Supervised learning: Classification

Contents

Part 1: to be completed at home before the lab **1**

K-Nearest Neighbours **4**

Part 2: to be completed during the lab **7**

Assessing classification **7**

Logistic regression **10**

Final exercise **14**

Part 1: to be completed at home before the lab

In this lab at home, two different classification methods will be covered: K-nearest neighbours and logistic regression. You can download the student zip including all needed files for practical 5 here.

Note: the completed homework has to be **handed in** on Black Board and will be **graded** (pass/fail, counting towards your grade for individual assignment). The deadline is two hours before the start of your lab. Hand-in should be a **PDF** or **HTML** file. If you know how to knit pdf files, you can hand in the knitted pdf file. However, if you have not done this before, you are advised to knit to a html file as specified below, and within the html browser, 'print' your file as a pdf file.

One of the packages we are going to use is class. For this, you will probably need to `install.packages("class")` before running the `library()` functions. In addition, you will again need the `caret` package to create a training and a validation split for the used dataset (*note*: to keep this at home lab compact, we will only use a training and validation split, and omit the test dataset to evaluate model fit). You can download the student zip including all needed files for practical 5 here.

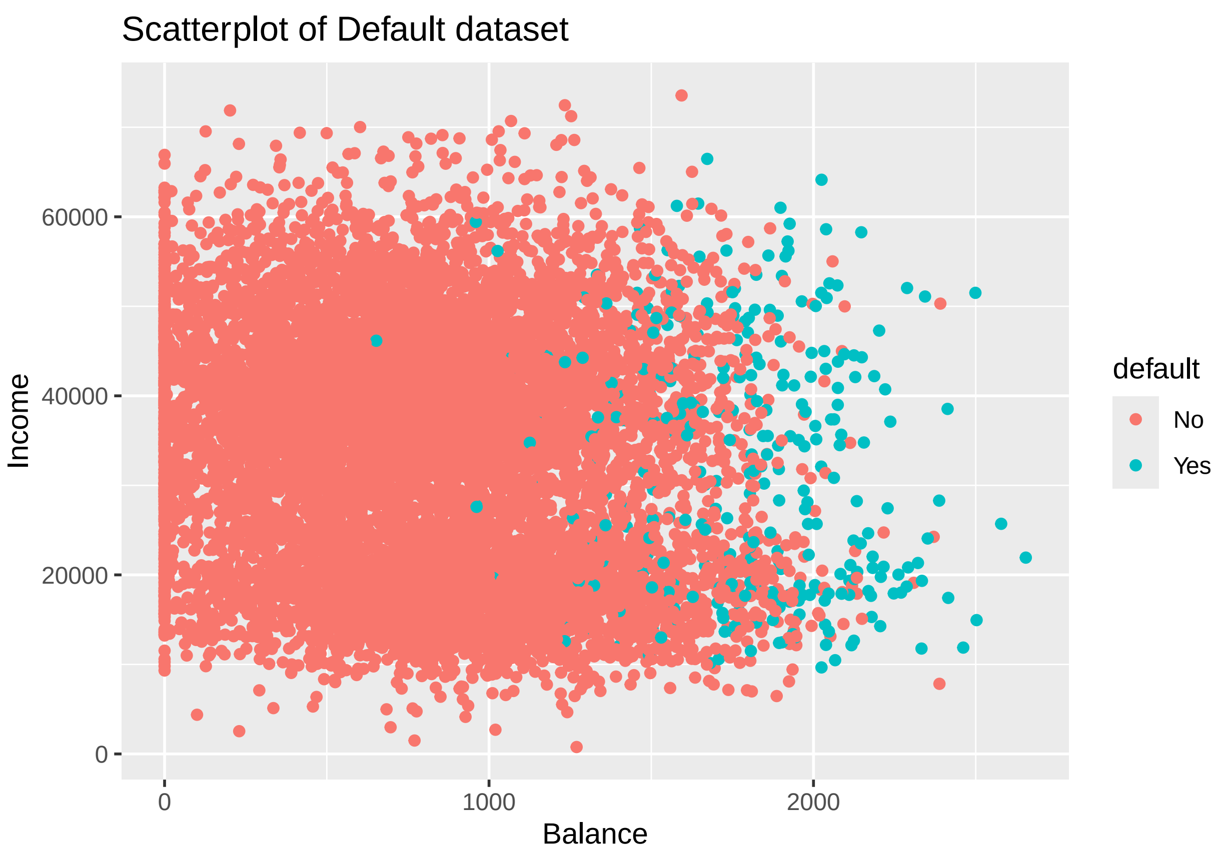
```r
library(MASS)
library(class)
library(caret)
library(ISLR)
library(tidyverse)
```

This practical will be mainly based around the `default` dataset which contains credit card loan data for 10 000 people. With the goal being to classify credit card cases as `yes` or `no` based on whether they will default on their loan.

