Supervised Learning

Classification

In the (supervised) classification problem, we have a response Y and predictors. In this case the response is categorical.

Example: suppose the outcome is:

$Y =$ loan defaults (1) or no default (0)

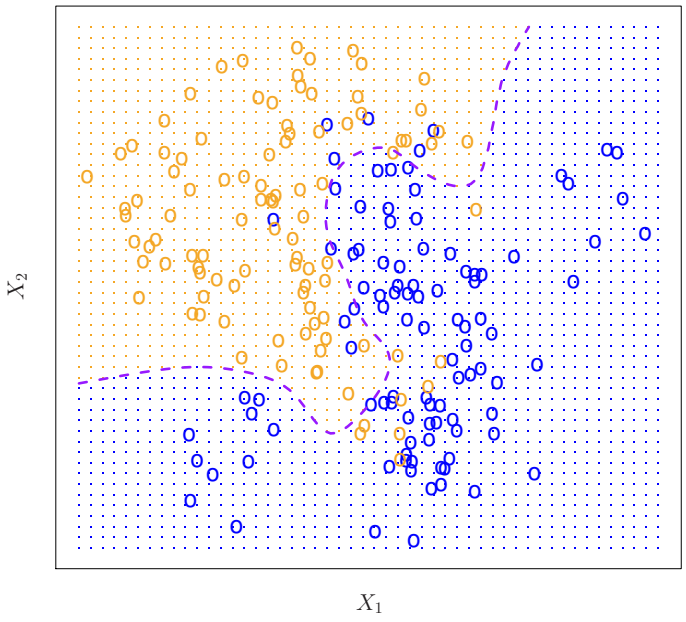
Inputs are:

$X_1 =$ credit card balance and

$X_2 =$ income

The goal is to estimate a function f, and use it to predict outcome $\hat{Y}$.

For two predictors and a two-class response, the result might look like



The blue and orange colours represent the classes of the observed responses.

The dashed line shows the decision boundary between the two classes.

In this example, the response is binary, but it could have more than 2 levels also.

Why not use linear regression

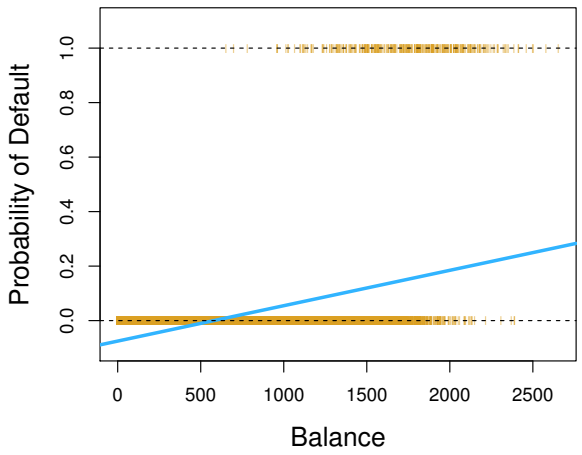
If you use linear regression with a binary response the result will be a fitted value that is an estimate of

$$P(Y = 1|X)$$

We could then predict default if $\hat{Y} > .5$ and no default otherwise.

The problem with this approach is

- it does not extend to the case where the response had more than 2 categories
- It can give fitted values (ie probability estimates) that are not between 0 and 1



This figure shows a linear regression fit to Y and X_1.

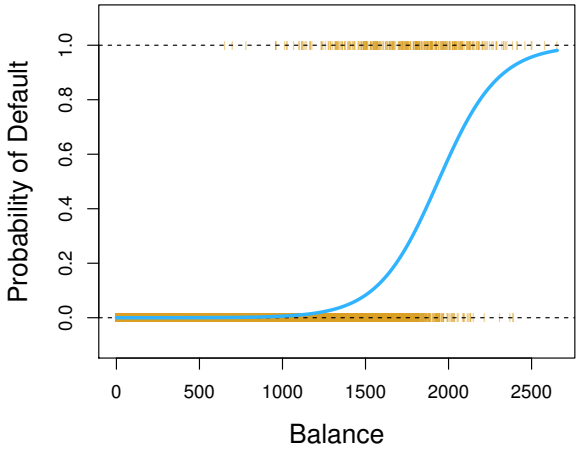
You can see that for low balances, the models predicts a negative probability of default.

