Practical: Classification 1

David Vichansky [6819516]

10-12-2020

Load the packages.

```r
library(ISLR)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.1
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

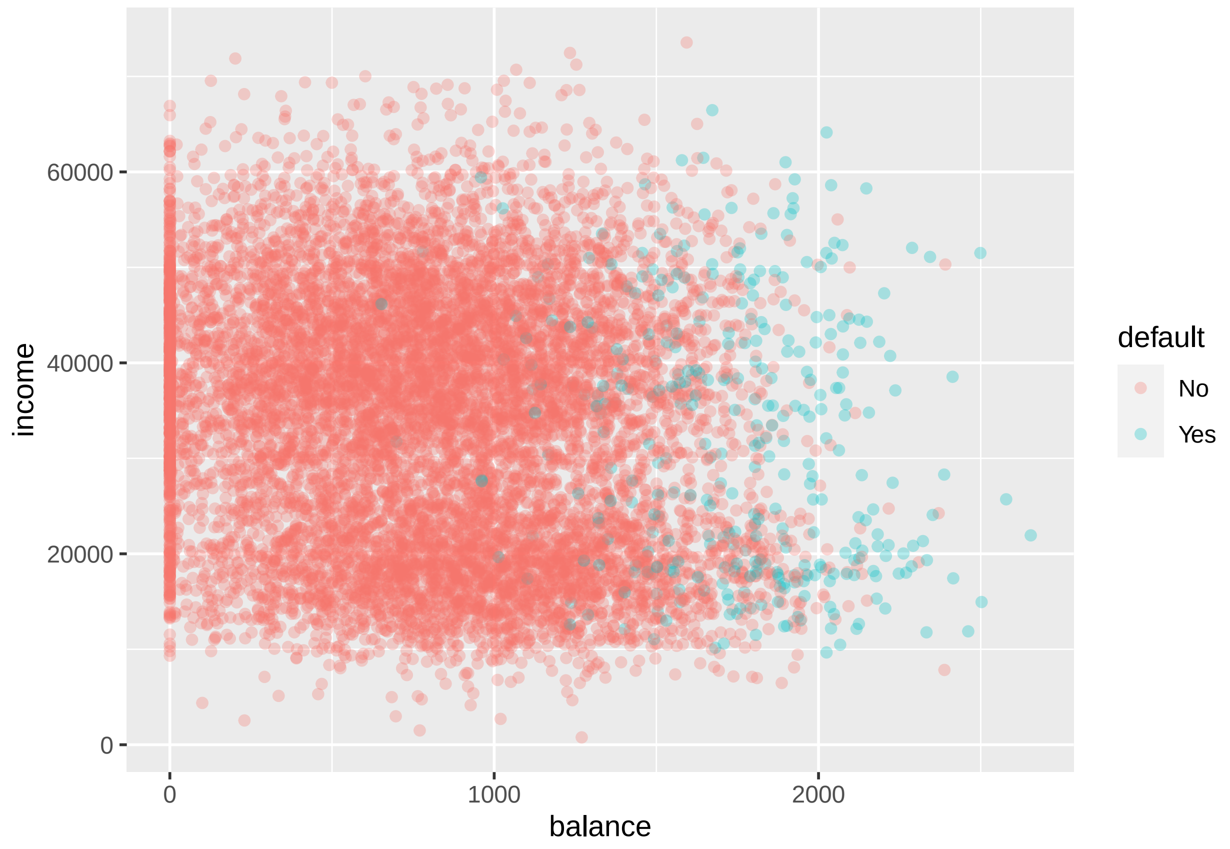
```r
library(haven)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(class)
library(rpart)
library(rpart.plot)
```

1. Create a scatterplot of the Default dataset, where balance is mapped to the x position, income is mapped to the y position, and default is mapped to the colour. Can you see any interesting patterns already?

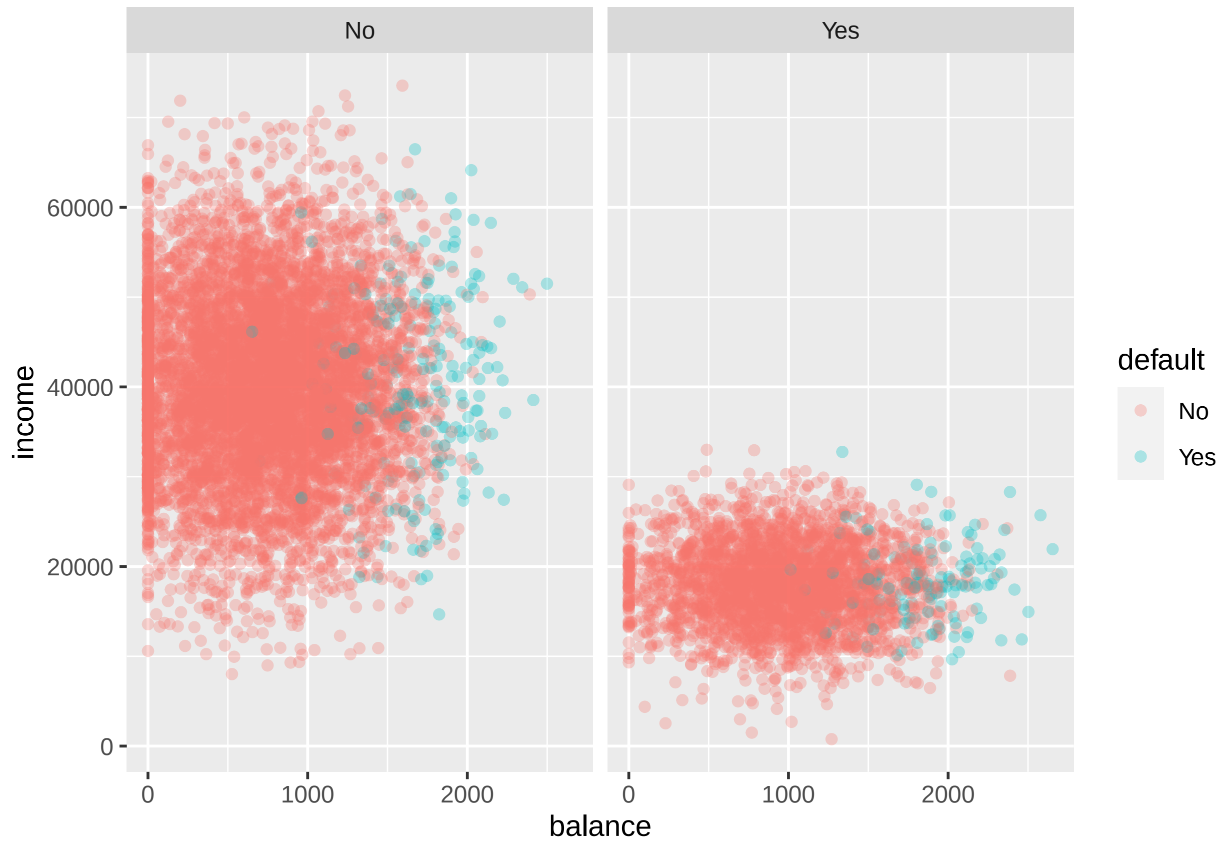
```r
##Data
#Default
ggplot(Default, aes(x = balance, y = income)) +
  geom_point(mapping = aes(color = default), alpha = 0.3)
```



We observe that most defaults suggest that people with a higher credit balance. There is also no suggestion that income influences whether a person may default or not.

2. Add facet_grid(cols = vars(student)) to the plot. What do you see?

