Classification using Naive Bayes

Predict flowering type depending plant genotype

Laura Sudupe Medinilla

12 de abril, 2021

Índice

Naive Bayes Algorithm

The *Naive Byes* algorithm describes a simple method to aplly Bayes theorem to classification problems. Althopugh it is not the only machine learning method that utilizes Bayesian methods, it is the most common one. This is particularly true for text classification, where it has become the de facto standard. The strengths and weaknesses of this algorithm are as follows:

Strenghts	**Weaknesses**
* Simple, fast, and very effective	* SRelies on an often-faulty assumption of equally important and independent features
* Does well with noisy and missing data	* Not ideal for datasets with many numeric features
* Requires relatively few examples for training, but also works well with very large numbers of examples	* Estimated probabilities are less reliable than the predicted classes
* Easy to obtain the estimated probability for a prediction	

Step 1 - Collecting the data

This algorithm objetive is to predict the flowering type (fast or slow) in function of the flower genotype.

We know the genotype of 697 flowers and the flowering day. We have 149 genotypes as predictor features. The genotype states are 0, 1 or 2, which belong to dominant homozygote, heterozygote and recessive homozigote.

Step 2 - Exploring and preparing the data

The first step towards constructing our classifier involves processing the raw data for analysis.

Using the `str()` function, we see our data structure

```
'data.frame':   697 obs. of  149 variables:
 $ gtype.1  : num  0 0 2 2 2 0 2 0 0 0 ...
 $ gtype.2  : num  2 1 2 0 2 0 0 0 2 2 ...
 $ gtype.3  : num  2 2 2 0 0 0 0 2 2 0 ...
 $ gtype.4  : num  2 2 0 0 0 0 2 0 2 2 ...
 $ gtype.5  : num  0 0 0 0 0 0 0 0 0 2 ...
  [list output truncated]
```

The element are numeric, it would be better to convert it into a factor

Check again the structure

```
'data.frame':   697 obs. of  149 variables:
 $ gtype.1  : Factor w/ 3 levels "0","1","2": 1 1 3 3 3 1 3 1 1 1 ...
 $ gtype.2  : Factor w/ 3 levels "0","1","2": 3 2 3 1 3 1 1 1 3 3 ...
 $ gtype.3  : Factor w/ 3 levels "0","1","2": 3 3 3 1 1 1 1 3 3 1 ...
 $ gtype.4  : Factor w/ 3 levels "0","1","2": 3 3 1 1 1 1 3 1 3 3 ...
 $ gtype.5  : Factor w/ 2 levels "0","2": 1 1 1 1 1 1 1 1 1 2 ...
  [list output truncated]
```

We need to create a new feature, *fast* or *slow* flowering. If the number of days is higher or same to 40 days, we codifice like 1, else, 0.

Take a look to the plants type numbers

```
flowering_factor
fast slow
 437  260
```

- Creating training and test datasets

We now need to split the data into training and test datasets.

```
set.seed(params$valor.seed)
train <- sample(nrow(genotype_f),floor(nrow(genotype_f)*params$p))
length(train)
```

```
[1] 464
```

```
train_data <- genotype_f[train,]
test_data <- genotype_f[-train,]

class_train <- flowering_factor[train]
class_test <- flowering_factor[-train]
```

Step 3 - Training a model on the data

The Naive Bayes implementation we will employ in the e1071 package. Build the classifier

```
library(e1071)
m <- naiveBayes(train_data, class_train, laplace=0)
```

The function will return a naive Bayes model object that can be used to make predictions

Step 4 - Evaluatin model performance

To evaluate the flowering classifier, we need to test its predictions on unseen test data. The `predict()` function is used to make the predictions

The function will return a vector of predicted class values ç

```
test_pred <- predict(m, test_data)
```

To compare the predictions to the true values, we´ll use the `CrossTable()` function

```
library(gmodels)
CrossTable(x =test_pred , y = class_test , prop.chisq=FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  233


             | class_test
   test_pred |      fast |      slow | Row Total |
-------------|-----------|-----------|-----------|
        fast |       116 |        43 |       159 |
             |     0.730 |     0.270 |     0.682 |
             |     0.817 |     0.473 |           |
             |     0.498 |     0.185 |           |
-------------|-----------|-----------|-----------|
        slow |        26 |        48 |        74 |
             |     0.351 |     0.649 |     0.318 |
             |     0.183 |     0.527 |           |
             |     0.112 |     0.206 |           |
-------------|-----------|-----------|-----------|
Column Total |       142 |        91 |       233 |
             |     0.609 |     0.391 |           |
-------------|-----------|-----------|-----------|
```