---

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?**

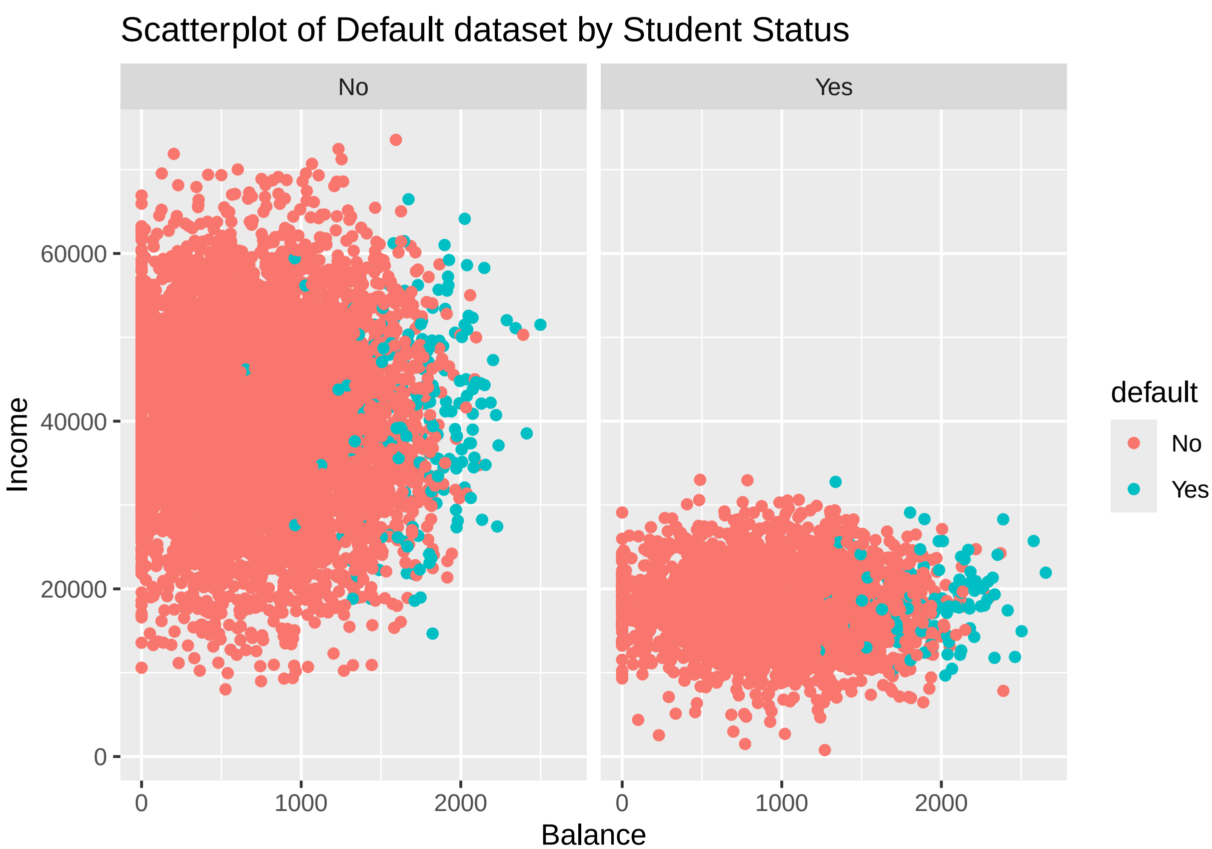
```r
ggplot(Default, aes(x = balance, y = income, color = default)) +
  geom_point() +
  labs(title = "Scatterplot of Default dataset", x = "Balance", y = "Income")
```



