Tutorial 3

Statistical models

École Nationale des Ponts et Chausées
Département Ingénieurie Mathématique et Informatique – Master II

Loïc BRIN • Benoit ROGER

Exercise 1: Term structure of default probability deduced from a Transition Matrix.

This exercise is based on the following S&P transition matrix:

	AAA	AA	A	BBB	BB	B	CCC	D
AAA	90,8%	8,3%	0,7%	0,1%	0,1%	0,0%	0,0%	0,0%
AA	0,1%	91,2%	7,9%	0,6%	0,1%	0,1%	0,0%	0,0%
A	0,9%	2,4%	90,0%	5,4%	0,7%	0,3%	0,1%	0,1%
BBB	0,0%	0,3%	5,9%	86,9%	5,3%	1,2%	0,1%	0,2%
BB	0,0%	0,1%	0,7%	7,7%	80,5%	8,8%	1,0%	1,2%
B	0,0%	0,1%	0,2%	0,5%	6,5%	82,7%	4,1%	5,9%
CCC	0,2%	0,0%	0,2%	1,3%	2,3%	12,9%	60,6%	22,5%
D	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	100,0%

available here http://defaultrisk.free.fr/data/TD2_1.csv.

1. Load the database and define a function that for a given number of year (n) and a given credit rating (l), returns a n-vector with the PD for each n years, deduced from the S&P transition matrix.

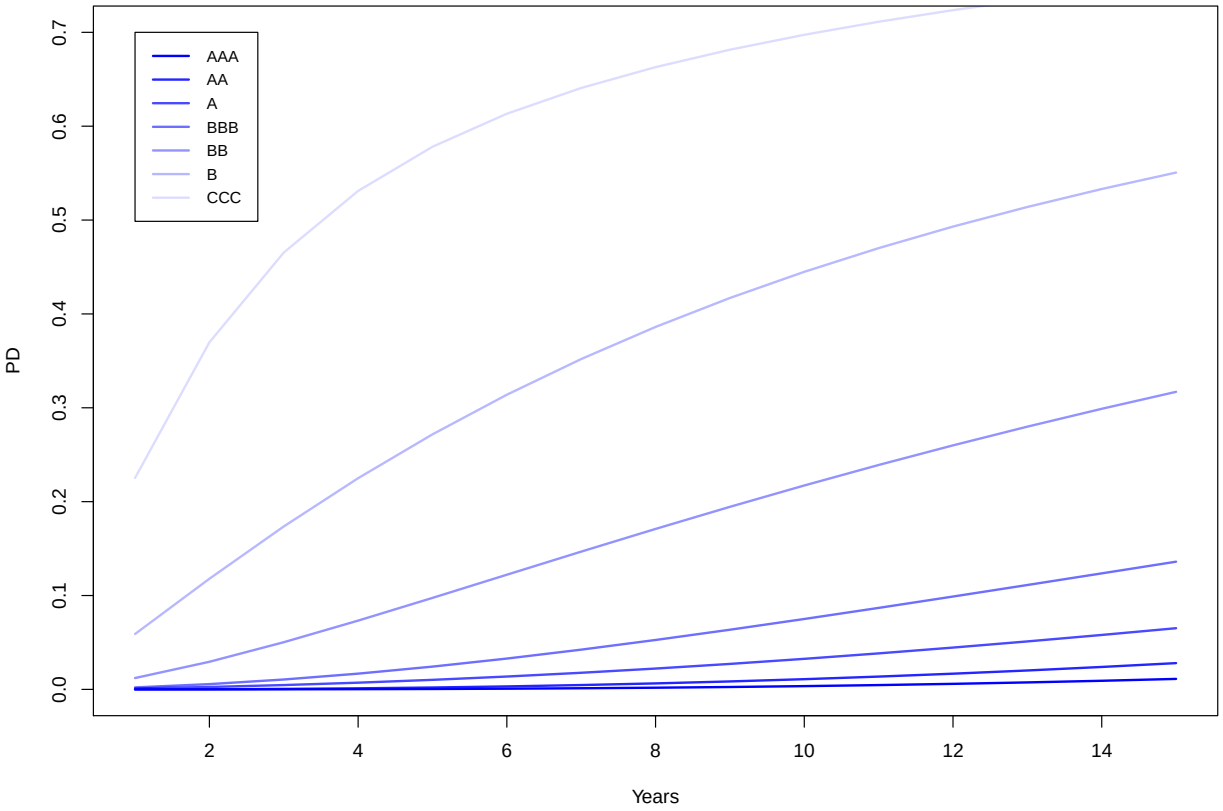
```r
### loading the transition matrix
data<-as.matrix(read.csv2("http://defaultrisk.free.fr/data/TD2_1.csv", header=F))

### recursive function to compute the matricial power of a matrix
powA <- function(A, n)
{
  if (n==1)  return (A)
  if (n==2)  return (A%*%A)
  if (n>2) return (A%*%powA(A,n-1))
}

### function that extracts the PD for n years of rating l
PDt<-function(n,l)
{
  res<-c()
  for (i in 1:n)
  {
    res<-c(res,powA(data,i)[l,8])
  }
  return(res)
}
```

`PDt` is a function that for a given number of years n, and rating l, returns the PD for each n years and stores them in a vector.