```
ggplot(Default, aes(x = balance, y = income)) +
  geom_point(mapping = aes(color = default), alpha = 0.3) +
  facet_grid(cols = vars(student))
```



Students income is lower than the rest of the population. Students also with the same balance as non-students are more likely to default on their debt.

3. Transform "student" into a dummy variable using ifelse() (0 = not a student, 1 = student). Then, randomly split the Default dataset into a training set default_train (80%) and a test set default_test (20%)

```
Default <- Default

#Create variable
student_dummy <- ifelse(Default$student == "No", 0, 1)

#Apned to table. Note: we chose to 'replace' student column with quantitative data
Default <- Default %>%
  mutate(student = student_dummy)

#Split data set into training and test
train_size <- floor(0.80 * nrow(Default))

train_ind <- sample(seq_len(nrow(Default)), size = train_size, replace = FALSE)

#Train data set
default_train <- Default[train_ind, ]
```

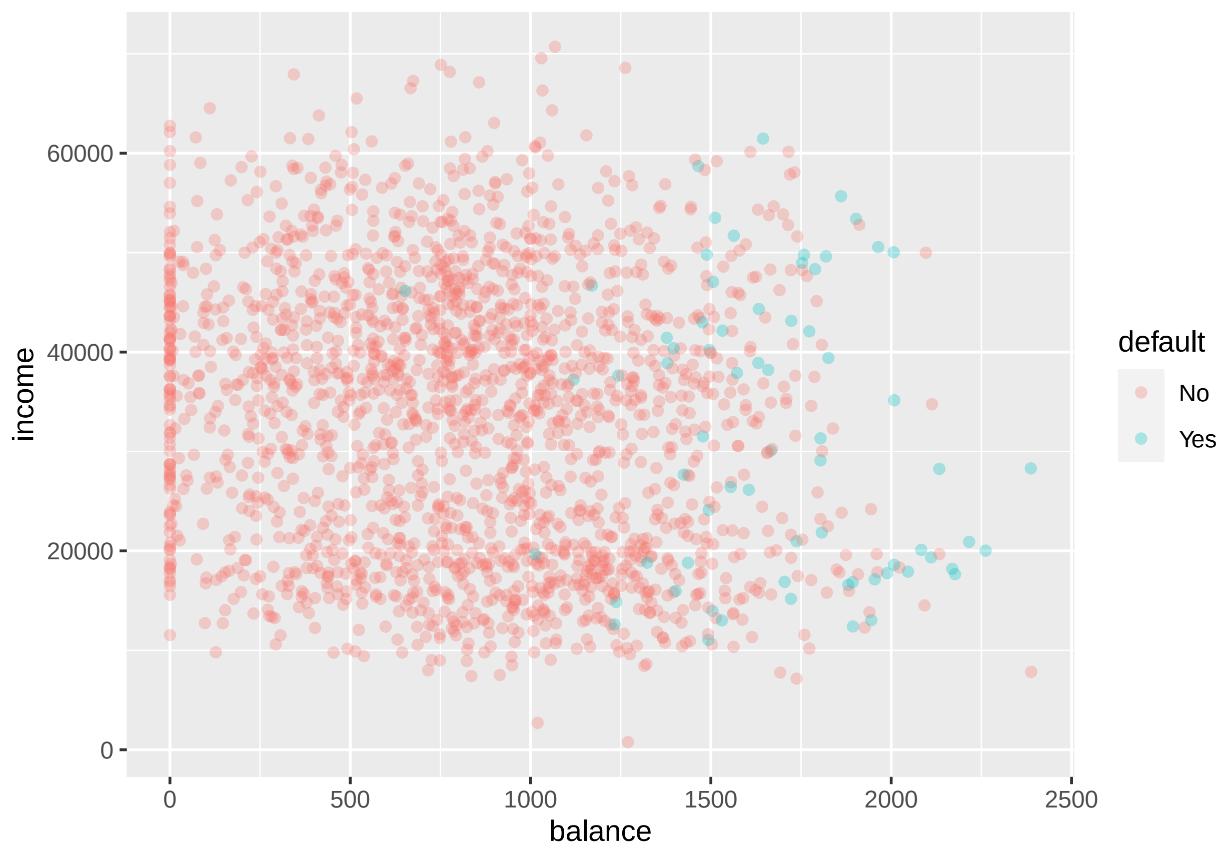
```
#Test data set
default_test <- Default[-train_ind, ]
```

4. Create class predictions for the test set using the knn() function. Use student, balance, and income (but no basis functions of those variables) in the default_train dataset. Set k to 5. Store the predictions in a variable called knn_5_pred.

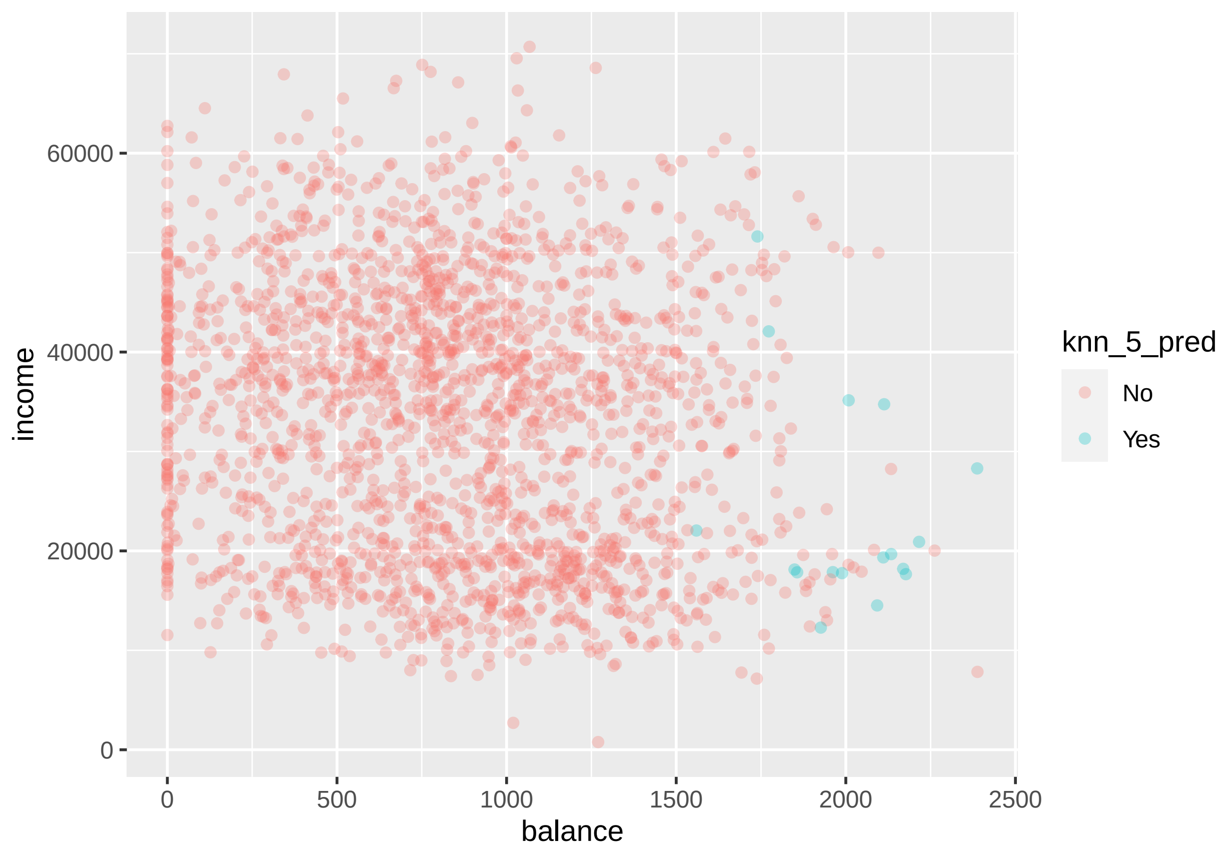
```
#If several observations are tied as nearest neighbors, then R will randomly break the
set.seed (1)
#Use knn()
#The third argument inside knn() is a vector containing the class labels from the trai
#Remove column 'default' which has index 1 in the data frame
knn_5_pred <- knn(default_train[-1], default_test[-1], default_train$default, k=5)
```

5. Create two scatter plots with income and balance as in the first plot you made. One with the true class (default) mapped to the colour aesthetic, and one with the predicted class (knn_5_pred) mapped to the colour aesthetic. Hint: Add the predicted class knn_5_pred to the default_test dataset before starting your ggplot() call of the second plot. What do you see?