---

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

```
ggplot(Default, aes(x = balance, y = income, color = default)) +
  geom_point() +
  facet_grid(cols = vars(student)) +
  labs(title = "Scatterplot of Default dataset by Student Status", x = "Balance", y = "I
```



---

3. **For use in the KNN algorithm, 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 validation set `default_valid` (20%) using the `createDataPartition()` function of the `caret` package.**

If you haven't used the function `ifelse()` before, please feel free to review it in Chapter 5 Control Flow (*particular section 5.2.2*) in Hadley Wickham's Book Advanced R, this provides a concise overview of choice functions (`if()`) and vectorised if (`ifelse()`).

```r
Default <- Default %>%
  mutate(student_dummy = ifelse(student == "Yes", 1, 0))

set.seed(123)
trainIndex <- createDataPartition(Default$default, p = 0.8, list = FALSE)
default_train <- Default[trainIndex, ]
default_valid <- Default[-trainIndex, ]
```

K-Nearest Neighbours

Now that we have explored the dataset, we can start on the task of classification. We can imagine a credit card company wanting to predict whether a customer will default on the loan so they can take steps to prevent this from happening.

The first method we will be using is k-nearest neighbours (KNN). It classifies datapoints based on a majority vote of the k points closest to it. In `R`, the `class` package contains a `knn()` function to perform knn.

4. **Create class predictions on the default_valid data using the test parameter from 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`.**

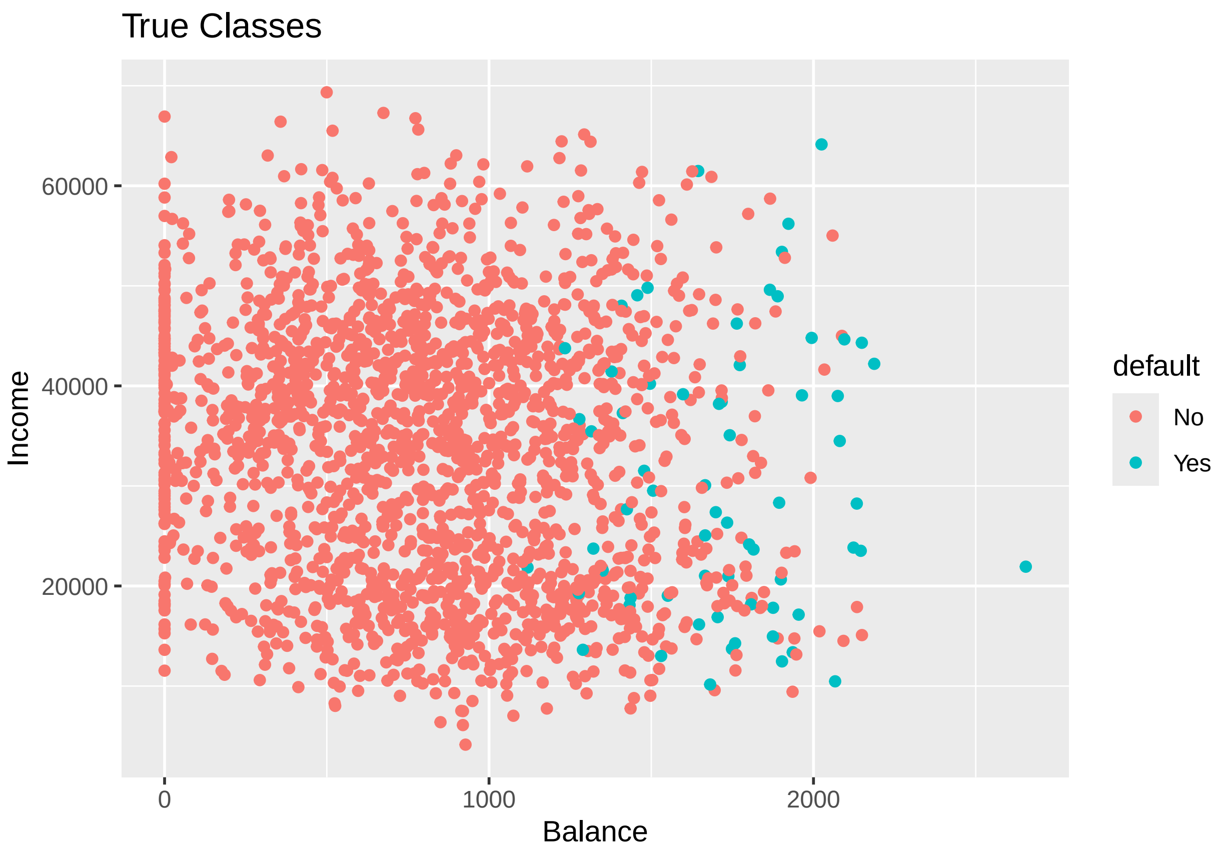
Remember: make sure to review the `knn()` function through the *help* panel on the GUI or through typing "?knn" into the console. For further guidance on the `knn()` function, please see *Section 4.7.6* in An introduction to Statistical Learning

```r
knn_5_pred <- knn(train = default_train[, c("student_dummy", "balance", "income")],
                  test = default_valid[, c("student_dummy", "balance", "income")],
                  cl = default_train$default, k = 5)
```

