Piyush Agrawal

Table of Contents

# This file contains Week 3 + Week 4 Practice content. For Week 4 practice, please search for ‘Week 4’

# Intro

As always, let’s start with clearing the workspace and load required packages. For this exercise, we will use the dataset, *Default* dataset, which is available from *ISLR* library. ISLR packages provide several datasets. The description of datasets are available here: <http://cran.r-project.org/web/packages/ISLR/ISLR.pdf>

rm(list = ls()) # clear the workspace   
library(ISLR) # load ISLR data package  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(ggplot2)

# Data Preparation

Examine the dataset. It contains four variables, default, student,balance, and income.

Default<-as\_tibble(Default)  
Default

## # A tibble: 10,000 × 4  
## default student balance income  
## <fct> <fct> <dbl> <dbl>  
## 1 No No 730. 44362.  
## 2 No Yes 817. 12106.  
## 3 No No 1074. 31767.  
## 4 No No 529. 35704.  
## 5 No No 786. 38463.  
## 6 No Yes 920. 7492.  
## 7 No No 826. 24905.  
## 8 No Yes 809. 17600.  
## 9 No No 1161. 37469.  
## 10 No No 0 29275.  
## # ℹ 9,990 more rows

glimpse(Default)

## Rows: 10,000  
## Columns: 4  
## $ default <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No, No, No…  
## $ student <fct> No, Yes, No, No, No, Yes, No, Yes, No, No, Yes, Yes, No, No, N…  
## $ balance <dbl> 729.5265, 817.1804, 1073.5492, 529.2506, 785.6559, 919.5885, 8…  
## $ income <dbl> 44361.625, 12106.135, 31767.139, 35704.494, 38463.496, 7491.55…

head(Default) # show the first six rows

## # A tibble: 6 × 4  
## default student balance income  
## <fct> <fct> <dbl> <dbl>  
## 1 No No 730. 44362.  
## 2 No Yes 817. 12106.  
## 3 No No 1074. 31767.  
## 4 No No 529. 35704.  
## 5 No No 786. 38463.  
## 6 No Yes 920. 7492.

tail(Default) # show the last six rows

## # A tibble: 6 × 4  
## default student balance income  
## <fct> <fct> <dbl> <dbl>  
## 1 No Yes 172. 14956.  
## 2 No No 712. 52992.  
## 3 No No 758. 19661.  
## 4 No No 845. 58636.  
## 5 No No 1569. 36669.  
## 6 No Yes 201. 16863.

names(Default) # variable names

## [1] "default" "student" "balance" "income"

nrow(Default) # the number of rows

## [1] 10000

ncol(Default) # the number of columns

## [1] 4

summary(Default) # basic summary statistics of the variables in Default dataset (default, student, balance, income)

## 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

# frequency table   
summary(Default$default) # summary of default variable

## No Yes   
## 9667 333

table(Default$default) # contingency table: frequency of each case (yes/no) in default variable

##   
## No Yes   
## 9667 333

table(Default$student) # contingency table: frequency of each case (yes/no) in student variable

##   
## No Yes   
## 7056 2944

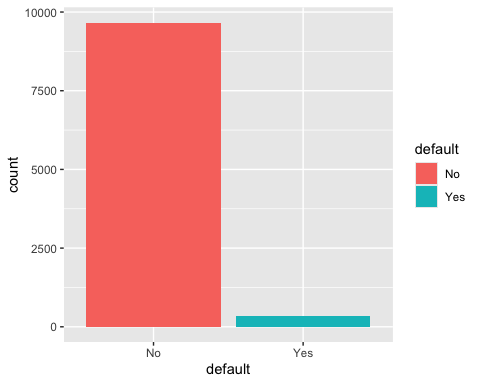
table(Default$default, Default$student) # cross-tabulation (first attribute: row, second attribute: column)