The probability of default is below .5 for all balances, but a conservative bank might chose to predict default if $\hat{Y} > .1$ and no default otherwise.

Logistic regression

Logistic regression is for regression situations with a categorical response, of 2 or more levels.

Logistic regression always gives probabilities between 0 and 1.
Here is the result of logistic regression for predicting defaults:



Logistic regression always fits an S shaped curve to the data.

The probability of default is below .5 for balances of below 2000.

The logistic model says

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

This model can be fit to the data using the technique of maximum likelihood, which gives estimates for β_0 and β_1

```
library(ISLR) # for the data
glm(default ~ balance, family = "binomial", data = Default)
```

```
Call:  glm(formula = default ~ balance, family = "binomial", data = Default)

Coefficients:
(Intercept)      balance
   -10.6513       0.0055

Degrees of Freedom: 9999 Total (i.e. Null);  9998 Residual
Null Deviance:     2920
Residual Deviance: 1600  AIC: 1600
```

You can see from this that $\hat{\beta}_0 = -10.65$ and $\hat{\beta}_1 = .0055$

To interpret the parameters in logistic regression, we can see (after some manipulation) that the odds of success (here default) is

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

and that the log odds (logit) is

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

where $p = P(Y=1|X=x)$.

- in linear regression, a 1 unit increase in x changes mean Y by β_1.
- in logistic regression, a 1 unit increase in x changes log odds by β_1.
- Equivalently, in logistic regression, a 1 unit increase in x multiplies the odds by e^{β_1}.

To predict the default probability if the balance is 1000, we get

$$\hat{P}(Y=1|X=1000) = \frac{e^{-10.65+.0055\times 1000}}{1+e^{-10.65+.0055\times 1000}} = .00576$$

To do this in R

```
fit1 <- glm(default ~ balance, family = "binomial", data = Default)
predict(fit1, data.frame(balance = 1000), type = "response")
```

```
     1
0.0058
```

If you omit `type="response"`, you will get predictions for the logit.

Just like in linear regression, the fitted model can be summarised:

```
summary(fit1)
```

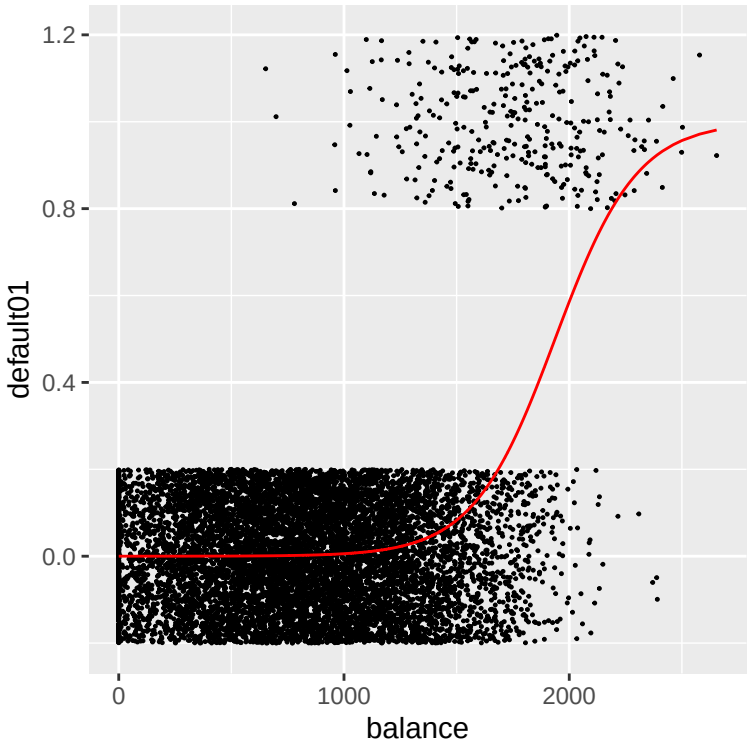
```

Call:
glm(formula = default ~ balance, family = "binomial", data = Default)
```

```
Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.270  -0.146  -0.059  -0.022   3.759

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -10.65133    0.36116   -29.5   <2e-16 ***
balance       0.00550    0.00022    24.9   <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

Number of Fisher Scoring iterations: 8
```

- This table shows coefficient (estimates), standard errors, Z statistics, and p-values.
- The use of the Normal distribution for p-values is approximate, but it is not appropriate to use the t distribution.
- In this example, balance has a very large Z-statistic and we would reject $H_0 : \beta_1 = 0$ in favour of $H_a : \beta_1 \neq 0$.