```
#Add 'knn_5_pred' to default_test data frame
default_test2 <- default_test %>%
  mutate(knn_5_pred = knn_5_pred)

#Make plot using default_test data
ggplot(default_test2, aes(x = balance, y = income)) +
  geom_point(mapping = aes(color = default), alpha = 0.3)
```



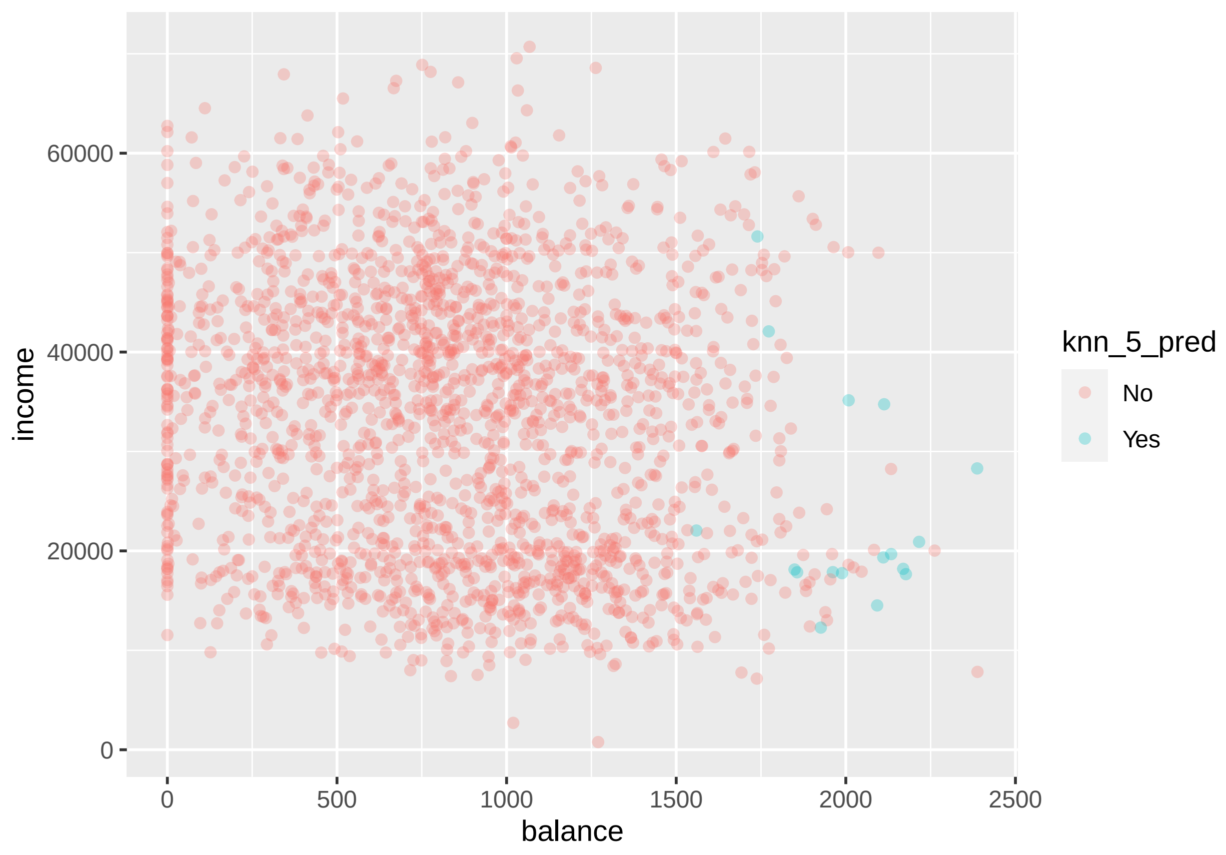
```
#Plot default_test data but by using predicted class knn_5_pred
ggplot(default_test2, aes(x = balance, y = income)) +
  geom_point(mapping = aes(color = knn_5_pred), alpha = 0.3)
```



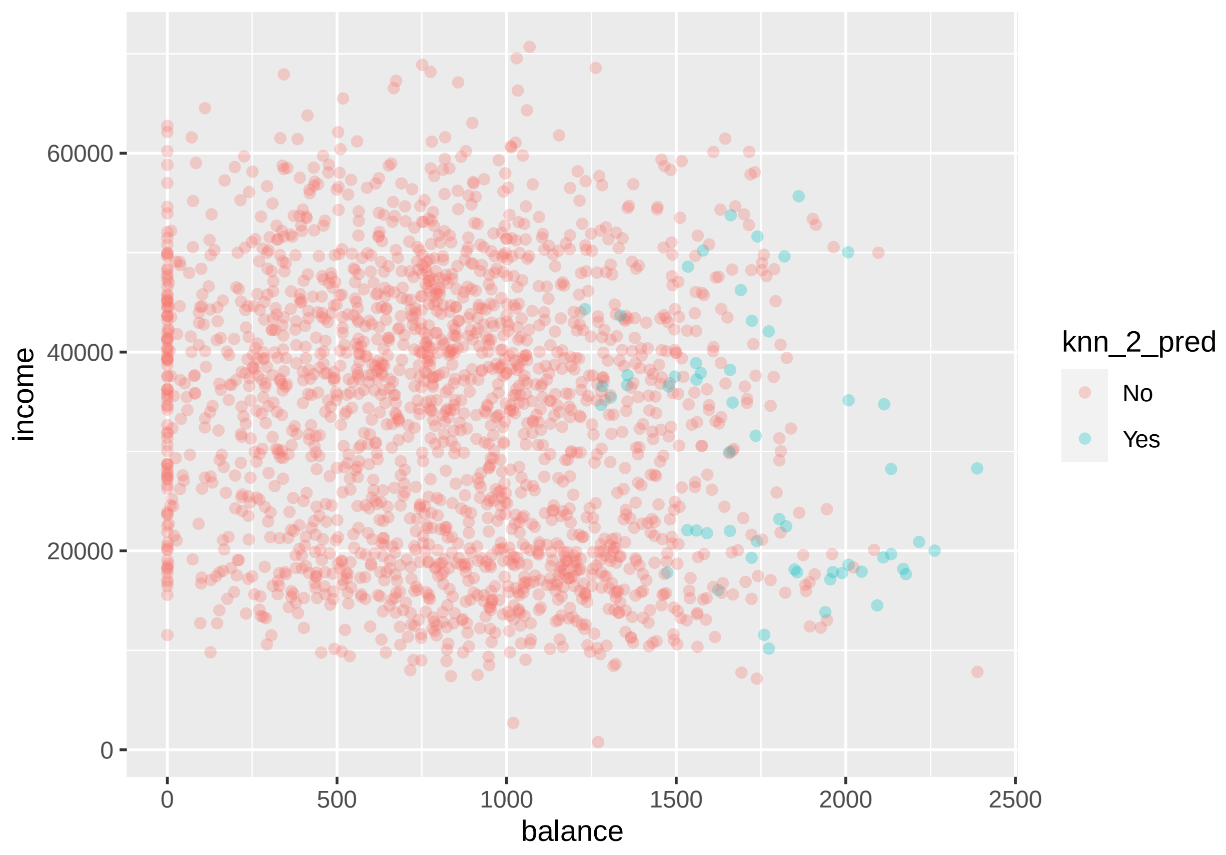
We observe that the test data predicts less individuals to go bankrupt.

6. Repeat the same steps, but now with a knn_2_pred vector generated from a 2-nearest neighbours algorithm. Are there any differences?

```r
knn_2_pred <- knn(default_train[-1], default_test[-1], default_train$default ,k=2)

#Make plot using default_test data
#ggplot(default_test2, aes(x = balance, y = income)) +
#  geom_point(mapping = aes(color = default), alpha = 0.3)

#Plot default_test data but by using predicted class knn_5_pred
ggplot(default_test2, aes(x = balance, y = income)) +
  geom_point(mapping = aes(color = knn_5_pred), alpha = 0.3)
```



```r
#Plot default_test data but by using predicted class knn_2_pred
ggplot(default_test2, aes(x = balance, y = income)) +
  geom_point(mapping = aes(color = knn_2_pred), alpha = 0.3)
```



We observe that when the 'k' (the number of neighbours considered) is closer to the types of the actual variable we are plotting (i.e. 2, being 'yes' and 'no' for default) then the knn() predictior function shows a greater variability in the prediction for k=2 than k=5, where in the case of the latter more of the data points were clumped together to show as not likely to default. Variance decreases for k=5 becasue we are taking more points as neighbours into account.

7. What would this confusion matrix look like if the classification were perfect?

```
#The confusion matrix values will change depending on the test/train split
table(true = default_test$default, predicted = knn_2_pred)
```

```
##      predicted
## true    No  Yes
##   No  1894   37
##   Yes   49   20
```

For a perfectly classified confusion matrix we would have correct predictions on the diagonal of the matrix (true positive and true negative) and we would have zero for all values outside the diagonal (false positive and false negative). default_test$default

8. Make a confusion matrix for the 5-nn model and compare it to that of the 2-nn model. What do you conclude?