##   
## No Yes  
## No 6850 2817  
## Yes 206 127

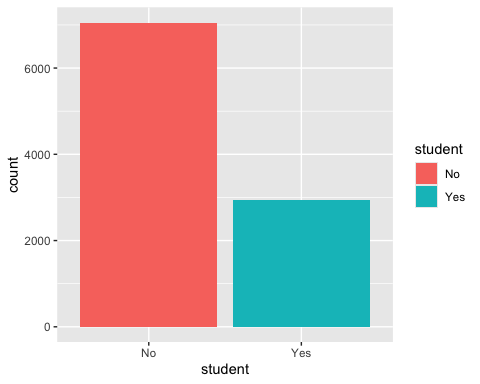
# Data Visualization

## Bar chart

Default %>%  
 ggplot(aes(x=default,fill=default)) +  
 geom\_bar()

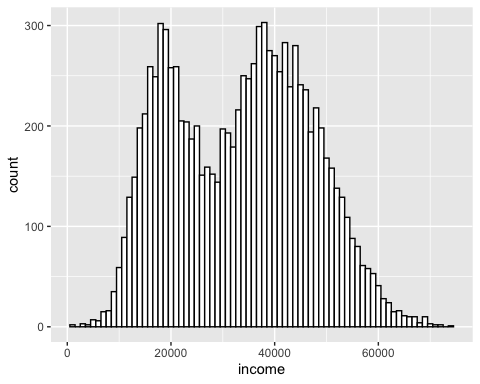


Default %>%  
 ggplot(aes(x=student,fill=student)) +  
 geom\_bar()

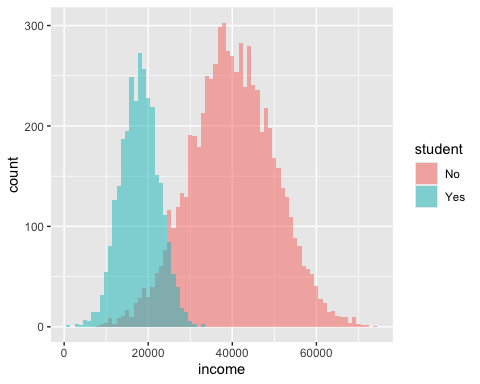


## Histograms

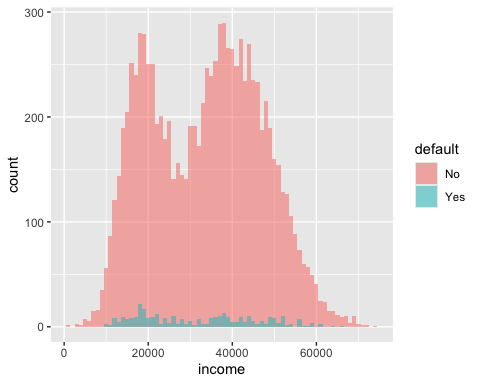
Default %>%  
 ggplot(aes(x=income)) +  
 geom\_histogram(binwidth=1000, colour="black",fill="white")



Default %>%  
 ggplot(aes(x=income,fill=student)) +  
 geom\_histogram(binwidth=1000,alpha=.5,position="identity")

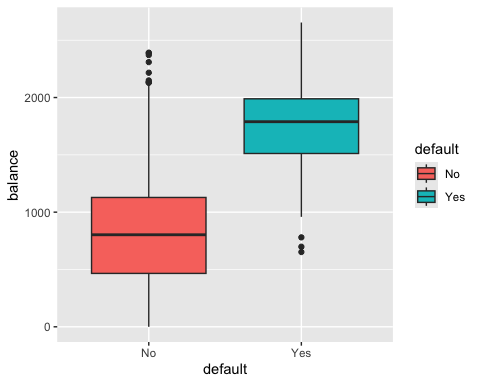


Default %>%  
 ggplot(aes(x=income,fill=default)) +  
 geom\_histogram(binwidth=1000,alpha=.5,position="identity")

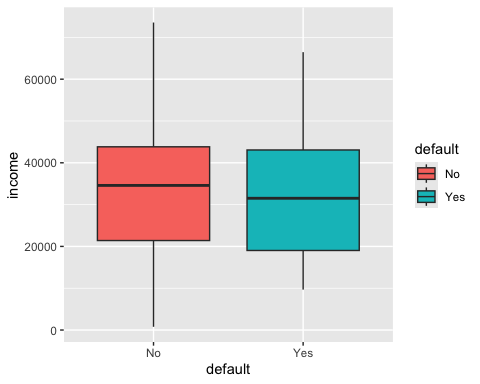


## Boxplots

ggplot(Default,aes(x=default,y=balance,fill=default))+geom\_boxplot()