We can plot the fit using

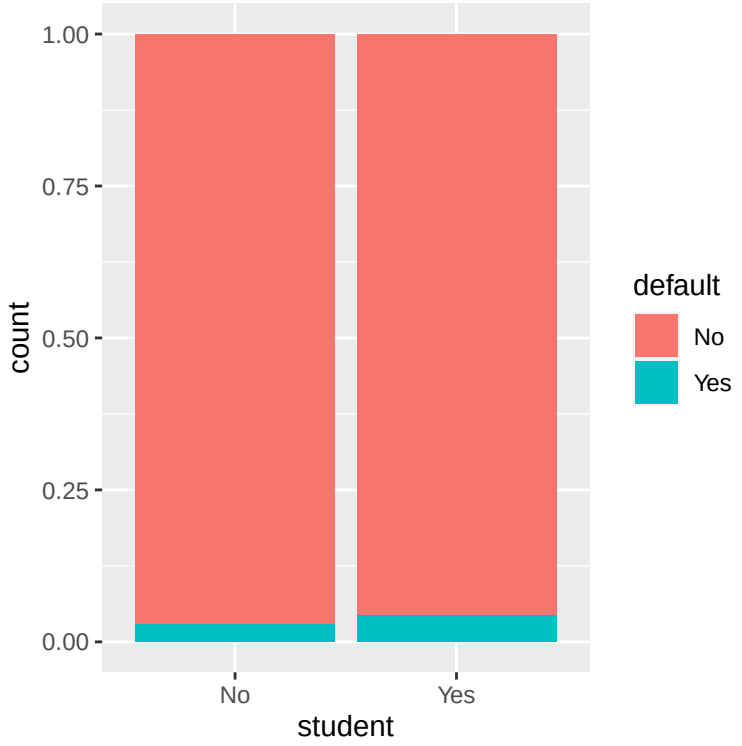
```
library(ggplot2)
Default$default01 <- as.numeric(Default$default) - 1
pred <- predict(fit1, Default, type="response")

ggplot(aes(x = balance, y = default01), data = Default) +
  geom_jitter(width = 0, height = 0.2, size = .2) +
  geom_line(aes(y = pred), color = "red")
```



Logistic regression with a binary predictor

The predictor is student status.

```
Default %>%
  ggplot(aes(x = student, fill = default)) +
    geom_bar(position = "fill")
```



This plot shows that the probability of default for students is a little higher than for non students.

```
fit2 <- glm(default ~ student, family = "binomial", data = Default)
summary(fit2)
```

```

Call:
glm(formula = default ~ student, family = "binomial", data = Default)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-0.297  -0.297  -0.243  -0.243   2.659

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.5041     0.0707  -49.55  < 2e-16 ***
studentYes    0.4049     0.1150    3.52  0.00043 ***
---
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: 2908.7  on 9998  degrees of freedom
AIC: 2913

Number of Fisher Scoring iterations: 6
```

From the above summary we can see that for students the log odds of default is .4 higher than for non students.

The odds of default for students are $e^{.4} = 1.49$ times that for non students.

```
predict(fit2, data.frame(student = c("Yes","No")), type = "response")
```

```
    1     2
0.043 0.029
```

