Logistic Regression

Contents

1 Credit card data

In this workshop, we will consider the credit card default dataset used in book Introduction to Statistical Learning.

```
# Load the data
Default <- read.csv("Default.csv", stringsAsFactors = TRUE)
# Recall main properties
str(Default)
```

```
## 'data.frame':    10000 obs. of  4 variables:
##  $ defaulted: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ student  : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2 1 1 ...
##  $ balance  : num  730 817 1074 529 786 ...
##  $ income   : num  44362 12106 31767 35704 38463 ...
```

The objective of the individuals that gathered this data was to predict individuals that default on their credit card debt. This information is contained in the variable "defaulted" (e.g., "Did this individual default on their debt? Yes or No"). For reasons that will become apparent, I will change the encoding of the class variable "defaulted" from a factor with levels "No", "Yes" to a numerical variable with values 1 and 0. I will do this by re-defining this variable:

```
# Count occurrences of each class
table(Default$defaulted)
```

```
##
##   No  Yes
```

```
## 9667  333
```

```
# Verify current levels
levels(Default$defaulted)
```

```
## [1] "No"  "Yes"
```

```
# Encode "No" as 0 and "Yes" as 1
# If you don't understand check each step separately
Default$defaulted <- 1*(Default$defaulted=="Yes")
# Attach data.frame
attach(Default)
```

Re-defining variables is something you will definitely need to do in any sort of data analyst role! Please make sure you feel comfortable with how to do this.

2 Linear regression

Let's first consider the relationship between "defaulted" and credit card balance ("balance") (which is the most informative variable in this dataset as we have seen in previous workshops).

The first model we will fit (another word for estimate) is a linear regression model. In linear regression the response (also called independent variable) Y is continuous, which is why we encoded "defaulted" as a numerical variable. Recall that a linear regression model estimates the **expected** value of the response Y, given the value of the predictor (explanatory variable), X. In mathematical notation, a linear regression estimates,

$$\mathbb{E}(Y|X = x) = \beta_0 + \beta_1 x$$

Our class variable "defaulted" takes only two values, 0 and 1. A random variable Y that takes only two values, $\{0, 1\}$, is called a Bernoulli random variable. The expected value of a Bernoulli random variable is given by,

$$\mathtt{E}(Y) = 1 \cdot P(Y = 1) + 0 \cdot P(Y = 0) = P(Y = 1)$$

(the first equality above is just the formula for the expected value.) In other words the expected value of Y is equal to the probability of $Y = 1$. If we want to estimate the expected value of Y conditional on another (explanatory) variable X then we rewrite the above but this time using conditional expectations/ probabilities:

$$\mathtt{E}(Y|X = x) = 1 \cdot P(Y = 1|X = x) + 0 \cdot P(Y = 0|X = x) = P(Y = 1|X = x)$$

Let's combine the above with the definition of linear regression. For a Bernoulli random variable Y,

$$\mathbb{E}(Y|X = x) = P(Y = 1|X = x) = \beta_0 + \beta_1 x$$

In words, if we use a linear regression model to predict a Bernoulli random variable Y ("defaulted" in this case), the linear model estimates the probability of $Y = 1$ (in our case "defaulted"=1) as a linear equation of the predictor(s).

Having encoded the variable "defaulted" as 1 if a customer defaulted and 0 if the customer did not, the linear regression model estimates the probability of a customer defaulting.