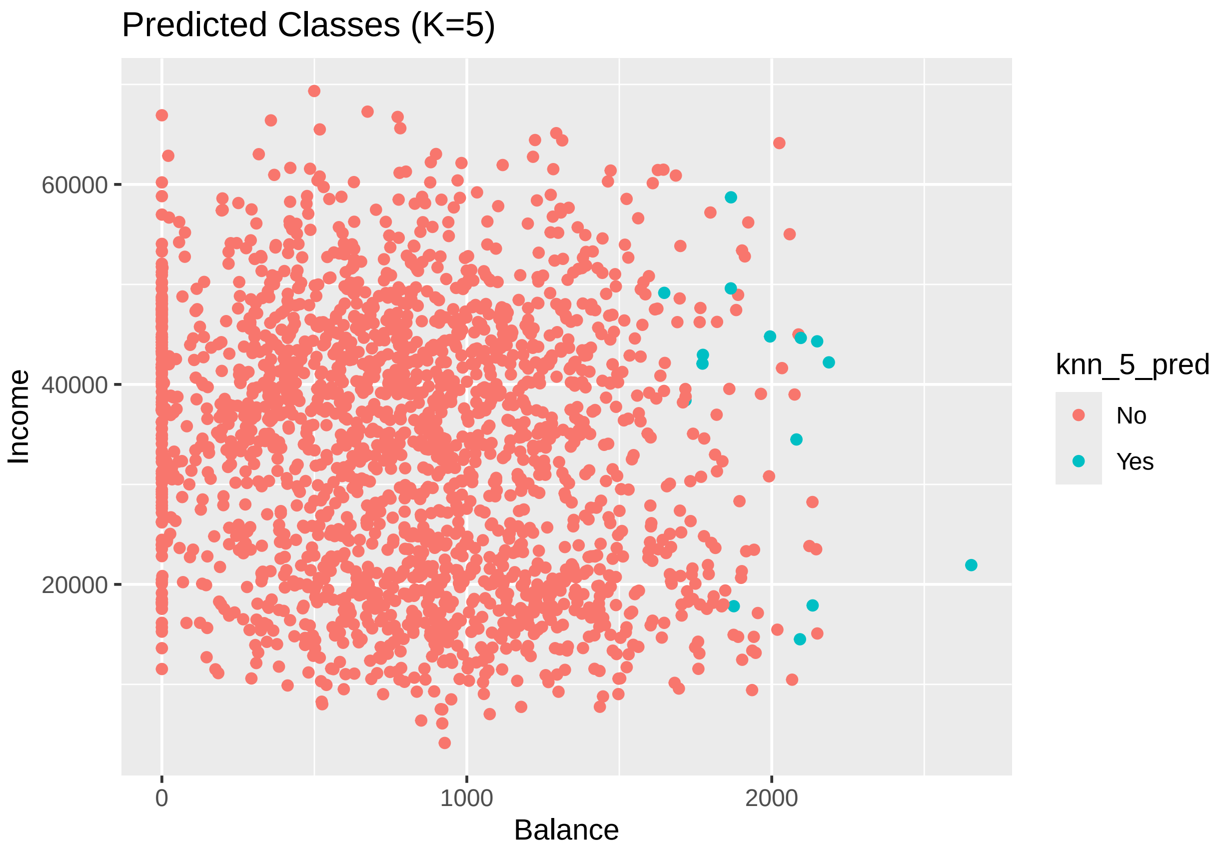
5. Create two scatter plots with income and balance as in the first plot you made but now using only the default_valid data. One with the true class (`default`) mapped to the colour aesthetic, and one with the predicted class (`knn_5_pred`) mapped to the colour aesthetic. Hint: Add the predicted class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

```r
default_valid <- default_valid %>%
  mutate(knn_5_pred = knn_5_pred)

ggplot(default_valid, aes(x = balance, y = income, color = default)) +
  geom_point() +
  labs(title = "True Classes", x = "Balance", y = "Income")
```




```r
ggplot(default_valid, aes(x = balance, y = income, color = knn_5_pred)) +
  geom_point() +
  labs(title = "Predicted Classes (K=5)", x = "Balance", y = "Income")
```