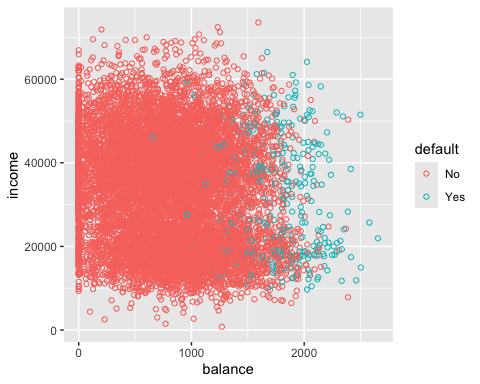


ggplot(Default,aes(x=default,y=income,fill=default))+geom\_boxplot()



## Scatter plots

Default %>%  
 ggplot(aes(x=balance,y=income,color=default)) +  
 geom\_point(shape=1)



How do these visualizations help us predict default?

# Week 3: Modeling - Tree-based classification model

## Classification Trees

We need *rpart* package for Classification Trees. More on rpart : <http://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf>

rpart.plot is a package for visualizing classification tree models.

library(rpart)  
library(rpart.plot)

Let’s build a model.

ct\_model<-rpart(default~student+balance+income, # model formula  
 data=Default, # dataset  
 method="class", # "class" indicates a classification tree model   
 control=rpart.control(cp=0,maxdepth=4)) # tree control parameters.

You can use rpart function to build a regression tree, but we will not do it in this class. method="class" indicates that we want to build a classification tree model.

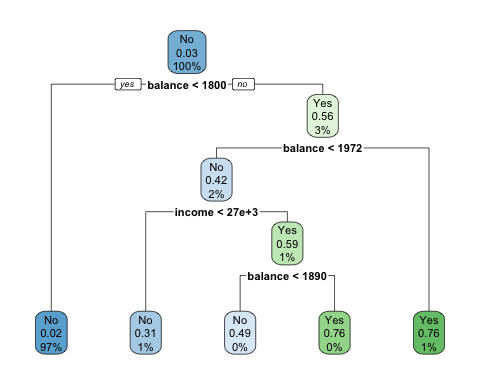
By changing the parameter values for control, you can change how the model is trained (and shaped). Try changing the values and see how the model changes. For more information, ?rpart.control. These are some examples.

* minsplit: minimum number of data points required to attempt a split
* cp: complexity parameter
* maxdepth: depth of a classification tree

Which value should we choose? We will discuss it later with model evaluation (Week 5).

Next, let’s visualize the tree model.

rpart.plot(ct\_model) # tree plot



sum(Default$balance >= 1800)

## [1] 288

sum(Default$balance >= 1800 & Default$default=="Yes")

## [1] 162

sum(Default$balance >= 1800 & Default$balance < 1972)

## [1] 170

sum(Default$balance >= 1800 & Default$balance < 1972 & Default$default=="Yes")

## [1] 72

sum(Default$balance >= 1800 & Default$balance < 1972 & Default$income < 27000)

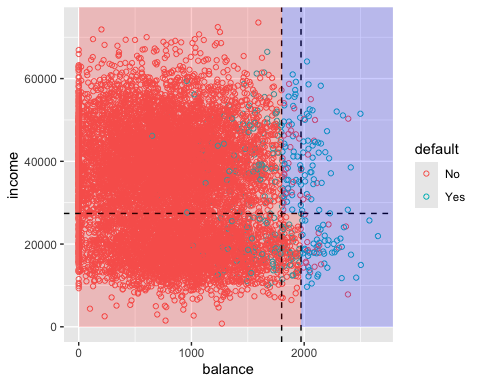
## [1] 102

Default%>%filter(balance>=1800 & balance<1972 & income<27000 & default=="Yes")%>%nrow()

## [1] 32

Here, I tried to visualize the classification tree model results on the scatterplot. It could be done, because the model used the two numeric variables.