Step 5 - Improving model performance

We are going to do the same but we will set a value for the Laplace estimator

```
m2 <- naiveBayes(train_data, class_train, laplace=1)
```

Do the prediction

```
test_pred2 <- predict(m2, test_data)
```

Evaluate the model

```
#library(gmodels)
CrossTable(x =test_pred2 , y = class_test , prop.chisq=FALSE)
```

```
   Cell Contents
|-------------------------|
|                       N |
```

```
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  233


             | class_test
  test_pred2 |      fast |      slow | Row Total |
-------------|-----------|-----------|-----------|
        fast |       116 |        42 |       158 |
             |     0.734 |     0.266 |     0.678 |
             |     0.817 |     0.462 |           |
             |     0.498 |     0.180 |           |
-------------|-----------|-----------|-----------|
        slow |        26 |        49 |        75 |
             |     0.347 |     0.653 |     0.322 |
             |     0.183 |     0.538 |           |
             |     0.112 |     0.210 |           |
-------------|-----------|-----------|-----------|
Column Total |       142 |        91 |       233 |
             |     0.609 |     0.391 |           |
-------------|-----------|-----------|-----------|
```

The results are very similar between them

ROC curve

We will performe the ROC value for each case `laplace=0` y `laplace= 1`.

Case `laplace=0`

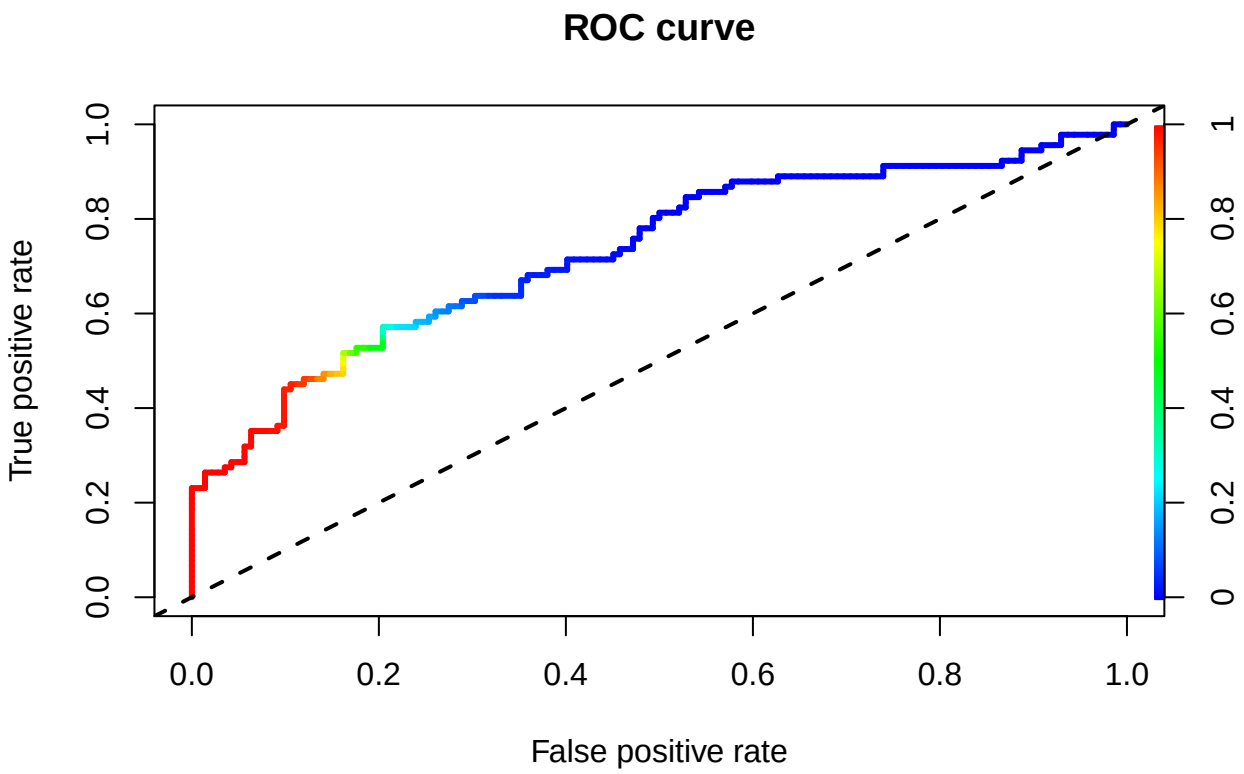
First of all we will obtain the flowering probabilities for each plant in the test data

```
test_pred_roc <- predict(m, test_data, type="raw")
```

With the positive class probabilities, we performe the ROC curve

```
pred_roc <- prediction(predictions= test_pred_roc[,2], labels=class_test)
perf_roc <- performance(pred_roc, measure="tpr", x.measure="fpr")
#unlist(perf_roc@alpha.values)


plot(perf_roc, main= "ROC curve", col= "blue", lwd=3, colorize=TRUE)
abline(a=0, b= 1, lwd= 2, lty = 2)
```




```
perf.auc <- performance(pred_roc, measure ="auc")
```

AUC value is **0.7303823**.