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

```
knn_2_pred <- knn(train = default_train[, c("student_dummy", "balance", "income")],
                  test = default_valid[, c("student_dummy", "balance", "income")],
                  cl = default_train$default, k = 2)

default_valid <- default_valid %>%
  mutate(knn_2_pred = knn_2_pred)

ggplot(default_valid, aes(x = balance, y = income, color = knn_2_pred)) +
  geom_point() +
  labs(title = "Predicted Classes (K=2)", x = "Balance", y = "Income")
```

During this we have manually tested two different values for K, this although useful in exploring your data. To know the optimal value for K, you should use cross validation.

Part 2: to be completed during the lab

Assessing classification

The confusion matrix is an insightful summary of the plots we have made and the correct and incorrect classifications therein. A confusion matrix can be made in R with the `table()` function by entering two `factors`:

```
conf_2NN <- table(predicted = knn_2_pred, true = default_valid$default)
conf_2NN
```

To learn more these, please see *Section 4.4.3* in An Introduction to Statistical Learning, where it discusses Confusion Matrices in the context of another classification method Linear Discriminant Analysis (LDA).

---

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

```r
perfect_conf_matrix <- table(predicted = default_valid$default, true = default_valid$def
perfect_conf_matrix
```

```
##          true
## predicted   No  Yes
##       No  1933    0
##       Yes    0   66
```

---

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

```r
knn_5_pred <- knn(train = default_train[, c("student_dummy", "balance", "income")],
                  test = default_valid[, c("student_dummy", "balance", "income")],
                  cl = default_train$default, k = 5)

conf_2NN <- table(predicted = knn_2_pred, true = default_valid$default)
conf_5NN <- table(predicted = knn_5_pred, true = default_valid$default)

conf_2NN
```

```
##          true
## predicted   No  Yes
##       No  1891   44
##       Yes   42   22
```

```r
conf_5NN
```

```
##          true
## predicted   No  Yes
##       No  1928   56
##       Yes    5   10
```

9. **Comparing performance becomes easier when obtaining more specific measures. Calculate the specificity, sensitivity, accuracy and the precision of the 2nn and 5nn model, and compare them. Which model would you choose? Keep the goal of our prediction in mind when answering this question.**

```
specificity_2NN <- conf_2NN[1,1] / sum(conf_2NN[,1])
sensitivity_2NN <- conf_2NN[2,2] / sum(conf_2NN[,2])
accuracy_2NN <- sum(diag(conf_2NN)) / sum(conf_2NN)
precision_2NN <- conf_2NN[2,2] / sum(conf_2NN[2,])

# 5-NN model
specificity_5NN <- conf_5NN[1,1] / sum(conf_5NN[,1])
sensitivity_5NN <- conf_5NN[2,2] / sum(conf_5NN[,2])
accuracy_5NN <- sum(diag(conf_5NN)) / sum(conf_5NN)
precision_5NN <- conf_5NN[2,2] / sum(conf_5NN[2,])

specificity_2NN
```

```
## [1] 0.9782721
```

```
sensitivity_2NN
```

```
## [1] 0.3333333
```

```
accuracy_2NN
```

```
## [1] 0.9569785
```

```
precision_2NN
```

```
## [1] 0.34375
```

```
specificity_5NN
```

```
## [1] 0.9974133
```

```r
sensitivity_5NN
```

```
## [1] 0.1515152
```

```r
accuracy_5NN
```

```
## [1] 0.9694847
```

```r
precision_5NN
```

```
## [1] 0.6666667
```

Logistic regression

KNN directly predicts the class of a new observation using a majority vote of the existing observations closest to it. In contrast to this, logistic regression predicts the `log-odds` of belonging to category 1. These log-odds can then be transformed to probabilities by performing an inverse logit transform:

$p = \frac{1}{1+e^{-\alpha}}$

where α; indicates log-odds for being in class 1 and p is the probability.