Default %>%  
 ggplot(aes(x=balance,y=income,color=default)) +  
 geom\_point(shape=1)+  
 geom\_vline(xintercept=1800.002,linetype="dashed")+  
 geom\_vline(xintercept=1971.915,linetype="dashed")+  
 geom\_hline(yintercept=27401.2,linetype="dashed")+  
 annotate("rect",xmin=1800.002, xmax=1971.915, ymin=0, ymax=27401.2,fill="red",alpha=0.2)+  
 annotate("rect",xmin=1971.915, xmax=Inf, ymin=0, ymax=Inf,fill="blue",alpha=0.2)+  
 annotate("rect",xmin=0, xmax=1800.002, ymin=0, ymax=Inf,fill="red",alpha=0.2)+  
 annotate("rect",xmin=1800.002, xmax=1971.915, ymin=27401.2, ymax=Inf,fill="blue",alpha=0.2)



#print(ct\_model) # model results

Get the predicted value - class membership (yes or no) –> using a cut-off of 50%.

ct\_pred\_class<-predict(ct\_model,type="class") # class membership (yes or no)   
head(ct\_pred\_class)

## 1 2 3 4 5 6   
## No No No No No No   
## Levels: No Yes

ct\_pred<-predict(ct\_model) # get the predicted values - class probabilities (default)  
head(ct\_pred)

## No Yes  
## 1 0.9823929 0.01760708  
## 2 0.9823929 0.01760708  
## 3 0.9823929 0.01760708  
## 4 0.9823929 0.01760708  
## 5 0.9823929 0.01760708  
## 6 0.9823929 0.01760708

Let’s create a new column in Default: save the predicted probability of default (yes) from the second column of dt\_pred.

Default$ct\_pred\_prob<-ct\_pred[,2]

Alternatively, you can specify a certain cut-off value to assign class membership. You can set the cut-off at 30%, 50%, 80%, or whatever you want.

Default$ct\_pred\_class<-ifelse(Default$ct\_pred\_prob>0.5,"Yes","No")

head(Default)

## # A tibble: 6 × 6  
## default student balance income ct\_pred\_prob ct\_pred\_class  
## <fct> <fct> <dbl> <dbl> <dbl> <chr>   
## 1 No No 730. 44362. 0.0176 No   
## 2 No Yes 817. 12106. 0.0176 No   
## 3 No No 1074. 31767. 0.0176 No   
## 4 No No 529. 35704. 0.0176 No   
## 5 No No 786. 38463. 0.0176 No   
## 6 No Yes 920. 7492. 0.0176 No

Default[253,] # get the information of 253th customer

## # A tibble: 1 × 6  
## default student balance income ct\_pred\_prob ct\_pred\_class  
## <fct> <fct> <dbl> <dbl> <dbl> <chr>   
## 1 No Yes 489. 15159. 0.0176 No

# show the customers whose predicted probability is greater than 70%  
Default%>%  
 filter(ct\_pred\_prob>0.7)

## # A tibble: 143 × 6  
## default student balance income ct\_pred\_prob ct\_pred\_class  
## <fct> <fct> <dbl> <dbl> <dbl> <chr>   
## 1 Yes Yes 2206. 14271. 0.763 Yes   
## 2 Yes No 1890. 48956. 0.76 Yes   
## 3 Yes No 1964. 39055. 0.76 Yes   
## 4 No Yes 2023. 18337. 0.763 Yes   
## 5 Yes No 1992. 42133. 0.763 Yes   
## 6 Yes No 1981. 28128. 0.763 Yes   
## 7 No Yes 2005. 27137. 0.763 Yes   
## 8 Yes No 1903. 53394. 0.76 Yes   
## 9 No No 2113. 34748. 0.763 Yes   
## 10 Yes No 1964. 50554. 0.76 Yes   
## # ℹ 133 more rows

# sort customers by probability of default in descending order  
Default%>%  
 arrange(desc(ct\_pred\_prob))

## # A tibble: 10,000 × 6  
## default student balance income ct\_pred\_prob ct\_pred\_class  
## <fct> <fct> <dbl> <dbl> <dbl> <chr>   
## 1 Yes Yes 2206. 14271. 0.763 Yes   
## 2 No Yes 2023. 18337. 0.763 Yes   
## 3 Yes No 1992. 42133. 0.763 Yes   
## 4 Yes No 1981. 28128. 0.763 Yes   
## 5 No Yes 2005. 27137. 0.763 Yes   
## 6 No No 2113. 34748. 0.763 Yes   
## 7 Yes No 2033. 44998. 0.763 Yes   
## 8 Yes No 2024. 51509. 0.763 Yes   
## 9 No Yes 1994. 14305. 0.763 Yes   
## 10 Yes No 2499. 51504. 0.763 Yes   
## # ℹ 9,990 more rows