```
table(true = default_test$default, predicted = knn_5_pred)
```

```
##      predicted
## true    No  Yes
##   No  1922    9
##   Yes   61    8
```

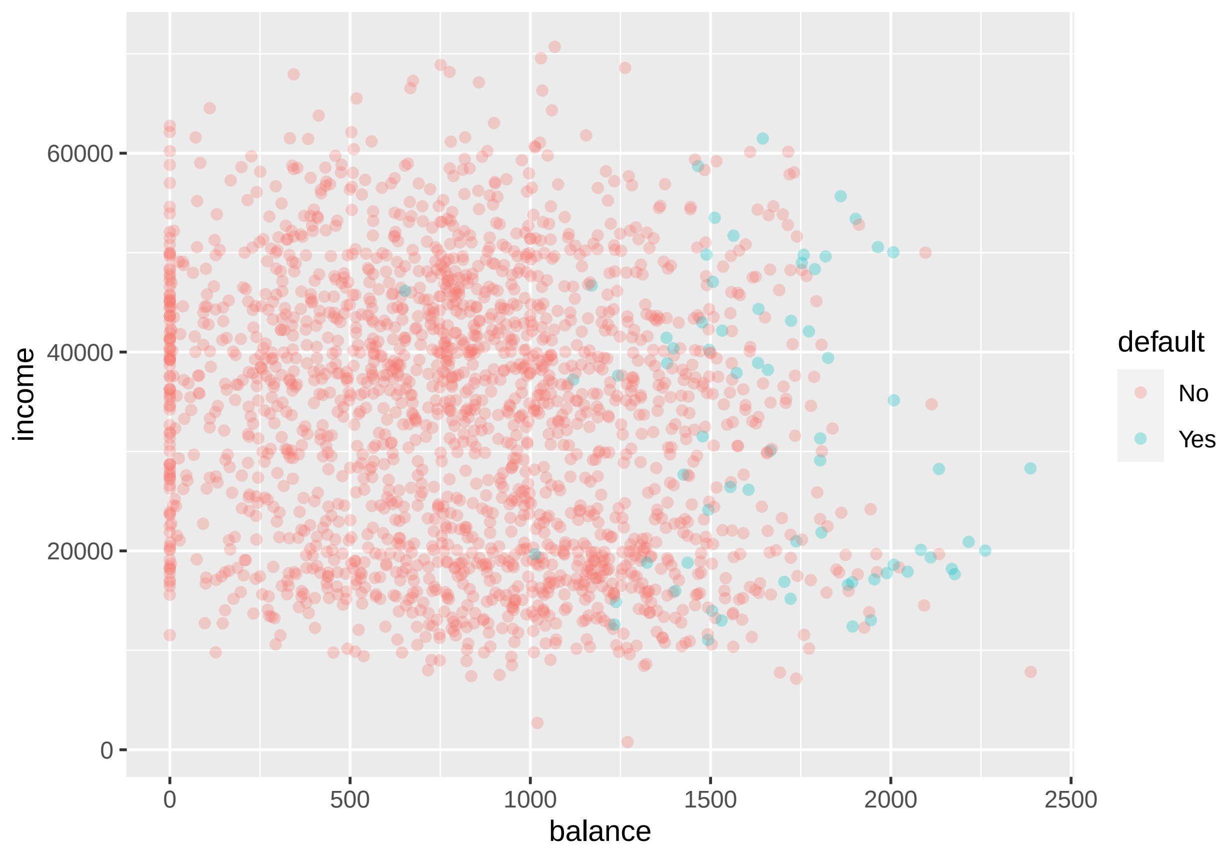
There is not much difference but it does seem that in the knn-prediction model with k=5 we obtain less false negative but more false positives. This would explain for the graphs above why in the case of k=2 we have more variation in the classification, and that perhaps more individuals are being incorrectly classfied if they perhaps have a smaller income than their neighbours. The k=2 model overfits the model but this can make sense if the lending industry wish to act 'safer rather than sorry'.

9. Use glm() with argument family = binomial to fit a logistic regression model lr_mod to the default_train data.

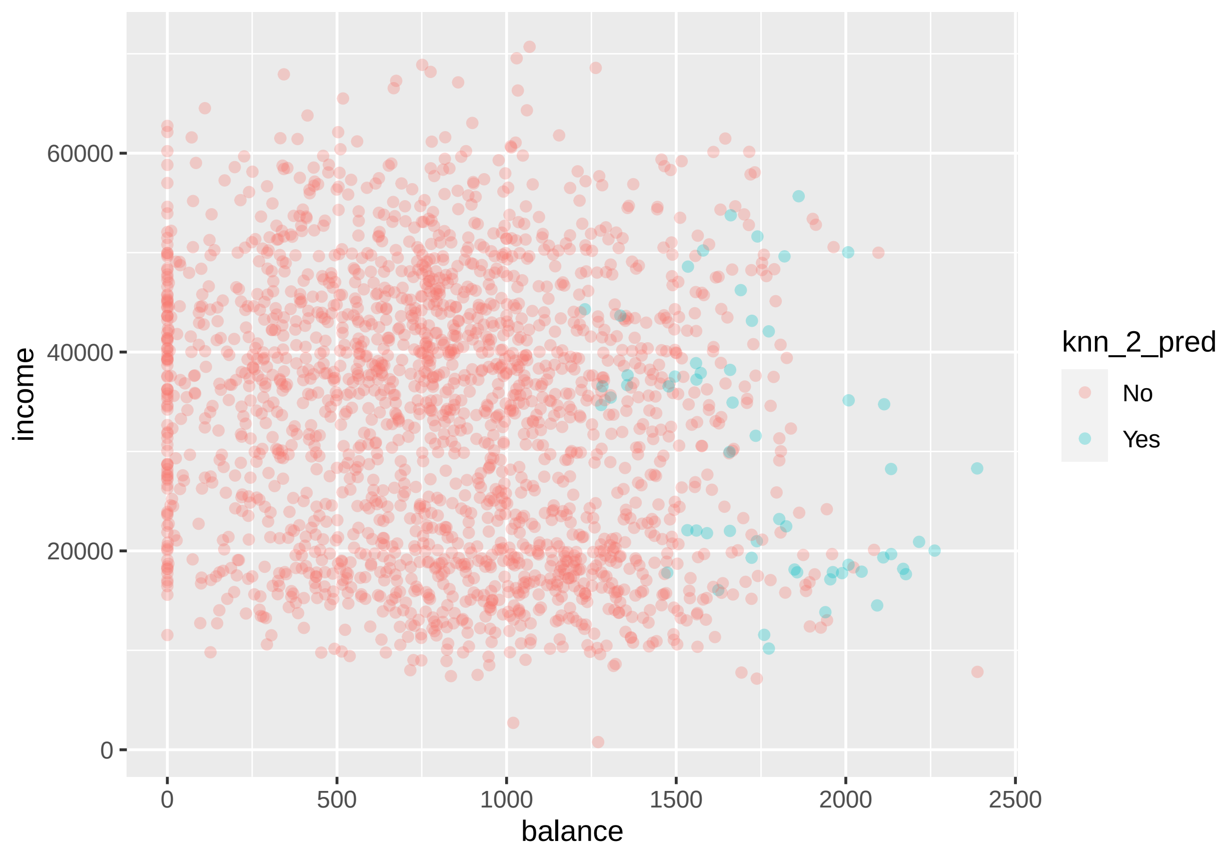
```r
#We train the model on only the training data
lr_mod <- glm(default   student + balance + income, data=default_train, family = binomia