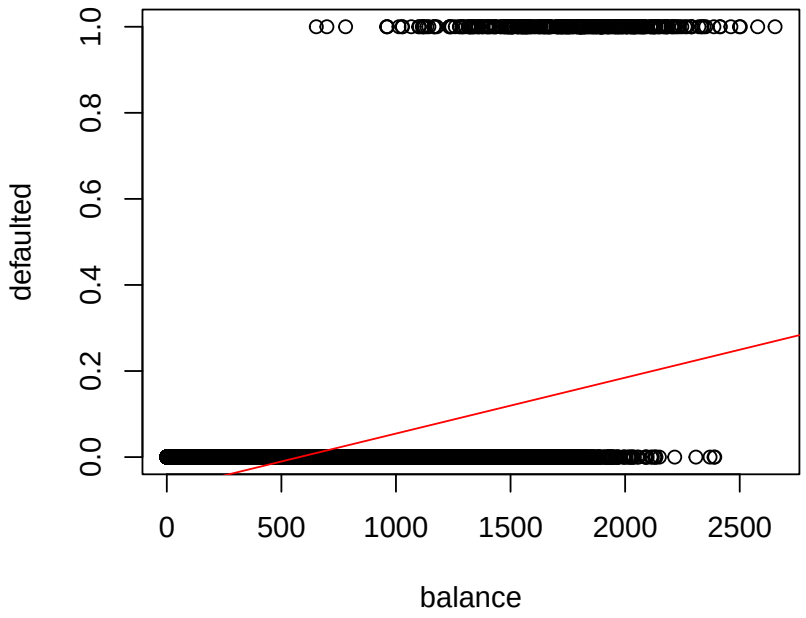
3 Estimation of linear regression in R

To estimate a linear model use the function `lm()` (from `linear model`). This function requires as input:

1. A formula in the form,
$$Y \sim X_1 + X_2 + \cdots + X_d$$
where the Y is the target (dependent) variable, and X_i are the features (independent variable).
2. A data.frame that contains these variables. The data.frame can be omitted if it has been attached as we did in prior tutorials.

Let's visualize the two variables, and the relationship posited by the linear model.

```
# Estimate linear regression
linear.model <- lm(defaulted ~ balance)
# Visualise data
plot(balance, defaulted)
# Visualise fitted linear regresion
abline(linear.model, col="red")
```



Using the summary command we can obtain information about the linear model we estimated. Most importantly, we can find out the values of the coefficients,

```
summary(linear.model)
```

```
##
## Call:
## lm(formula = defaulted ~ balance)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.23533 -0.06939 -0.02628  0.02004  0.99046
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -7.519e-02  3.354e-03  -22.42   <2e-16 ***
## balance      1.299e-04  3.475e-06   37.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1681 on 9998 degrees of freedom
## Multiple R-squared:  0.1226, Adjusted R-squared:  0.1225
## F-statistic:  1397 on 1 and 9998 DF,  p-value: < 2.2e-16
```

The interpretation of the coefficients of a linear model is well known:

1. $\hat{\beta}_0$: The estimated value of Y when $x = 0$. In our case this interpretation doesn't make sense. The expected value of Y when balance=0 in this case is the probability $P(Y = 1|X = 0)$, which according to the above model is negative.
2. $\hat{\beta}_1$: How much the expected value of Y will change if "balance" increases by one unit (in this case the units are US dollars). Therefore the probability $P(Y = 1|X = x)$ will change by a constant amount 1.2987218×10^{-4} if x increases by one.

As we see from the last figure, the linear regression model allows the probability of default to be negative. Although not shown in the figure, if "balance" is large enough this probability will also exceed one. This is clearly not satisfactory. We next fit a logistic regression model that does not have these problems.

4 Logistic regression model

A logistic regression model assumes that the response Y for a fixed value of the predictors, X, is a Binomially distributed random variable. This also includes categorical variables. Let's explain what this means using the current dataset as an example:

If in our dataset we have n customers (observations) that have the same value of the predictor variable "balance", then:

1. Each of these customers has the same probability of defaulting. We denote this probability as $P(\texttt{defaulted} = \texttt{Yes}|\texttt{balance} = x)$ or more succinctly as $P(Y = 1|X = x)$
2. Whether each customer defaults or not is **independent** of what the other $(n - 1)$ customers did

If the above two assumptions hold then the number of customers that default follows a Binomial distribution. In this distribution, the probability that k out of n customers default is equal to:

$$P(K = k) = \binom{n}{k} P(Y = 1|X = x)^k P(Y = 0|X = x)^{n-k}, \quad k = 0, 1, \ldots, n$$

note that k can be any integer between 0 and n.

If there is only **one** customer with a particular value for the balance variable, then $n = 1$ and the Binomial distribution is the same with the Bernoulli distribution:

$$P(K = k) = P(Y = 1|X = x)^k P(Y = 0|X = x)^{1-k}, \quad k = 0, 1$$

where k can be only 0 or 1.