Therefore, logistic regression is a `probabilistic` classifier as opposed to a `direct` classifier such as KNN: indirectly, it outputs a probability which can then be used in conjunction with a cutoff (usually 0.5) to classify new observations.

Logistic regression in `R` happens with the `glm()` function, which stands for generalized linear model. Here we have to indicate that the residuals are modeled not as a Gaussian (normal distribution), but as a `binomial` distribution.

---

10. **Use `glm()` with argument `family = binomial` to fit a logistic regression model `lr_mod` to the `default_train` data. Use student, income and balance as predictors.**

```r
lr_mod <- glm(default ~ student + income + balance, data = default_train, family = binom
```

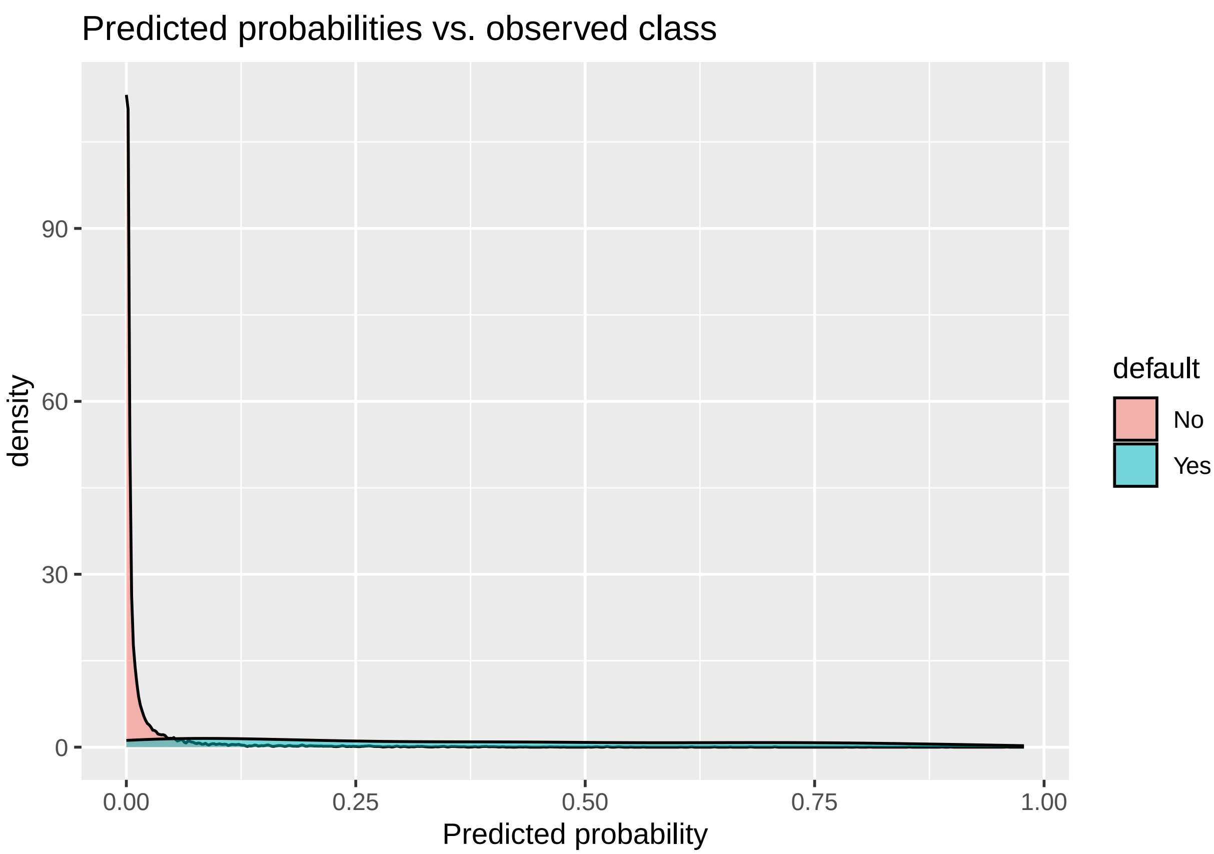
Now we have generated a model, we can use the `predict()` method to output the estimated probabilities for each point in the training dataset. By default `predict` outputs the log-odds, but we can transform it back using the inverse logit function of before or setting the argument `type = "response"` within the predict function.