## Random Forest

set.seed(1)  
#install.packages("randomForest")  
library(randomForest)

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

rf\_model<-randomForest(default~income+balance+student, # model formula  
 data=Default,ntree=500, cutoff=c(0.5,0.5))

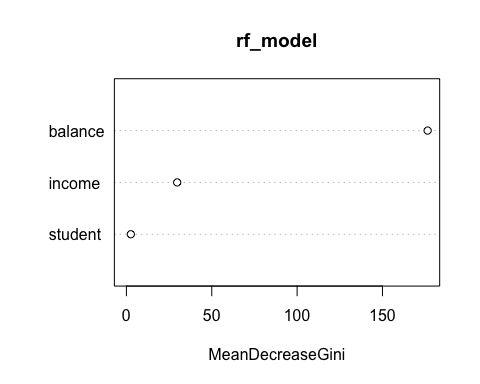
#print(rf\_model)  
head(rf\_model$votes) # indicates the % of trees that voted for each class

## No Yes  
## 1 1 0  
## 2 1 0  
## 3 1 0  
## 4 1 0  
## 5 1 0  
## 6 1 0

head(rf\_model$predicted) # the class favored by more trees (i.e. majority vote wins)

## 1 2 3 4 5 6   
## No No No No No No   
## Levels: No Yes

varImpPlot(rf\_model) # importance of variables



head(rf\_model$vote)

## No Yes  
## 1 1 0  
## 2 1 0  
## 3 1 0  
## 4 1 0  
## 5 1 0  
## 6 1 0

Default$rf\_vote<-predict(rf\_model,type="prob")[,2]  
head(Default)

## # A tibble: 6 × 7  
## default student balance income ct\_pred\_prob ct\_pred\_class rf\_vote  
## <fct> <fct> <dbl> <dbl> <dbl> <chr> <dbl>  
## 1 No No 730. 44362. 0.0176 No 0  
## 2 No Yes 817. 12106. 0.0176 No 0  
## 3 No No 1074. 31767. 0.0176 No 0  
## 4 No No 529. 35704. 0.0176 No 0  
## 5 No No 786. 38463. 0.0176 No 0  
## 6 No Yes 920. 7492. 0.0176 No 0

# Week 4: Modeling - Linear classifier

## Support Vector Machine (SVM)

library(e1071)  
model\_svm<-svm(formula= default ~ balance+income+student, # model formula   
 data=Default, # data set  
 kernel="linear", # this is the form of the decision boundary. Let's start with a linear kernel.   
 cost=0.1) # Cost parameter is for regularization  
model\_svm

##   
## Call:  
## svm(formula = default ~ balance + income + student, data = Default,   
## kernel = "linear", cost = 0.1)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 0.1   
##   
## Number of Support Vectors: 672

The model may not converge, and it is not uncommon. Also, note that it is not an error. It is less desire, but it provides classification results. To improve performance, you may try different cost parameters, or you may even try other kernel functions, other than “linear”. Other option is normalizing data. But we will move on with this result. <https://www.rdocumentation.org/packages/e1071/versions/1.7-14/topics/svm>

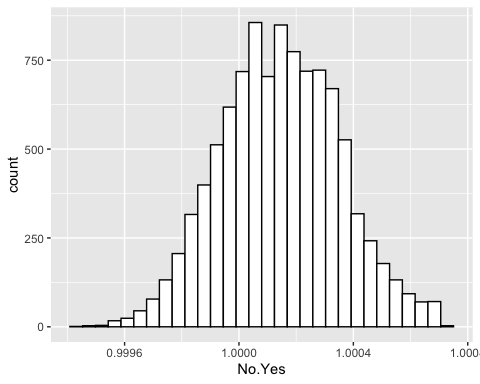
Conceptually, you may interpret decision values as the distance between the observation and the decision boundary. The positive fitted value indicate one class, and negative value indicates the other class.

head(model\_svm$decision.values)