The logistic regression model is also using a linear combination of the predictors, but the linear equation does **not** estimate $P(Y = 1|X = x)$ but rather the log-odds:

$$\log\left(\frac{P(Y = 1|X = x)}{1 - P(Y = 1|X = x)}\right) = \beta_0 + \beta_1 x$$

This means that the actual probability we are interested in is given by:

$$P(Y = 1|X = x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}} = \frac{1}{1 + e^{-\beta_0-\beta_1 x}}$$

5 Estimating Logistic Regression in R

Logistic regression is a type of linear regression that uses a different distribution. More specifically, it is a member of a broad class of models known as Generalized Linear Models. Very briefly these are regression models in which the dependent variable Y is not normally distributed conditionally on X but follows another distribution, such as: Binomial (logistic regression model); Multinomial (multinomial logistic regression); Poisson (Y refers to counts) etc.

For this reason the function that fits logistic regression models in `R` is called `glm()` (from `g`eneralized `l`inear `m`odel). This function requires:

1. A formula as in the case of linear regression
2. `data =` this argument specifies the data.frame that contains the variables in the formula
3. `family =` this argument specifies the distribution of Y conditional on X. In our case this distribution is binomial

Let's give an example of using this function:

```
# Fit logistic regression model
log.model <- glm(defaulted ~ balance, data = Default, family = binomial)
# Summary of estimated logistic regression model
summary(log.model)
```

```
##
## Call:
## glm(formula = defaulted ~ balance, family = binomial, data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2697  -0.1465  -0.0589  -0.0221   3.7589
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
## balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
## ---
## 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: 1596.5  on 9998  degrees of freedom
## AIC: 1600.5
##
## Number of Fisher Scoring iterations: 8
```

From the lectures recall the interpretation of the coefficients of the logistic regression model.

1. The intercept $\hat{\beta}_0$ is interpreted as:

$$P(Y = 1|X = 0) = P(\texttt{defaulted} = \texttt{Yes}|\texttt{balance} = 0) = \frac{1}{1 + e^{-\hat{\beta}_0}}$$

   or equivalently,

$$o(Y = 1|X = 0) = o(\texttt{defaulted} = \texttt{Yes}|\texttt{balance} = 0) = e^{\hat{\beta}_0}$$

2. The slope coefficient $\hat{\beta}_1$ is interpreted as the change in the log-odds, $\log(o(Y = 1|X = x))$, when X (credit card balance) increases by one unit (in this case 1 dollar).

Equivalently the change in the odds of default $o(Y = 1|X = x)$ when credit card balance increases by one is $e^{\hat{\beta}_1}$

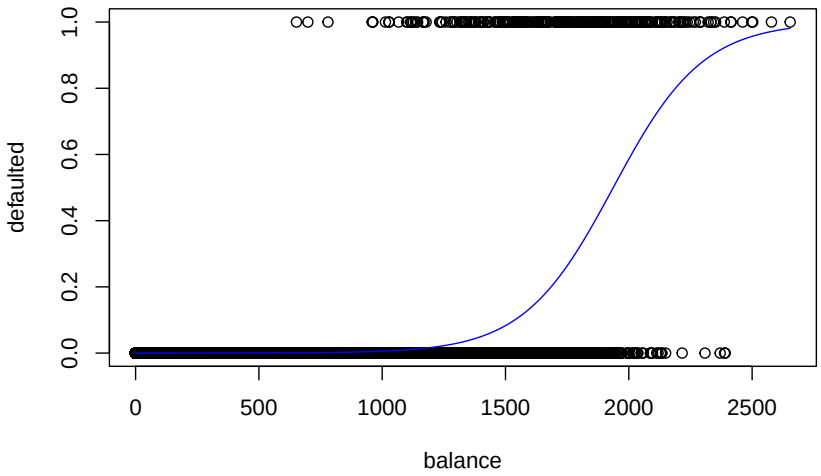
Note that the interpretations of the slope is very different in the logistic regression model from that in the linear model. **Note** also that although the log-odds and the odds change by a constant amount as "balance" increases by one unit, the same is **not** true for the probability of default. This can also be seen by plotting the estimated probability of default for the different values of "balance". Recall that in logistic regression:

$$P(\texttt{defaulted} = 1|\texttt{balance} = x) = \frac{1}{1 + \exp(-\beta_0 - \beta_1 x)}$$

```
# First plot the observations (as points)
plot(balance, defaulted)
# To plot the estimated probability first define a vector x that
# takes values in the range of the "balance" variable
x <- seq(from = min(balance), to=max(balance))
# obtain estimated coefficients using the coef() function
hat.beta <- coef(log.model)
hat.beta
```

```
##   (Intercept)       balance
## -10.651330614   0.005498917
```

```
# Estimate probabilities from Log. Regr. formula
lines(x, (1 + exp(-hat.beta[1] - hat.beta[2]*x))^(-1), col="blue")
```



The above figure shows that the estimated probability of default increases as "balance" increases. It becomes very close to one for the customers with the very high balance, and very close to zero for customers with zero credit card balance. This is much more sensible than the output of the linear regression model.

6 Obtaining predictions

Next we want to obtain predictions from the estimated logistic regression model, and use these to assess how well we can predict the behavior of different customers.

The generic function, `predict()` produces predictions from the results of a number of fitting functions including `lm()` and `glm()`. The three most important arguments for this function are: `predict(object, newdata, type)`

- `object` is the output of the fitting function. In our case it can be linear model object `linear.model` or the logistic regression object `log.model`
- `newdata` is an optional argument that allows user to specify a different dataset for which predictions will be made. If it is unspecified, predictions for the data.frame used to fit/estimate the model are produced
- `type` specifies the type of prediction that the user requires. For logistic regression the default is `type="link"` in which case the function returns the linear part of the model (i.e. $\beta_0 + \beta_1$`balance`). If you want to obtain the estimated probabilities you need to set `type="response"`. (Remember that response is a synonym for dependent variable)

```
# Compute the estimated probability of default for all the data in Default
probs <- predict(log.model, newdata = Default, type="response")
```

We have discussed in class numerous times that a classifier is a function that takes as input a description of an object (in the present case the "object" is an individual and the description is their credit card "balance") and outputs an estimated probability that the object belongs to a class (in our case the class is "defaulted", and the classifier estimates $P($`defaulted` $=$ `Yes`$|$`balance` $= x)$). The last figure illustrating the estimated probabilities clearly demonstrates this points.

To **decide** to which class to assign (classify) a customer, defaulted="Yes"/"No", we compare the estimated probability with a **threshold**. All classification methods operate in this way. Also note that the chosen threshold does not depend on the classifier, but rather on the characteristics of the application.

In the following we will choose the threshold of 0.5, but beware that this is an arbitrary choice. Based on these classifications we estimate the **misclassification error**, which is effectively the proportion of times the estimated class is incorrect.

```
# Predict class 1 (i.e. default=Yes) if estimated probability > 0.5
class.pred <- 1*(probs > 0.5)
# Verify predictions
head(cbind(probs,class.pred), 10)
```

```
##           probs class.pred
## 1  1.305680e-03          0
## 2  2.112595e-03          0
## 3  8.594741e-03          0
## 4  4.344368e-04          0
## 5  1.776957e-03          0
## 6  3.704153e-03          0
## 7  2.211431e-03          0
## 8  2.016174e-03          0
## 9  1.383298e-02          0
## 10 2.366877e-05          0
```

7 Truth table & misclassification error

We can assess how many times the predictions are right and how many times they are correct using a **truth table**, also known as a **confusion matrix**. Such a table can be constructed with the `table()` function:

```
# Create truth table: Rows represents actual class, Column represents predicted
truth.table <- table(defaulted, class.pred)
truth.table
```

```
##           class.pred
## defaulted    0    1
##         0 9625   42
##         1  233  100
```

Recall how the `table()` function operates: the first argument determines the row while the second determines the column. Thus