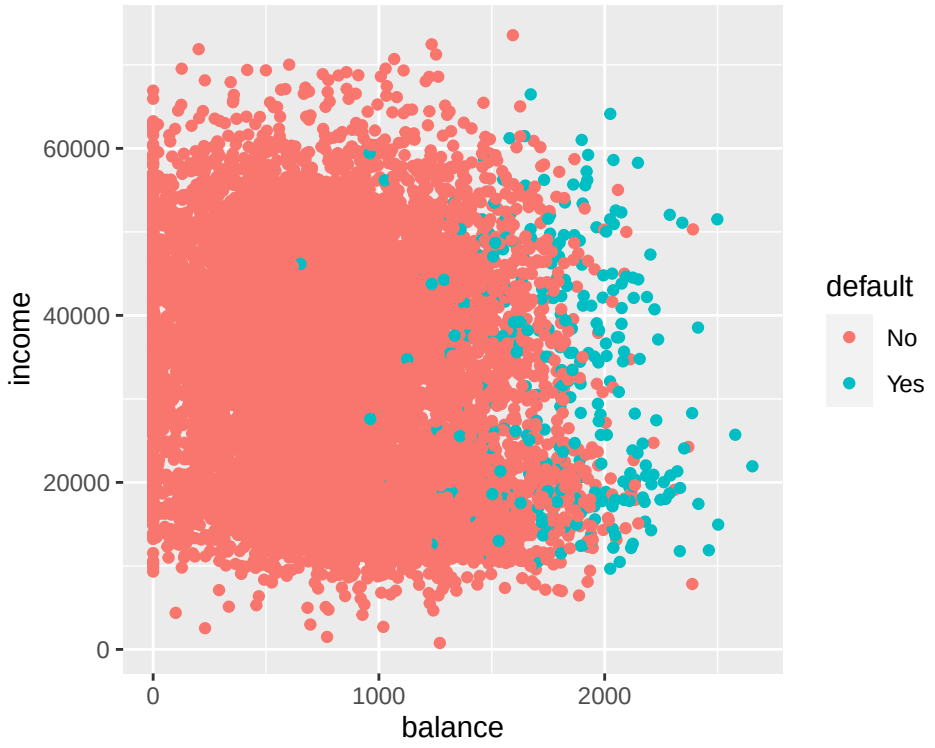
From this calculation we see the probability of default is .043 for students and .029 for non-students.

Logistic regression with multiple quantitative predictors

This time we use balance and income to predict the probability of default.

First we plot the data

```
Default %>%
  ggplot(aes(x = balance, y = income, color = default)) +
  geom_point()
```




```
fit3 <- glm(default ~ balance + income,
            family = "binomial",
            data = Default)
summary(fit3)
```

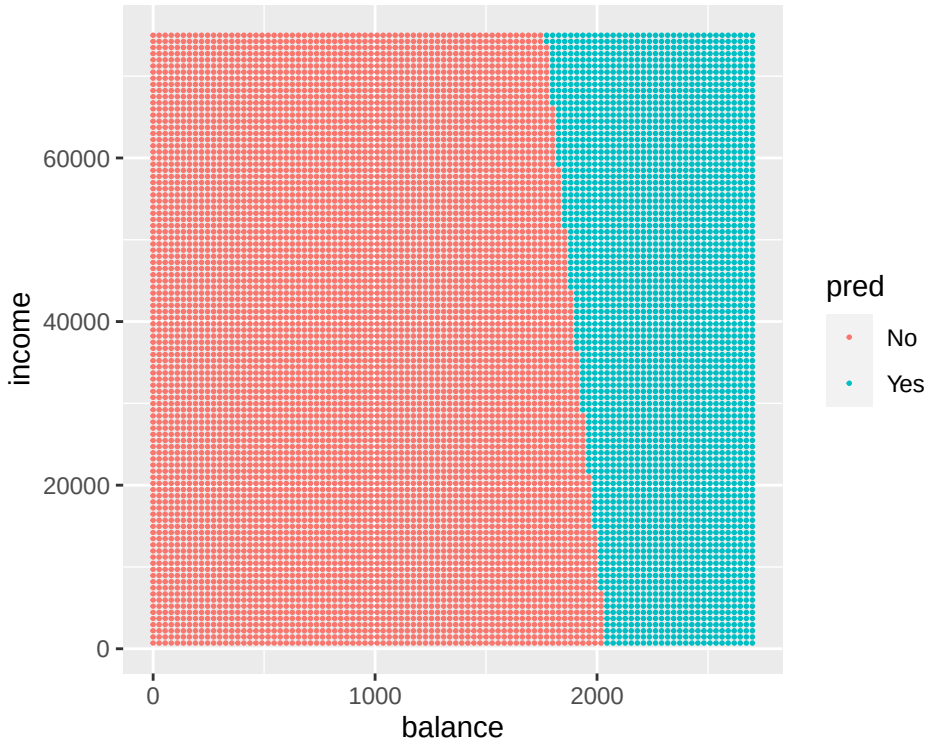
```

Call:
glm(formula = default ~ balance + income, family = "binomial",
    data = Default)
```

```
Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.473  -0.144  -0.057  -0.021   3.724

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.15e+01   4.35e-01  -26.54   <2e-16 ***
balance      5.65e-03   2.27e-04   24.84   <2e-16 ***
income       2.08e-05   4.99e-06    4.17    3e-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
```