2. Plot the PD for all the ratings, for the next 15 years, deduced from the transition matrix.

```r
### ploting the term structure of default for each rating
library(RColorBrewer)
```

```
col<-colorRampPalette(c(rgb(0,0,1,1), rgb(0,0,1,0)), alpha = TRUE)(8)
plot(PDt(15,1), type="l", ylim=c(0,0.7), xlab="Years", ylab="PD", lwd=2, col=col[1])
for (i in 2:7)
{
  lines(PDt(15,i), lwd=2, col=col[i])
}
legend(1, 0.7, legend=c("AAA", "AA", "A", "BBB", "BB", "B", "CCC"),
       col=col, lty=1, lwd = 2, cex=0.8)
```



3. What do you observe?

We observe that in all the cases, the term structure of the probability of default is increasing. Moreover, we can see that investment grade ratings tend to have a convex term structure of probability of default whereas high yield ratings have a concave one.

Exercise 2: Load and explore the "German Credit" database.

1. Load the "German Credit" database available here http://defaultrisk.free.fr/data/TD2_2.csv. Name it `Credit` and display the first ten rows.

```
### Load the German Credit database
Credit <- read.csv("http://defaultrisk.free.fr/data/TD2_2.csv")

### Display the first ten rows
```

```r
head(Credit)

### You can also display the whole table in a nice way (Rstudio)
View(Credit)
```

2. Display the size of the dataframe `Credit`.

```r
### Get the dimension of the dataset Credit
dim(Credit)
```

The dataset `Credit` consists in 1000 observations and 18 variables (17 predictors and 1 target variable).

3. Which columns are numerical? Which are categorical?

```r
### Collect the class of each column in Credit
col_class <- as.character(lapply(Credit, "class"))

# or even simpler and safer with the library "purrr"
library(purrr)
col_class<-map_chr(Credit,class)

### Count the columns from each class and display the result
table(col_class)

### Get the names of the columns from each class and display them
writeLines(c("The categorical variables are:",
             colnames(Credit)[col_class =="factor"],
             "\n"))

writeLines(c("The numerical variables are:",
             colnames(Credit)[col_class =="integer"],
             "\n"))
```

There are 3 numerical variables (`Age`, `No_Credits` and `No_dependents`) and 15 categorical variables.

4. The variable `Default` is the variable we want to predict. Is the dataset balanced?

```r
### Count the number of observations from each category
table(Credit$Default)

### Check how this binary variable will be encoded
contrasts(Credit$Default)

### Relevel Credit$Default to have category "Default" encoded as 1
Credit$Default<-relevel(Credit$Default,ref="No Default")
contrasts(Credit$Default)
```

The target variable `Default` is binary ("Default" or "No Default"). The dataset is imbalanced since 70% of the observations belong to the category explicitly named "No Default" and 30% to the category explicitly named "Default". However, it can be considered as relatively balanced for a credit dataset. Indeed, the proportion of "Default" is often much lower. Hence, we will not consider it as an issue for the rest of the exercises.

5. Split the data set into a training set and a test set (70-30%) keeping the same proportion of "Default" and "No Default" in both sets.

```r
### Install and load the package named caret
install.packages("caret")
library(caret)

### Set the seed to make results reproducible
```

```r
set.seed(123)

### Get the indices of a training set that maintain the same proportion of "No Default
    "
i_train<-createDataPartition(Credit$Default,p=0.7)[[1]]

### Split the whole data set into a train set and a test set
Train_set <- Credit[i_train,]
Test_set <- Credit[-i_train,]

### Verify that the stratification worked
table(Train_set$Default)
table(Test_set$Default)
```

Note that in order to make the results reproducible, we set the seed of the random number generator before randomly splitting the dataset. The same procedure must be followed for non-deterministic algorithms like bagging or random forests.

Exercise 3: Predict Default using a logistic regression.