#To fit the model using predict() we have to use the test data next
```

10. Visualise the predicted probabilities versus observed class for the training dataset in lr_mod. You can choose for yourself which type of visualisation you would like to make. Write down your interpretations along with your plot.

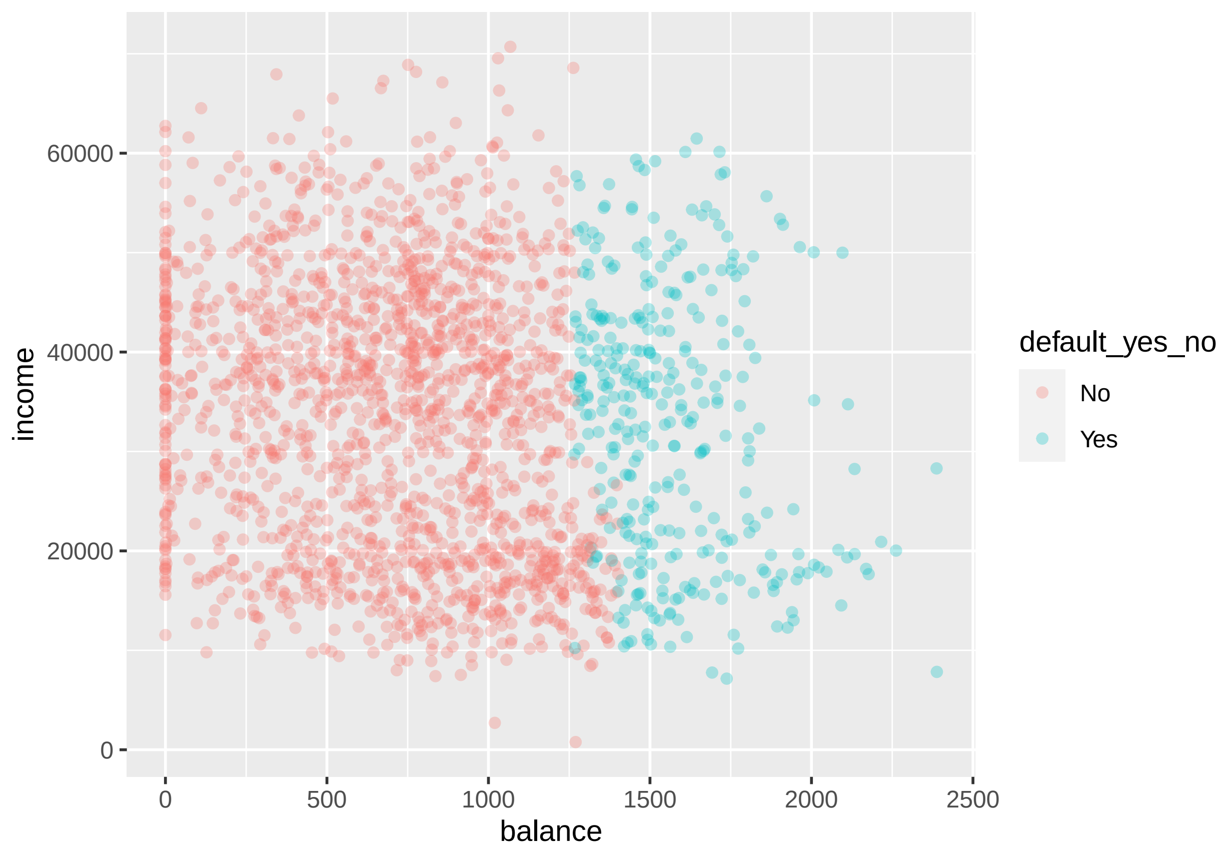
```r
#Predict probabilities for default
glm.probs <- predict(lr_mod, newdata = default_test, type = "response")

##We must calculate some threshold for which we say an individual is bankrupt with som
#Sum the second column of the confusion matrix with k=2 predictors, taking that as a p
confusion_matrix <- table(true = default_test$default, predicted = knn_2_pred)

#Calculate probability of defaulting
default_prob <- sum(confusion_matrix[,2]) / sum(confusion_matrix)

##Apply this probability using a 'logic' argument to the test data set together with t
#Create vector to store "yes" and "no" values
glm.pred = rep("Yes", nrow(default_test))

#Insert threshold value we calculated earlier for probability of default
glm.pred[glm.probs <= default_prob] = "No"

default_test3 <- default_test %>%
  mutate(default_yes_no = glm.pred)

##Now plot
#Plot observed values for default_test
ggplot(default_test, aes(x = balance, y = income)) +
  geom_point(mapping = aes(color = default), alpha = 0.3)
```



```r
#Plot default_test2 data but by using predicted class knn_2_pred
ggplot(default_test2, aes(x = balance, y = income)) +
  geom_point(mapping = aes(color = knn_2_pred), alpha = 0.3)
```



```r
#Plot default_test3 data but by using predicted class default_yes_no from the linear m
ggplot(default_test3, aes(x = balance, y = income)) +
  geom_point(mapping = aes(color = default_yes_no), alpha = 0.3)
```



For the linear model we followed a similar example from the text book 'ISLR' whereby a threshold for defaulting on a debt was set using the confusion matrix (the sum of defaults, i.e. true negatives + false negatives, in numerator and total sum in denominator). We had then appended this information to our test data frame (renaming this to 'default_test3') and plotted the scatter plots for comparison.

11. Look at the coefficients of the lr_mod model and interpret the coefficient for balance. What would the probability of default be for a person who is not a student, has an income of 40000, and a balance of 3000 dollars at the end of each month? Is this what you expect based on the plots we've made before?

```
#Intercept
coef(lr_mod)
```

```
##   (Intercept)       student       balance        income
## -1.096960e+01 -8.120426e-01  5.896100e-03 -8.086965e-07
```

```
#Recall this is the model we trained our data on: lr_mod <- glm(default  student + ba
predict(lr_mod, newdata = data.frame(student=c(0), income=c(40000), balance=c(3000)), ty
```