---

11. **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.**

```
train_prob <- predict(lr_mod, type = "response")

ggplot(data.frame(prob = train_prob, default = default_train$default), aes(x = prob, fil
  geom_density(alpha = 0.5) +
  labs(title = "Predicted probabilities vs. observed class", x = "Predicted probability"
```



Another advantage of logistic regression is that we get coefficients we can interpret.

---

12. **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?**

```
summary(lr_mod)$coefficients
```

```
##                  Estimate   Std. Error    z value      Pr(>|z|)
## (Intercept) -1.129089e+01 5.666206e-01 -19.926719  2.387052e-88
## studentYes  -5.467190e-01 2.679263e-01  -2.040557  4.129484e-02
## income       1.213662e-05 9.273698e-06   1.308714  1.906313e-01
## balance      5.788308e-03 2.618596e-04  22.104624 2.852980e-108
```

```
predict(lr_mod, newdata = data.frame(student = "No", income = 40000, balance = 3000), ty
```

```
##         1
## 0.9985854
```

Let's visualise the effect `balance` has on the predicted default probability.

---

13. **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.**

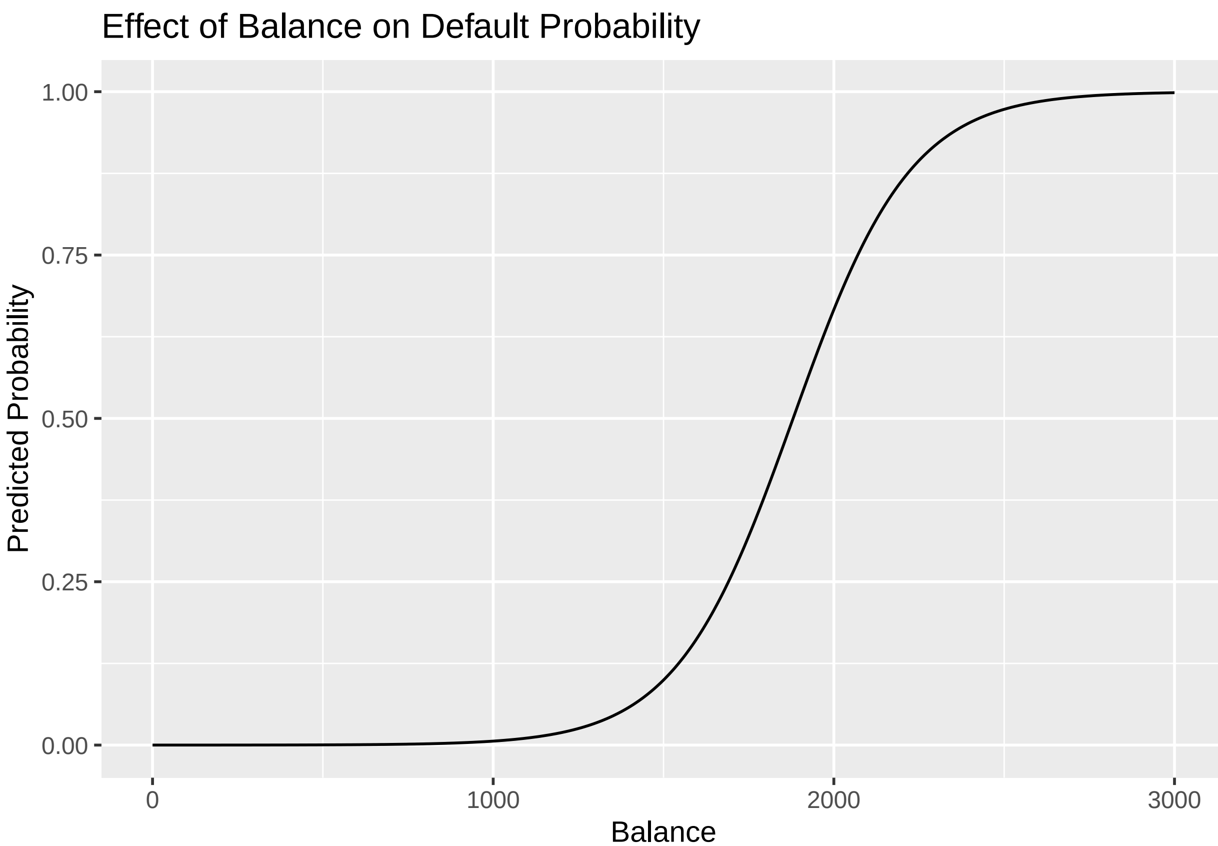
```
mean_income <- mean(default_train$income)
balance_df <- data.frame(student = factor(rep("No", 500), levels = c("No", "Yes")),
                         balance = seq(0, 3000, length.out = 500),
                         income = mean_income)
```