This fit says that the higher balance and income, the higher the probabiliy of default.

The next plot shows, for which values of balance and income, the probability of default exceeds .5.

```
grid <- expand.grid(
  balance = seq(0, 2700, length = 100),
  income = seq(700, 75000, length = 100)
)
grid$prob <- predict(fit3, grid, type="response")
grid$pred <- factor(ifelse(grid$prob < .5, "No", "Yes"))
ggplot(aes(x=balance,y=income, color=pred), data=grid)+
  geom_point(size=.3)
```



Next we will we add student as a predictor:

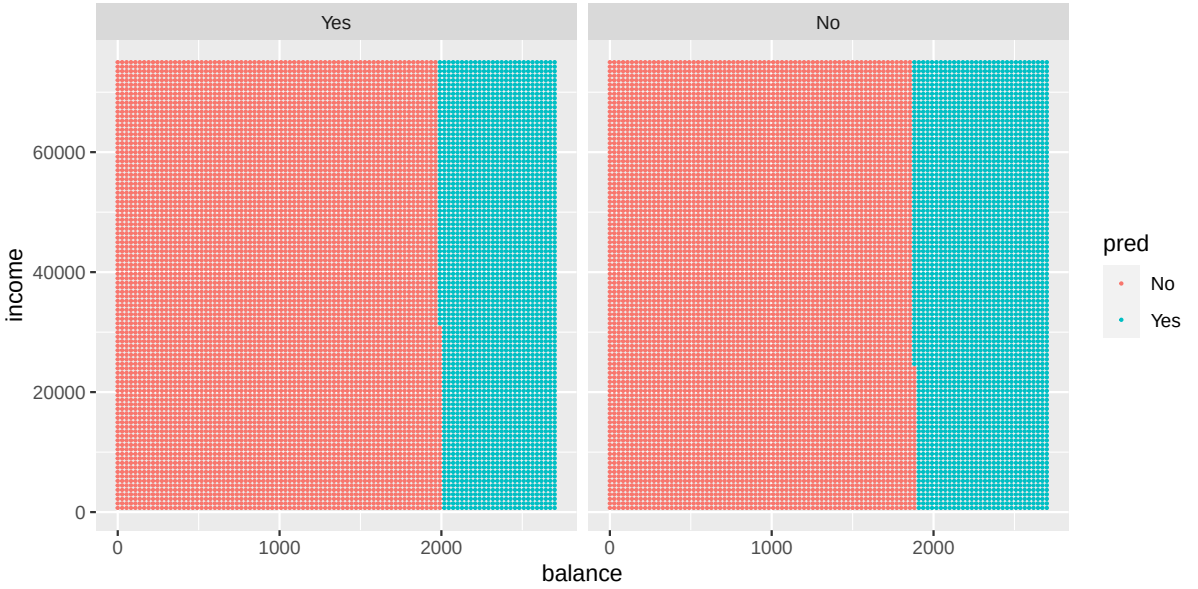
```
fit4 <- glm(default ~ balance + income + student,
            family = "binomial",
            data = Default)
summary(fit4)


Call:
glm(formula = default ~ balance + income + student, family = "binomial",
    data = Default)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.469  -0.142  -0.056  -0.020   3.738

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.09e+01   4.92e-01  -22.08   <2e-16 ***
balance      5.74e-03   2.32e-04   24.74   <2e-16 ***
income       3.03e-06   8.20e-06    0.37   0.7115
studentYes  -6.47e-01   2.36e-01   -2.74   0.0062 **
```

```
---
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: 1580

Number of Fisher Scoring iterations: 8
```

The coeficient of student is now negative, saying that for fixed balance and income, students are less likely to default than for non students.