```
##         1
## 0.9987537
```

There is a positive relation between the credit balance increasing, i.e. borrowing increases, and the probability of default also increasing. The probability of default is quite high. This is in some way in line with what we would expect because majority of individuals with a balance => 2000 (in our case we predict with 3000) default on their loan, so we can expect individuals with a balance of greater or equal to 3000 to also defualt. What may be slightly surprising is the extent to high large that probability is considering individuals receive a large income.

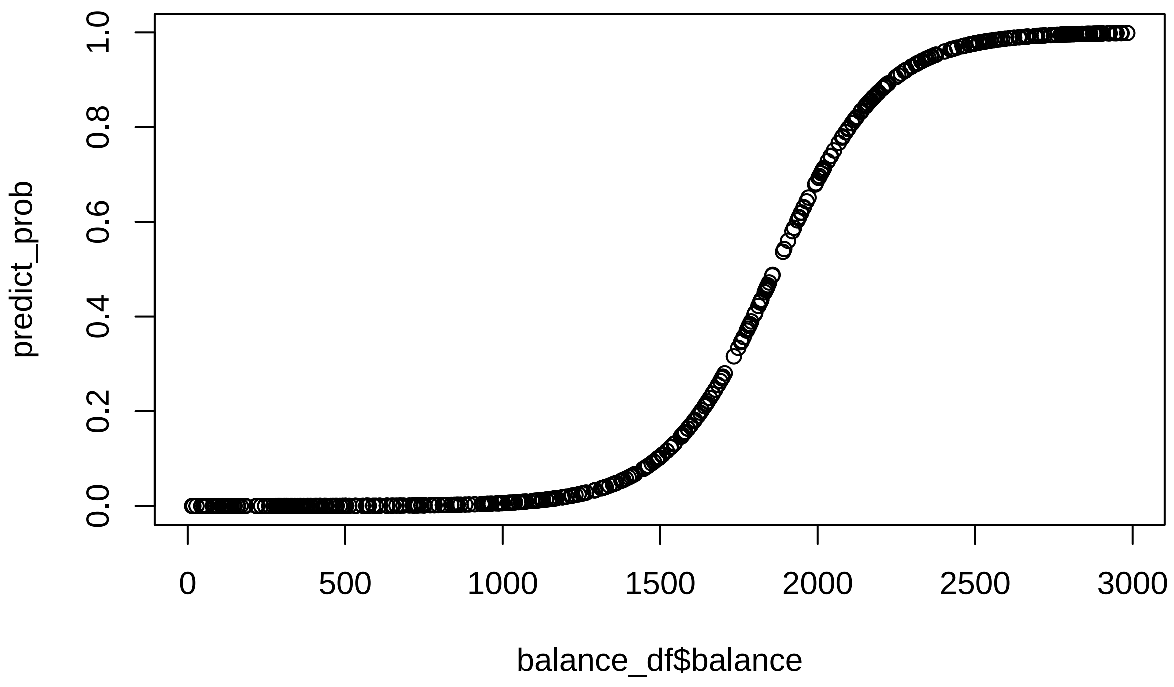
In two steps, we will visualise the effect balance has on the predicted default probability.

12. Create a data frame called balance_df with 3 columns and 500 rows: student always 0, balance ranging from 0 to 3000, and income always the mean income in the default_train dataset.

```r
student <- rep(0, times = 500)
balance <- sample(0:3000, 500, replace=TRUE)
income <- rep(mean(default_train$income), times = 500)

#Bind columns together into data frame
balance_df <- data.frame(student, balance, income)
```

13. Use this dataset as the newdata in a predict() call using lr_mod to output the predicted probabilities for different values of balance. Then create a plot with the balance_df$balance variable mapped to x and the predicted probabilities mapped to y. Is this in line with what you expect?

```r
##Our linear model from earlier was default ~ student + balance + income
##We now want to model this using our newly created data frame

predict_prob <- predict(lr_mod, newdata = balance_df, type="response")

plot(balance_df$balance, predict_prob)
```



Yes, we would expect the probability of default to ramp up pretty rapidly at a certain point and we expect the highest probability of default to happen to individuals with a balance of greater or equal to 2500.

14. Create a confusion matrix just as the one for the KNN models by using a cutoff predicted probability of 0.5. Does logistic regression perform better?

```
#Vector with 500 "no" default values
balance_pred <- rep("No", nrow(balance_df))

#Create cut off with when probabiltiy of default is >50%
balance_pred[predict_prob>0.5]="Yes"

default_pred <- balance_pred

#Apend this column to balance_df
balance_df2 <- balance_df %>%
  mutate(default = default_pred)

#Take random sample of 500 from the default_test data frame to match 'length'
default_test_sample <- sample_n(default_test, 500)
```

```r
#Create confusion matrix
table(true = default_test_sample$default, predicted = default_pred)
```

```
##      predicted
## true   No Yes
##   No  300 180
##   Yes  13   7
```

It does not seem so like the logistic regression performs better since we get a very large number of false-positives. That is, we overestimate the number of people who would default on their debt (top-right cell) when we use a logistic regression model with threshhold >50%.

15. Train an LDA classifier lda_mod on the training set.

```r
lda_mod <- lda(default   student + balance + income, data=default_train)
```

16. Look at the lda_mod object. What can you conclude about the characteristics of the people who default on their loans?

```r
lda_mod
```

```
## Call:
## lda(default ~ student + balance + income, data = default_train)
##
## Prior probabilities of groups:
##    No   Yes
## 0.967 0.033
##
## Group means:
##       student   balance   income
## No  0.2922699  802.1897 33589.86
## Yes 0.3750000 1765.8840 32125.68
##
## Coefficients of linear discriminants:
##                   LD1
## student -2.383849e-01
## balance  2.246633e-03
## income   1.931398e-06
```

From the LDA classifier we observe that students have a much greater probability of defaulting and that the mean balance of someone who defaults is more than twice as much as the mean balance of someone who does not default. Interestingly, income makes little difference to individuals defaulting.

17. Create a confusion matrix and compare it to the previous methods.

```r
#Use test data set
lda_pred <- predict(lda_mod, default_test)
```

```r
names(lda_pred)
```

```
## [1] "class"     "posterior" "x"
```

```r
#Now use the 'class' feature to assess performance
lda_class <- lda_pred$class

table(true = default_test$default, predicted = lda_class)
```

```
##      predicted
## true    No  Yes
##   No  1926    5
##   Yes   58   11
```

```r
#Calculate the accuracy (diagonal entries) for this table
conf_mat <- table(true = default_test$default, predicted = lda_class)

lda_acc <- (sum(conf_mat[1,1] + conf_mat[2,2]) / sum(conf_mat))

paste(lda_acc*100, "%", sep="")
```

```
## [1] "96.85%"
```

We obtain a very large accuracy, at nearly 100%, for the LDA model and hence we can assume this to be the best classifier for our model.

18. Create a model (using knn, logistic regression, or LDA) to predict whether a 14 year old boy from the 3rd class would have survived the Titanic disaster. You can find the data in the data/ folder. Would the passenger have survived if they were a girl in 2nd class?