---

14. **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
balance_df$predicted_prob <- predict(lr_mod, newdata = balance_df, type = "response")

ggplot(balance_df, aes(x = balance, y = predicted_prob)) +
  geom_line() +
  labs(title = "Effect of Balance on Default Probability", x = "Balance", y = "Predicted
```



---

15. **Use lr_mod to predict the probability of defaulting for the observations in the validation dataset. Use these to 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?**

```r
valid_prob <- predict(lr_mod, newdata = default_valid, type = "response")
lr_pred <- ifelse(valid_prob > 0.5, "Yes", "No")
conf_lr <- table(predicted = lr_pred, true = default_valid$default)
conf_lr
```

```
##          true
## predicted   No  Yes
##       No  1921   49
##       Yes   12   17
```

---

16. **Calculate the specificity, sensitivity, accuracy and the precision for the logistic regression using the above confusion matrix. Again, compare the logistic regression to KNN.**

```r
specificity_lr <- conf_lr[1,1] / sum(conf_lr[,1])
sensitivity_lr <- conf_lr[2,2] / sum(conf_lr[,2])
accuracy_lr <- sum(diag(conf_lr)) / sum(conf_lr)
precision_lr <- conf_lr[2,2] / sum(conf_lr[2,])

specificity_lr
```

```
## [1] 0.993792
```

```r
sensitivity_lr
```

```
## [1] 0.2575758
```

```r
accuracy_lr
```

```
## [1] 0.9694847
```

```r
precision_lr
```

```
## [1] 0.5862069
```

Final exercise

Now let's do another - slightly less guided - round of KNN and/or logistic regression on a new dataset in order to predict the outcome for a specific case. We will use the Titanic dataset also discussed in the lecture. The data can be found in the `/data` folder of your project. Before creating a model, explore the data, for example by using `summary()`.

---

17. Create a model (using knn or logistic regression) to predict whether a 14 year old boy from the 3rd class would have survived the Titanic disaster.

```
titanic_file_path <- "titanic.csv"  # Ensure the path is correct relative to your worki
if (!file.exists(titanic_file_path)) {
  stop("The Titanic dataset file does not exist at the specified path.")
}

titanic_data <- read.csv(titanic_file_path)

# Check if titanic_data is a data frame
if (!is.data.frame(titanic_data)) {
  stop("The Titanic dataset is not loaded as a data frame.")
}

# Print structure of titanic_data to debug
str(titanic_data)
```

```
## 'data.frame':    1313 obs. of  5 variables:
##  $ Name    : chr  "Allen, Miss Elisabeth Walton" "Allison, Miss Helen Loraine" "Allis
##  $ PClass  : chr  "1st" "1st" "1st" "1st" ...
##  $ Age     : num  29 2 30 25 0.92 47 63 39 58 71 ...
##  $ Sex     : chr  "female" "female" "male" "female" ...
##  $ Survived: int  1 0 0 0 1 1 1 0 1 0 ...
```

---

18. **Would the passenger have survived if they were a 14 year old girl in 2nd class?**

```
titanic_data$Survived <- as.factor(titanic_data$Survived)
titanic_data$PClass <- as.factor(titanic_data$PClass)
titanic_data$Sex <- as.factor(titanic_data$Sex)
```