- truth.table[1,1] = 9625 is the number of good customers (`defaulted=0`) which are predicted to be good customers (`class.pred=0`)
- truth.table[1,2]= 42 is number of good customers (`defaulted=0`) which are predicted to be bad customers (`class.pred=1`)
- truth.table[2,1]= 233 is number of bad customers (`defaulted=1`) which are predicted to be good customers (`class.pred=0`)
- truth.table[2,2]= 100 is number of bad customers (`defaulted=1`) which are predicted to be bad customers (`class.pred=1`)

An error is made when the actual and predicted class are different. Errors therefore correspond to the off-diagonal elements of the truth table. The misclassification error (also called the error rate) is the proportion of times the classifier incorrectly predicted the class.

```
# Total number of observations in truth.table
N <- sum(truth.table)
# Misclassification error
(truth.table[1,2]+ truth.table[2,1])/N
```

```
## [1] 0.0275
```

```
# Accuracy = Proportion of correct predictions
(truth.table[1,1]+ truth.table[2,2])/N
```

```
## [1] 0.9725
```

Accuracy is simply the proportion of times the classifier was correct. By definition: Accuracy = 1 - Misclassification error

Food for thought This classifier has an estimated error rate of 0.0275. Are you satisfied with this? Is this a good model that you would trust?

Note that there is nothing special about the 0.5 threshold. If the loss from a bad customer is higher than the gain from a good customer then the bank should use a much lower threshold. In other words it should give loans (credit cards) to people that have a probability of default much lower than 0.5. Let's repeat the above exercise but now using a threshold of 0.1.

Exercise Replicate the above code to obtain the truth.table using 0.1 as the threshold for classifying a customer to be bad, $\hat{Y} = 1$. Estimate the misclassification rate and the accuracy and compare these to the previous values. Which of the two thresholds would you recommend using and why?

8 Interpretation of threshold

It is very important to understand that classifying customers based on any threshold, e.g. `probs>0.5`, defines a threshold with respect to the value of the predictor variable "balance". Let's explain this in more detail:

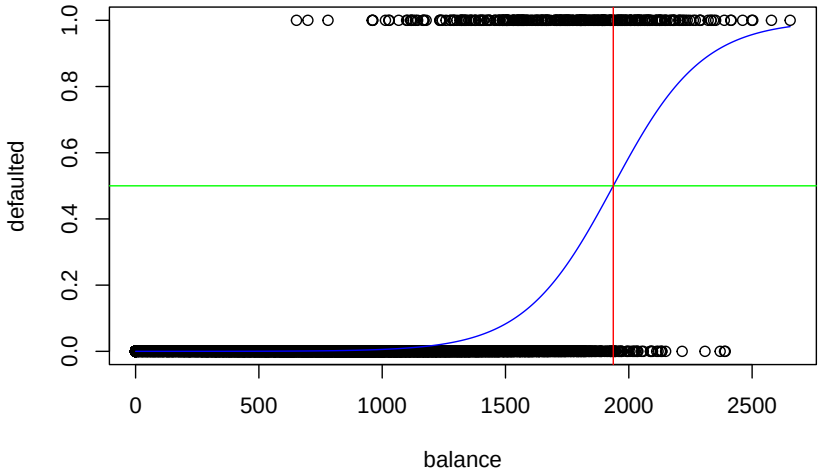
$P(\texttt{defaulted} = 1|\texttt{balance} = x) = \frac{1}{1+\exp(10.65-0.0055x)} > \frac{1}{2} \Rightarrow$

$1 + \exp(10.65 - 0.0055x) < 2 \Rightarrow$

$10.65 - 0.0055x < \ln(1) \Rightarrow x > \frac{10.65}{0.0055} = 1936.3636364\$$

Therefore, customers with credit card "balance" greater than 1936.3636364 are categorized as customers that will default, $\hat{Y} = 1$, and any customers with lower credit card balance are categorized as customers who will not default, $\hat{Y} = 0$.

```
# First plot the observations (as points)
plot(balance, defaulted)
# Plot estimated probabilities
lines(x, (1 + exp(-hat.beta[1] - hat.beta[2]*x))^(-1), col="blue")
# Plot classification threshold
# (straight "h"orizontal line)
abline(h=0.5, col="green")
# Plot partition of input space
# (straight "v"ertical line)
abline(v=-hat.beta[1]/hat.beta[2], col="red")
```



In the above figure the classification threshold of 0.5 is illustrated with a green horizontal line ($Y = 0.5$). The red line indicates the partition of the input space (in this case there is only one input: "balance"). Customers with credit card balance smaller or equal to 1936.3636364 (i.e. all customers to the left of the red line) are classified as good risk ($\hat{Y} = 0$). Customers with credit card balance larger than this value (i.e. to the right of the red line) are classified as bad risk ($\hat{Y} = 1$).