1. Fit a logistic regression on the training set to predict the binary variable `Default` using the whole set of predictors. Fit another logistic regression using only `Age` and `Status`.

```r
### Fit a logistic regression using all the predictors and display the results
log_mod <- glm(Default ~ ., data = Train_set, family = binomial("logit"))

### Fit a logistic regression using only the predictors "Age" and "Status" (easier to
    interpret)
log_mod1<-glm(Default~Age+Status,data=Train_set, family=binomial("logit"))
```

If we fit a model using only `Age` and `Status`, we can see that there is only three out of the four categories from `Status` that actually have a coefficient. The category "Female" is put aside to circumvent colinearity between predictors.

2. What are the significant predictors? As a matter of simplicity use only the predictors `Age` and `Status` ?

```r
### Show the summary of log_mod1 (see above)
summary(log_mod1)

### you can display the results in anexportable way
library(broom)
tidy(log_mod1)
```

We can see that if we set the threshold at $\alpha = 5\%$ for the coefficient nullity test, only `Status - Male divorced` is considered not significant.

3. Fit a lasso logistic regression and optimize the regularization parameter using cross-validation. AUC will be used to select the regularization/penalty parameter.

```r
### Load the packages glment to perform the lasso regression
library(glmnet)

### Create a design matrix of the predictors
X_Train<-model.matrix(Default~.,Train_set)
X_Test<-model.matrix(Default~.,Test_set)

### Using cross-validation, fit lasso regression with different penalties
cv_lasso_mod <- cv.glmnet(X_Train, y=Train_set$Default, alpha=1,type.measure="auc",
    family="binomial")

```

```r
### Plot the results
plot(cv_lasso_mod)

### Identify the penalty that gives the best AUC
(best_lambda <- cv_lasso_mod$lambda.min)

### Fit a lasso regression with the best lambda on the whole train set
best_lasso_mod<- glmnet(X_Train, y=Train_set$Default, alpha=1,family="binomial",lambda
    =best_lambda)

### Show the predictors with non-zero coefficient
library(broom)
tidy(best_lasso_mod)
```

We can see that out of the 50 variables in the model, only 35 have non-zero coefficients thanks to regularization.

4. Compare the AUC of the two models (before and after regularization) on the test set and conclude.

```r
### Load the package "ROCR"
library(ROCR)

### Make predictions on the test set
fit_log <-predict(log_mod,newdata=Test_set,type="response)
fit_lasso <- predict(best_lasso_mod,newx=X_Test,type="response")

### Create intermediary prediction object to compute the AUC
pred_log<-prediction(fit_log,Test_set$Default)
pred_lasso<-prediction(fit_lasso,Test_set$Default)

### Compute and display the AUC for both models
(AUC_log=performance(pred_log,measure="auc")@y.values[[1]])
(AUC_lasso=performance(pred_lasso,measure="auc")@y.values[[1]])
```

Exercise 4: Predict Default using tree-based methods.

1. Fit a tree to classify the variable `Default`.

```r
### Load library trees
library(tree)

### Train a tree on the training set using the Gini index
basic_tree <- tree(Default~., data=Train_set,split="gini")
```

2. Display the confusion matrix of the predictions on the test set.

```r
### Make prediction on the test set
pred_basic_tree <- predict(basic_tree, Test_set, type = "class")

### Confusion matrix
(confusion_tree <- table(pred_basic_tree, Test_set$Default))
```

3. Try to improve your classifier with bagging (optional: modify the cutoff to improve your detection of Default). Why modifying the cutoff is of particular interest when trying to predict default?

```r
### Load the package randomForest
library(randomForest)

### Set the seed to make results reproducible
set.seed(123)

```

```r
### Train a bagging algorithm on the training set (cutoff is set to improve "Default" detection)
bagging_tree <-
  randomForest(
    Default ~ .,
    data = Train_set,
    mtry = ncol(Train_set) - 1, #number of predictors used at each split
    importance = TRUE,
    cutoff = c(0.2, 0.8),# see ?randomForest
    ntree = 500
  )

### Make predicitions on the test set and display the confusion matrix
pred_bagging_tree <- predict(bagging_tree, Test_set, type = "class")
(confusion_bagging <- table(pred_bagging_tree, Test_set$Default))
```

4. Use a random forest algorithm to improve your classifier. What are the most important variables?

```r
### Set the seed to make results reproducible
set.seed(123)

### Train a random forest algorithm on the training set (cutoff is set to improve "Default" detection)
RF <-
  randomForest(
    Default ~ .,
    data = Train_set,
    mtry = round(sqrt(ncol(Train_set) - 1), 0),
    importance = TRUE,
    cutoff = c(0.2, 0.8),
    ntree = 500
  )

### Make predicitions on the test set and display the confusion matrix
pred_RF <- predict(RF, Test_set, type = "class")
(confusion_RF <- table(pred_RF, Test_set$Default))

### Display the variable importance Plot
importance(RF)
varImpPlot(RF)
```