The reason for this is that Student status is associated with balance, as students tend to have a lower balance.

```
grid <- expand.grid(
  balance = seq(0, 2700, length = 100),
  income = seq(700, 75000, length = 100),
  student=c("Yes", "No")
)

grid$prob <- predict(fit4, grid, type="response")
grid$pred <- factor(ifelse(grid$prob < .5, "No", "Yes"))

grid %>%
  ggplot(aes(x = balance, y = income, color = pred)) +
  geom_point(size = .3) +
  facet_wrap( ~ student)
```



A student with an income of 40,000 and balance of 2,000 is not predicted to default, whereas a non-student with that income and balance is predicted to default.

Confusion matrix

For any model fit previously we can compare the observed classes with the predicted classes.

```
prob <- predict(fit4, type = "response")
pred <- factor(ifelse(prob < .5, "No", "Yes"))

table(Default$default, pred)
```

```
     pred
        No  Yes
  No  9627   40
  Yes  228  105
```

- You can see from this that the model incorrectly classified $40 + 228 = 268$ loans, which is a mis-classification rate of 2.68 %

- Also, we can see that $228/(105+228) = 68.47$ % is the percentage of true defaulters that are misclassified

- And 40/(9627+40) = 0.41% is the percentage of non-defaulters that are misclassified.

Confusion matrix - terminology

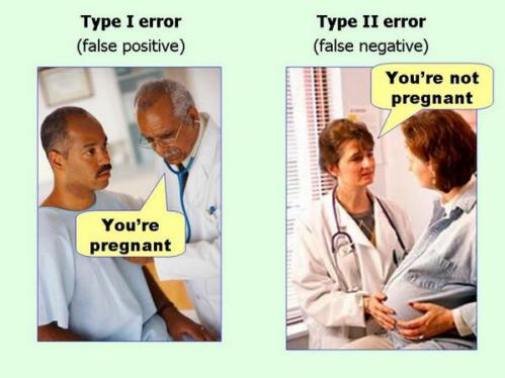
When there are two classes, designate one outcome as positive $y = +$ and the other as negative $y = -$.

y denotes the true class and $\hat{y}$ the predicted class.

Then the confusion matrix is of this form:

	$\hat{y} = +$	$\hat{y} = -$	Total
$y = +$	TP	FN	P
$y = -$	FP	TN	N

- The true negative rate is TNR = TN/N. This is known as the **specificity**.
- The false positive rate is FPR = FP/N. This is the **type I error**, 1- **specificity**.
- The true positive rate is TPR = TP/P. This is also known as the **sensitivity**.
- The false negative rate is FNR = FN/P is the **type II error**, 1- **sensitivity**.
- The positive predictive value is TP/(TP +FP). This is known as the **precision**.
- The negative predictive value is TN/(TN +FN).
- **Accuracy** ACC = (TP +TN)/(P+N).



Accuracy/misclassification rate: beware of unbalanced classes!

Change threshold

If we changed the classification threshold from .5 to .2, say, we will get a different result.

```
pred <- factor(ifelse(prob < .2, "No", "Yes"))
table(Default$default, pred)
```

```
     pred
        No  Yes
  No  9390  277
  Yes  130  203
```