## No/Yes  
## 1 1.0000632  
## 2 1.0004151  
## 3 0.9999982  
## 4 1.0001708  
## 5 1.0000694  
## 6 1.0003995

dv<-data.frame(model\_svm$decision.values)  
  
ggplot(dv,aes(x=No.Yes)) +  
 geom\_histogram(colour="black",fill="white")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



head(model\_svm$fitted) #class prediction result

## 1 2 3 4 5 6   
## No No No No No No   
## Levels: No Yes

table(model\_svm$fitted)

##   
## No Yes   
## 10000 0

predicted\_svm<-predict(model\_svm, Default, decision.values = TRUE) # to get the decision value  
head(attr(predicted\_svm, "decision.values"))

## No/Yes  
## 1 1.0000632  
## 2 1.0004151  
## 3 0.9999982  
## 4 1.0001708  
## 5 1.0000694  
## 6 1.0003995

Default

## # A tibble: 10,000 × 7  
## default student balance income ct\_pred\_prob ct\_pred\_class rf\_vote  
## <fct> <fct> <dbl> <dbl> <dbl> <chr> <dbl>  
## 1 No No 730. 44362. 0.0176 No 0  
## 2 No Yes 817. 12106. 0.0176 No 0  
## 3 No No 1074. 31767. 0.0176 No 0  
## 4 No No 529. 35704. 0.0176 No 0  
## 5 No No 786. 38463. 0.0176 No 0  
## 6 No Yes 920. 7492. 0.0176 No 0  
## 7 No No 826. 24905. 0.0176 No 0  
## 8 No Yes 809. 17600. 0.0176 No 0  
## 9 No No 1161. 37469. 0.0176 No 0  
## 10 No No 0 29275. 0.0176 No 0  
## # ℹ 9,990 more rows

Default$svm\_pred\_class <- predict(model\_svm, Default) #class prediction  
Default$svm\_dv<-c(attr(predicted\_svm, "decision.values"))  
Default

## # A tibble: 10,000 × 9  
## default student balance income ct\_pred\_prob ct\_pred\_class rf\_vote  
## <fct> <fct> <dbl> <dbl> <dbl> <chr> <dbl>  
## 1 No No 730. 44362. 0.0176 No 0  
## 2 No Yes 817. 12106. 0.0176 No 0  
## 3 No No 1074. 31767. 0.0176 No 0  
## 4 No No 529. 35704. 0.0176 No 0  
## 5 No No 786. 38463. 0.0176 No 0  
## 6 No Yes 920. 7492. 0.0176 No 0  
## 7 No No 826. 24905. 0.0176 No 0  
## 8 No Yes 809. 17600. 0.0176 No 0  
## 9 No No 1161. 37469. 0.0176 No 0  
## 10 No No 0 29275. 0.0176 No 0  
## # ℹ 9,990 more rows  
## # ℹ 2 more variables: svm\_pred\_class <fct>, svm\_dv <dbl>

## Logistic Regression

logit\_model<-glm(default~student+balance+income, # generalized linear models  
 family="binomial", # specifying error distribution  
 data=Default) # dataset  
summary(logit\_model)

##   
## Call:  
## glm(formula = default ~ student + balance + income, family = "binomial",   
## data = Default)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.087e+01 4.923e-01 -22.080 < 2e-16 \*\*\*  
## studentYes -6.468e-01 2.363e-01 -2.738 0.00619 \*\*   
## balance 5.737e-03 2.319e-04 24.738 < 2e-16 \*\*\*  
## income 3.033e-06 8.203e-06 0.370 0.71152   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2920.6 on 9999 degrees of freedom  
## Residual deviance: 1571.5 on 9996 degrees of freedom  
## AIC: 1579.5  
##   
## Number of Fisher Scoring iterations: 8

### Use of the model to predict

Default$log\_odd<-predict(logit\_model) # get predicted log odds (default)  
Default$logit\_pred\_prob<-predict(logit\_model,type="response") # get predicted probabilities  
glimpse(Default)