Exercise Repeat the above but this time require a much stricter threshold. Classify as bad risk any customer with probability of default greater than 0.2. Estimate the misclassification error and compare the two classifiers

9 Multiple logistic regression

Nothing changes in terms of R code when estimating a logistic regression model with more than one variables. Things do change however, when it comes to the interpretation of the model. We will cover this with the simplest possible example involving two variables, "balance" and "income":

```
# Estimate logistic regression with 2 predictors
log.model2 <- glm(defaulted ~ balance + income, data=Default, family = binomial)
# model summary
summary(log.model2)
```

```
##
## Call:
## glm(formula = defaulted ~ balance + income, family = binomial,
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## ---
## 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: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

Note that the interpretation now of the slope coefficients, $\hat{\beta}_1$ ($\hat{\beta}_2$) is: how much the log-odds of default change when balance (income) changes by one unit, **assuming all the other predictors are constant**. The interpretation of coefficients is valid only to the extent to which this condition is met! In terms of our example if balance is strongly correlated with income then this interpretation of $\hat{\beta}_1$ is not correct because changing balance implies a change in income.

To obtain the estimated probabilities of default we use the `predict()` function:

```
# Estimate probabilities of default
probs2 <- predict(log.model2, data=Default, type="response")
```

To produce classifications we compare the estimated probabilities with a threshold. I will once more use the (arbitrary) threshold of 0.5.

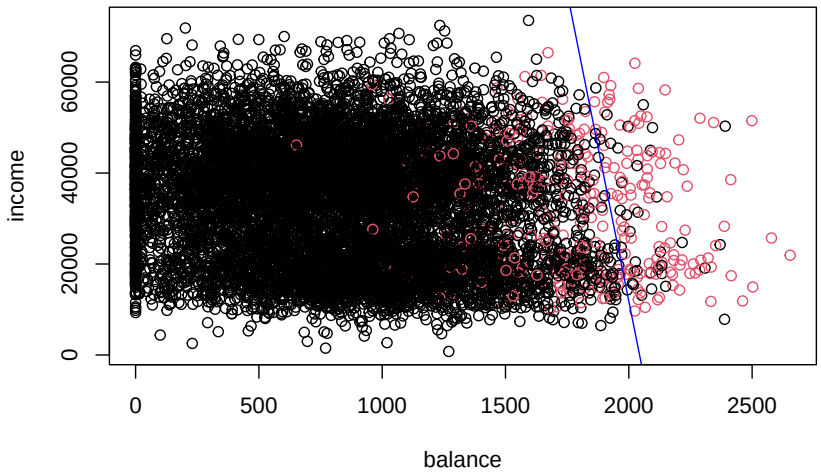
$P(\texttt{defaulted} = 1|\texttt{balance} = x_1, \texttt{income} = x_2) = \frac{1}{1+\exp(11.54-0.0056x_1-0.00002x_2)} > \frac{1}{2} \Rightarrow$

$1 + \exp\left(11.54 - 0.0056x_1 - 0.00002x_2\right) < 2 \Rightarrow$

$11.54 - 0.0056x_1 - 0.00002x_2 < \ln(1) = 0 \Rightarrow x_2 > \frac{11.54}{0.00002} - \frac{0.0056}{0.00002}x_1$

Note that the last inequality defines a straight line in the input space (i.e. in the space of balance, income). Let's visualise this:

```
# Scatterplot of income against balance using default to colour data
plot(balance, income, col = as.factor(defaulted))
# Get estimated coefficients
hb <- coef(log.model2)
# Define straight line using intercept and slope
abline(a=-hb[1]/hb[3], b=-hb[2]/hb[3], col="blue")
```



10 Decision boundary

The blue line in the above figure defines the **decision boundary**. All the points on the blue line (i.e. all the combinations of balance and income) the estimated probability of default, $P(Y = 1|X_1 = x_1, X_2 = x_2)$, is exactly equal to 0.5. Points above and to the right of the blue line have higher estimated probability of default higher than 0.5; points below and to the left have estimated probability of default smaller than 0.5.

What we just did illustrates the very important fact that **logistic regression imposes a linear decision boundary in the input space**. This boundary is determined by solving for the pairs of (X_1, X_2) that make the estimated probability exactly equal to the threshold.

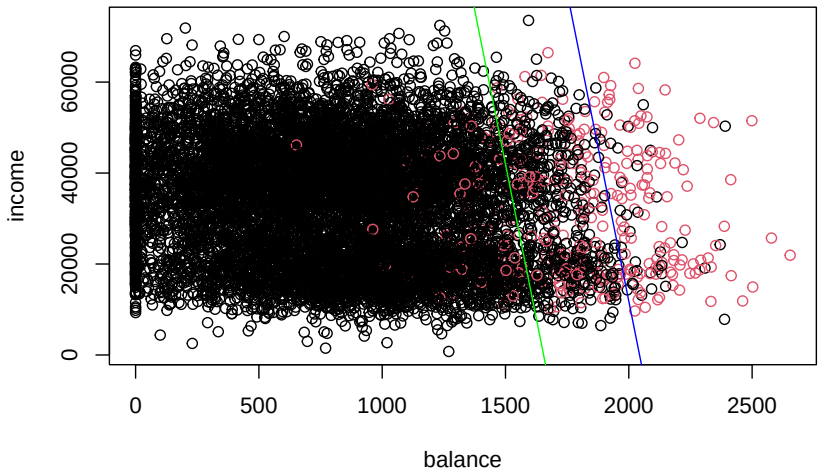
If we use a different threshold an alternative threshold for classification e.g. 0.1, or 0.7 this will have as effect to shift this line vertically up or down. In other words in the above equation only the **intercept** will change. I will illustrate this for the 0.1 threshold. Try to replicate the analysis for the other case:

$P(\texttt{default} = 1|\texttt{balance} = x_1, \texttt{income} = x_2) = \frac{1}{1+\exp(11.54-0.0056x_1-0.00002x_2)} > \frac{1}{10} \Rightarrow$

$1 + \exp\left(11.54 - 0.0056x_1 - 0.00002x_2\right) < 10 \Rightarrow$

$11.54 - 0.0056x_1 - 0.00002x_2 < \ln(9) \Rightarrow x_2 > \frac{11.54-\ln(9)}{0.00002} - \frac{0.0056}{0.00002}x_1$

```
# Scatterplot of income against balance using default to colour data
plot(balance, income, col = as.factor(defaulted))
# Get estimated coefficients
hb <- coef(log.model2)
# Decision boundary for threshold = 0.5
abline(a=-hb[1]/hb[3], b=-hb[2]/hb[3], col="blue")
# Decision boundary for threshold = 0.1
abline(a=(-hb[1]-log(9))/hb[3], b=-hb[2]/hb[3], col="green")
```



Note that

1. The blue and green lines are parallel. This occurs because only the intercept of the decision boundary has changed and not the slope.
2. The observations (customers) that lie between the blue and green lines are classified as good risk when the threshold is 0.5, and as bad risk when the threshold is 0.1. As expected, decreasing the threshold causes some of the observations that were previously predicted to be in class 0 (good risk) to now be in class 1 (bad risk). The opposite would occur if the threshold was increased.

Observe that being more conservative about our willingness to classify customers as good risk allows us to correctly classify many more customers that have defaulted (this is shown by the number of red points on the scatterplot that lie between the green and blue lines, but you should be able to measure this exactly by constructing the truth table). On the other hand, it also decreases the number of good risk customers that were refused a credit card.

This trade-off exists in all classification problems and we will discuss this more in future lectures and workshops. For now it is important to understand that it exists, and why.