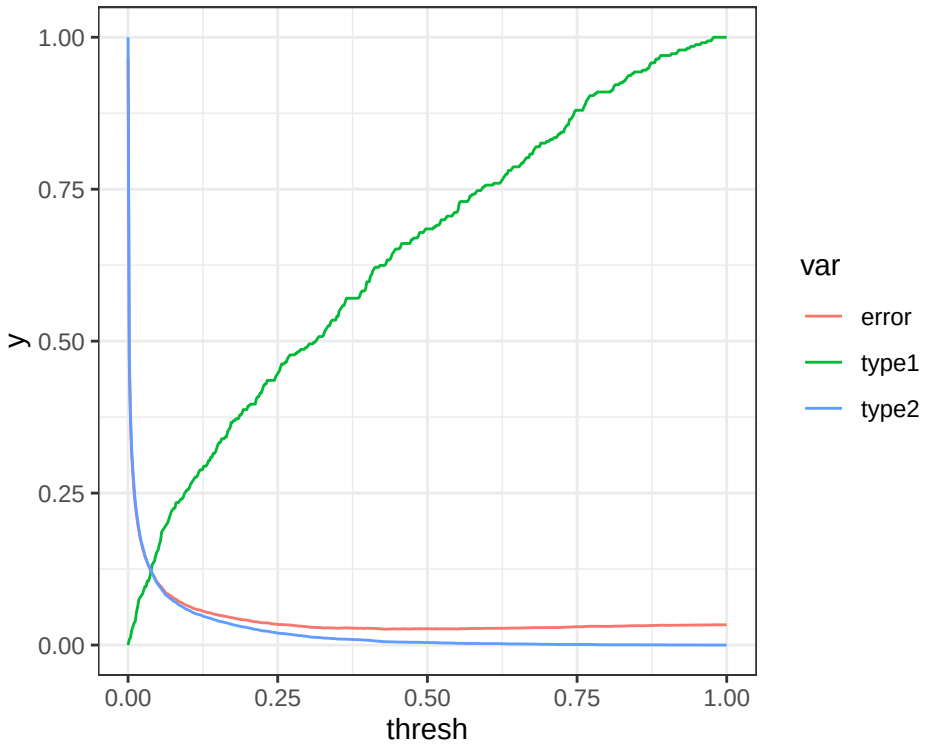
- The overall mis-classification rate is 4.07%
- The percentage of true defaulters that are mis classified is 39.04%
- And 2.87% is the percentage of non-defaulters that are misclassified.

Generally, as one of these rates goes up, the other goes down.

We can try out different thresholds and see what happens:

```
source("https://raw.githubusercontent.com/rafamoral/courses/main/intro_stats_ml/script
```

```
e <- calcERR(prob, Default$default,
             thresh = seq(0, 1, length.out = 500))

e %>%
  pivot_longer(2:4,
               names_to = "var",
               values_to = "y") %>%
  ggplot(aes(x = thresh, y = y, col = var)) +
  theme_bw() +
  geom_line()
```



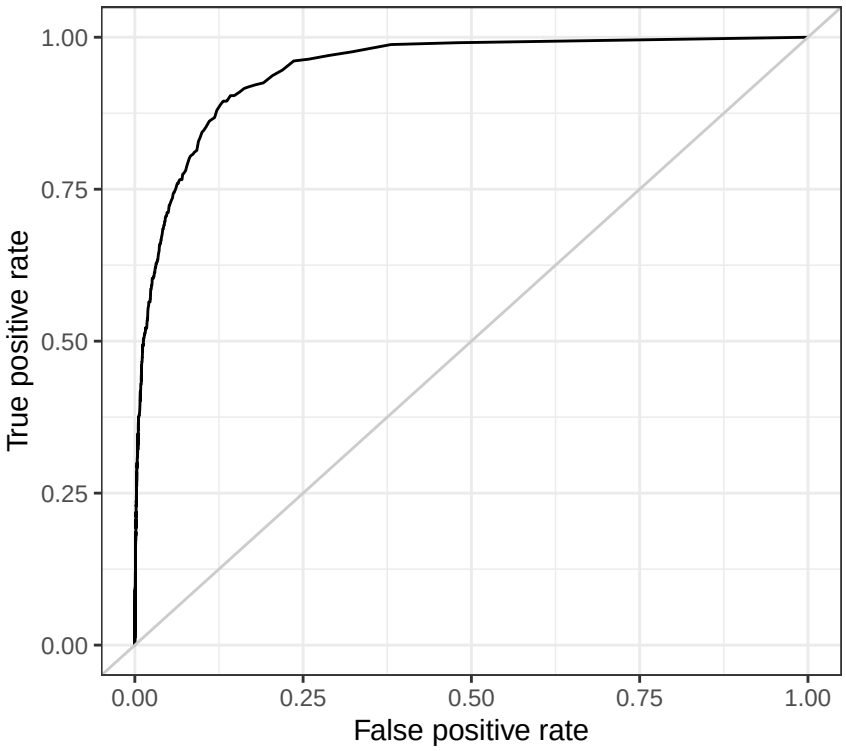
- The red line is the overall error rate. This is minimised when the threshold is .5 (always)
- The green line shows the error rate for the defaulters (FNR, type I)
- The blue line shows the error rate for the non-defaulters (FPR, type II).