Case `laplace=1`

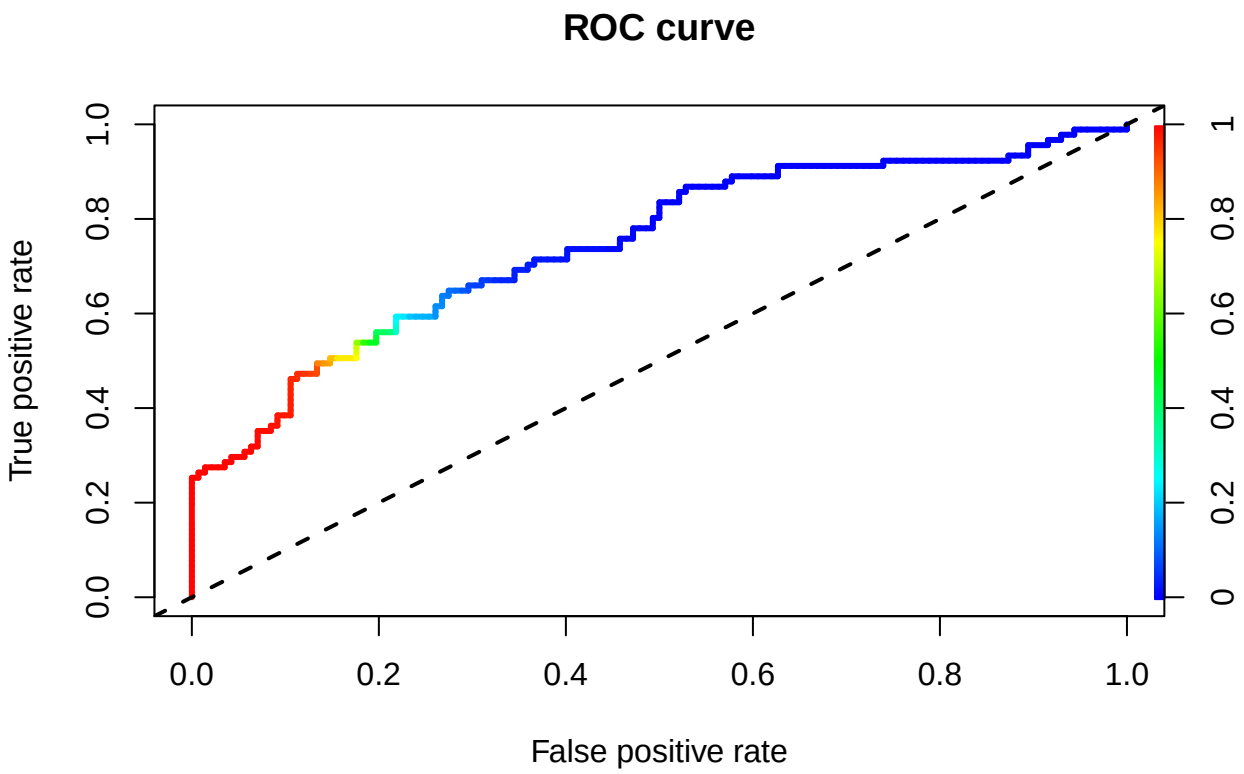
First of all we will obtain the flowering probabilities for each plant in the test data

```
test_pred2 <- predict(m2, test_data, type="raw")
```

With the positive class probabilities, we performe the ROC curve

```
pred2 <- prediction(predictions= test_pred2[,2], labels=class_test)
perf2 <- performance(pred2, measure="tpr", x.measure="fpr")
#unlist(perf@alpha.values)


plot(perf2, main= "ROC curve", col= "blue", lwd=3, colorize=TRUE)
abline(a=0, b= 1, lwd= 2, lty = 2)
```




```
perf2.auc <- performance(pred2, measure ="auc")


#str(perf)
```

AUC value is **0.7450085**.