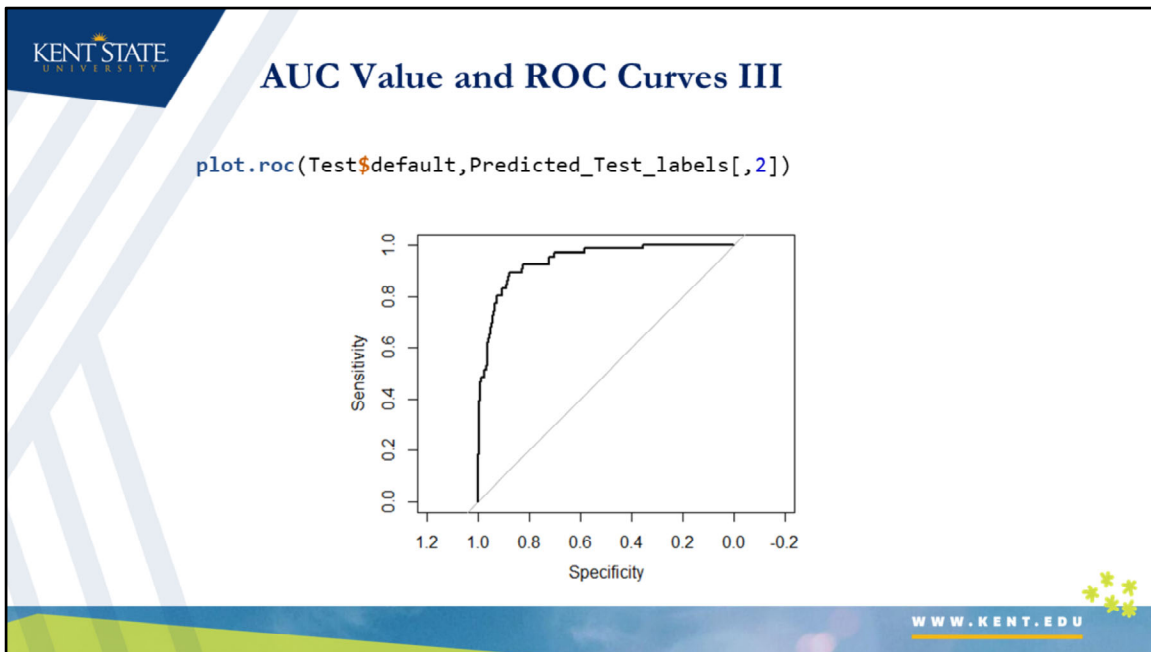
```
library(pROC)

#Passing the second column of the predicted probabilities
#That column contains the probability associate to 'yes'
roc(Test$default, Predicted_Test_labels[,2])

##
## Call:
## roc.default(response = Test$default, predictor = Predicted_Test_labels[,2])
##
## Data: Predicted_Test_labels[, 2] in 1933 controls (Test$default No) < 66
cases (Test$default Yes).
## Area under the curve: 0.9395
```

AUC Value

This shows the area under the curve whose value is 0.9395



This plot shows Sensitivity versus reversed ordered specificity, which is identical to (1-specificity) or false positive rate. This classifier has high AUC, as we saw from the previous output.

The plot shows the ROC (Receiver Operating Characteristic) curve. Starting from the lower left, the ROC curve plots the pairs {sensitivity, specificity} as the cut-off value descends from 1 to 0. Better performance is reflected by curves that are closer to the top-left corner. The comparison curve is the diagonal, which reflects the performance of the naive rule, using varying cutoff values (i.e., setting different thresholds on the level of majority used by the majority rule). A common metric to summarize an ROC curve is "area under the curve (AUC)," which ranges from 1 (perfect discrimination between classes) to 0.5 (no better than the naive rule). The ROC curve for our example was 0.9395.

This concludes our discussion for this Module. We next examine, some practical considerations in deploying NB models.