## Rows: 10,000  
## Columns: 11  
## $ default <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No…  
## $ student <fct> No, Yes, No, No, No, Yes, No, Yes, No, No, Yes, Yes, N…  
## $ balance <dbl> 729.5265, 817.1804, 1073.5492, 529.2506, 785.6559, 919…  
## $ income <dbl> 44361.625, 12106.135, 31767.139, 35704.494, 38463.496,…  
## $ ct\_pred\_prob <dbl> 0.01760708, 0.01760708, 0.01760708, 0.01760708, 0.0176…  
## $ ct\_pred\_class <chr> "No", "No", "No", "No", "No", "No", "No", "No", "No", …  
## $ rf\_vote <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …  
## $ svm\_pred\_class <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No…  
## $ svm\_dv <dbl> 1.0000632, 1.0004151, 0.9999982, 1.0001708, 1.0000694,…  
## $ log\_odd <dbl> -6.549544, -6.791338, -4.614261, -7.724689, -6.245449,…  
## $ logit\_pred\_prob <dbl> 1.428724e-03, 1.122204e-03, 9.812272e-03, 4.415893e-04…

Default%>%  
 select("default","student","log\_odd","logit\_pred\_prob")

## # A tibble: 10,000 × 4  
## default student log\_odd logit\_pred\_prob  
## <fct> <fct> <dbl> <dbl>  
## 1 No No -6.55 0.00143   
## 2 No Yes -6.79 0.00112   
## 3 No No -4.61 0.00981   
## 4 No No -7.72 0.000442   
## 5 No No -6.25 0.00194   
## 6 No Yes -6.22 0.00199   
## 7 No No -6.06 0.00233   
## 8 No Yes -6.82 0.00109   
## 9 No No -4.09 0.0164   
## 10 No No -10.8 0.0000208  
## # ℹ 9,990 more rows

1/(1+exp(.5373))

## [1] 0.3688159

With the predicted probabilities, you can sort customers by the predicted probability of default in descending order.

Default%>%  
 arrange(desc(logit\_pred\_prob))%>%  
 select(default, student, balance, income, ct\_pred\_prob, ct\_pred\_class,logit\_pred\_prob)

## # A tibble: 10,000 × 7  
## default student balance income ct\_pred\_prob ct\_pred\_class logit\_pred\_prob  
## <fct> <fct> <dbl> <dbl> <dbl> <chr> <dbl>  
## 1 Yes Yes 2654. 21930. 0.763 Yes 0.978  
## 2 Yes No 2499. 51504. 0.763 Yes 0.974  
## 3 Yes Yes 2578. 25707. 0.763 Yes 0.966  
## 4 Yes No 2413. 38541. 0.763 Yes 0.957  
## 5 No No 2391. 50303. 0.763 Yes 0.953  
## 6 Yes Yes 2503. 14948. 0.763 Yes 0.947  
## 7 Yes No 2344. 51095. 0.763 Yes 0.939  
## 8 Yes Yes 2462. 11879. 0.763 Yes 0.933  
## 9 Yes No 2288. 52044. 0.763 Yes 0.918  
## 10 Yes Yes 2415. 17430. 0.763 Yes 0.916  
## # ℹ 9,990 more rows

And use a different cut-off for class prediction using ifelse().

Default$logit\_pred\_class<-ifelse(Default$logit\_pred\_prob>0.5,"Yes","No")  
Default

## # A tibble: 10,000 × 12  
## default student balance income ct\_pred\_prob ct\_pred\_class rf\_vote  
## <fct> <fct> <dbl> <dbl> <dbl> <chr> <dbl>  
## 1 No No 730. 44362. 0.0176 No 0  
## 2 No Yes 817. 12106. 0.0176 No 0  
## 3 No No 1074. 31767. 0.0176 No 0  
## 4 No No 529. 35704. 0.0176 No 0  
## 5 No No 786. 38463. 0.0176 No 0  
## 6 No Yes 920. 7492. 0.0176 No 0  
## 7 No No 826. 24905. 0.0176 No 0  
## 8 No Yes 809. 17600. 0.0176 No 0  
## 9 No No 1161. 37469. 0.0176 No 0  
## 10 No No 0 29275. 0.0176 No 0  
## # ℹ 9,990 more rows  
## # ℹ 5 more variables: svm\_pred\_class <fct>, svm\_dv <dbl>, log\_odd <dbl>,  
## # logit\_pred\_prob <dbl>, logit\_pred\_class <chr>