```r
##Use LDA as this is more stable than linear regression because the classes are well s
Titanic <- read.csv("data/Titanic.csv")

#Inspect first few ros of data frame
head(Titanic)
```

```
##                                            Name PClass   Age    Sex Survived
## 1                  Allen, Miss Elisabeth Walton    1st 29.00 female        1
## 2                   Allison, Miss Helen Loraine    1st  2.00 female        0
## 3           Allison, Mr Hudson Joshua Creighton    1st 30.00   male        0
## 4 Allison, Mrs Hudson JC (Bessie Waldo Daniels)    1st 25.00 female        0
## 5                 Allison, Master Hudson Trevor    1st  0.92   male        1
## 6                            Anderson, Mr Harry    1st 47.00   male        1
```

```r
#Inspect classes
lapply(Titanic, class)
```

```
## $Name
## [1] "factor"
```

```
##
## $PClass
## [1] "factor"
##
## $Age
## [1] "numeric"
##
## $Sex
## [1] "factor"
##
## $Survived
## [1] "integer"
```

```
#Probability of survival, used in calculation later when generating a 14-year-old male
survival.prob <- sum(Titanic$Survived) / nrow(Titanic)

##Change certain variable types
library(stringr) #Import stringr library

#Function to extract
numextract <- function(string){
  str_extract(string, "\\-*\\d+\\.*\\d*")
}

#Transform variables into integers and transform the response variable of 'Survived' i
PClass_dummy <- as.integer( numextract(Titanic$PClass) ) #Persons class
Sex_dummy <- as.integer( ifelse(Titanic$Sex == "female", 0, 1) )
Survived_dummy <- as.factor( ifelse(Titanic$Survived == 1, "Yes", "No") )

#Apend the newly created varibles to data frame
Titanic <- Titanic %>%
  mutate(PClass = PClass_dummy) %>%
  mutate(Sex = Sex_dummy) %>%
  mutate(Survived = Survived_dummy
  )

##Create training and test data sets for Titanic df
#Make sure the results are reproducible
set.seed(1027)

#Randomise the rows
nobs <- nrow(Titanic)   #Number of rows
idx_df <- 1:nobs        #Indices of rows
Titanic <- Titanic[sample(idx_df), ] #Randomize
```

```r
#Split the data into 80% train and 20% test data
train_idx <- seq(1, nobs*0.8) #Training data indices
test_idx <- seq((max(train_idx) + 1), nobs) #Test  data indices

Titanic_train <- Titanic[train_idx, ] #Training data
Titanic_test <- Titanic[test_idx, ] #Test data

#Train the model on only the training data, with 'Survived' as the predictor variable
Titanic.lda <- lda(Survived   PClass + Age + Sex, data=Titanic_train)

#Use test data set
Titanic.pred <- predict(Titanic.lda, newdata = Titanic_test, type="response")

names(Titanic.pred)
```

```
## [1] "class"     "posterior" "x"
```

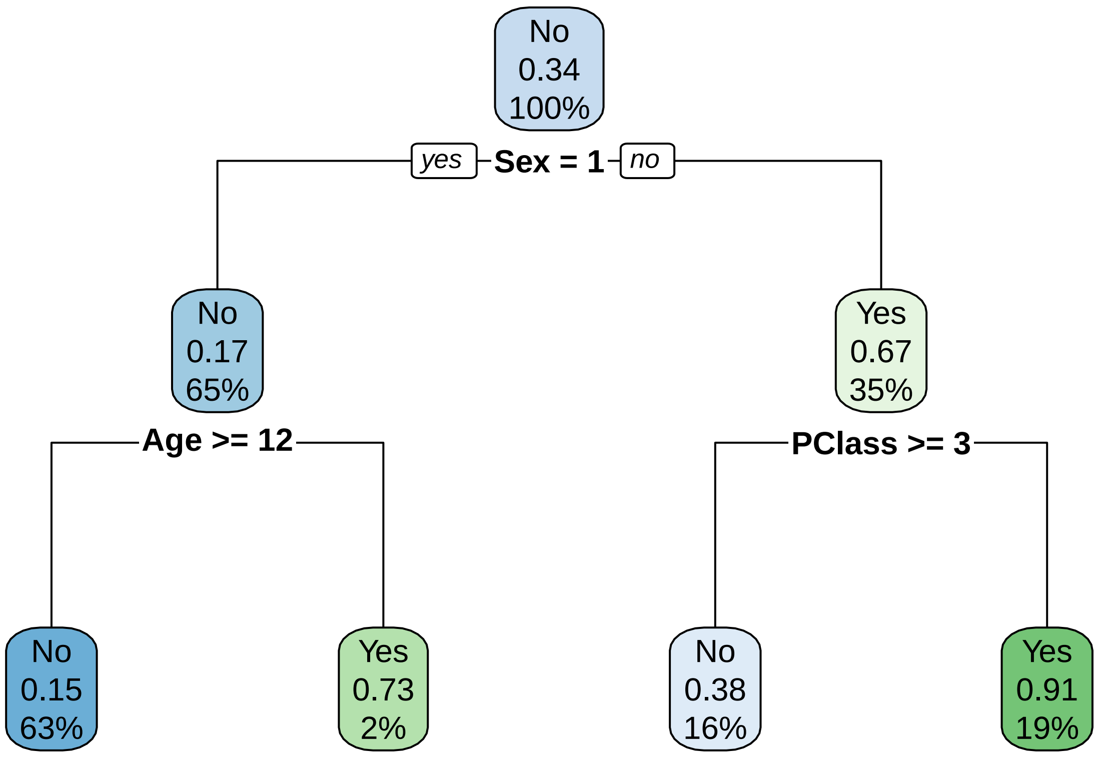
```r
#Now use the 'class' feature to assess performance
Titanic.class <- Titanic.pred$class

table(true = Titanic_test$Survived, predicted = Titanic.class)
```

```
##      predicted
## true  No Yes
##   No  70   9
##   Yes 24  50
```

```r
#Calculate the accuracy (diagonal entries) for this table
Titanic.matrix <- table(true = Titanic_test$Survived, predicted = Titanic.class)

Titanic.acc <- (sum(Titanic.matrix[1,1] + Titanic.matrix[2,2]) / sum(Titanic.matrix))

paste(format(Titanic.acc*100, digits=2, nsmall=2), "%", sep="")
```

```
## [1] "78.43%"
```

```r
##Now answer the question by creating a prediction tree
Titanic.tree <- rpart(Survived ~ PClass + Age + Sex, data = Titanic, control = list(cp

rpart.plot(Titanic.tree)
```



Survival probability of 14-year-old boy in 3rd class and a girl in 2nd class.

```
pred.table <- predict( Titanic.lda, data.frame( PClass=c(3,2), Age=c(14, na.rm=TRUE), Se

boy.survival <- pred.table$posterior[1,2]
girl.survival <- pred.table$posterior[2,2]

print(paste0("Survival probability of 14-year-old boy in 3rd class: ", paste(format(boy
```

```
## [1] "Survival probability of 14-year-old boy in 3rd class: 7.04%"
```

```
print(paste0("Survival probability of a girl (any age) in 2nd class: ", paste(format(gi
```

```
## [1] "Survival probability of a girl (any age) in 2nd class: 94.27%"
```