ROC curve

The ROC (receiver operating characteristic) curve is a plot of the true positive rate versus the false positive rate for different thresholds.

It is just another way of presenting the information in the previous plot,

```
e %>%
  ggplot(aes(x = type2, y = 1 - type1)) +
  theme_bw() +
  geom_line() +
  geom_abline(intercept = 0, slope = 1, col = "grey80") +
  xlab("False positive rate") +
  ylab("True positive rate")
```



As we would like a high true positive rate and a low false positive rate the ideal curve hugs the top left corner.

The area under the ROC curve (AOC) is a measure of the overall value of the classifier, the closer AOC is to 1 the better.

```
auroc <- approxfun(x = e$type2, y = 1 - e$type1)
```

```
Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  collapsing
to unique 'x' values
```

```
## approximating using trapezoidal rule
x_grid <- seq(0, 1, length = 500)
y_grid <- auroc(x_grid)
sum((y_grid[-1] + y_grid[-length(y_grid)]) * diff(x_grid)[1] / 2)
```

```
[1] 0.95
```

```
## approximating using adaptive quadrature
integrate(auroc, lower = 0, upper = 1)
```

```
0.95 with absolute error < 8.2e-05
```

K nearest neighbours

The Bayes classifier rule for future x_0 is to allocate it to the class $j = 1, 2, \ldots, J$ which assigns it the highest value of

$$P(Y = j|X = x_0)$$

To do this, you would need to know P, or at least be able to estimate it.

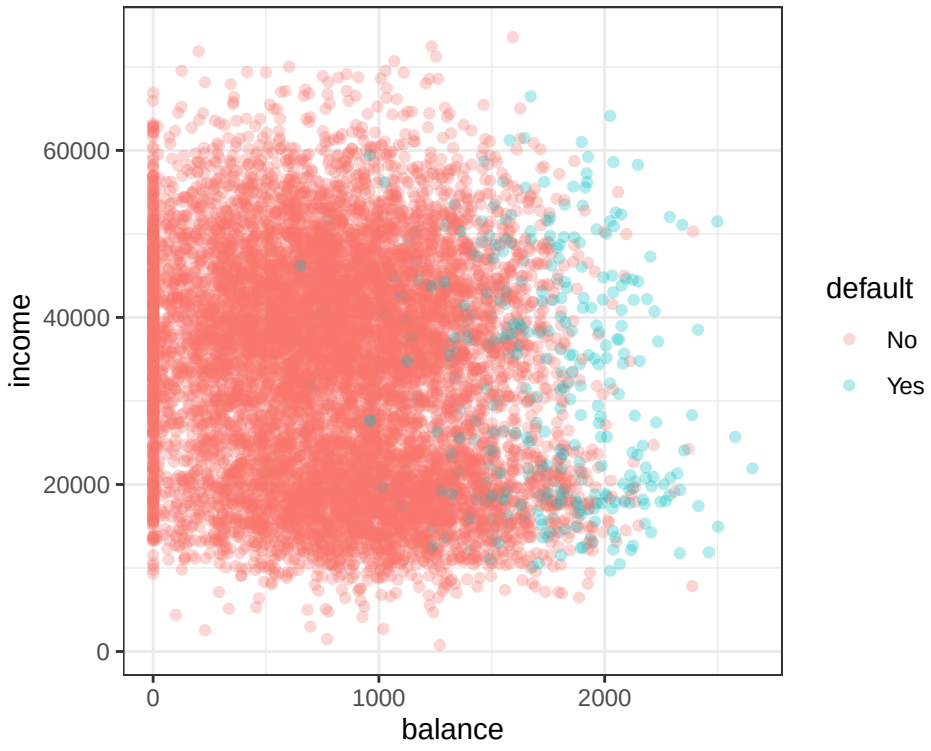
Knn estimates $P(Y = j|X = x_0)$ non-parametrically. It looks at the K nearest neighbours (in Euclidean distance) of the point x_0 and calculate the proportion of them that equal j.

In R, use the function `knn` in library `class`.

Defaults data, 2 predictors

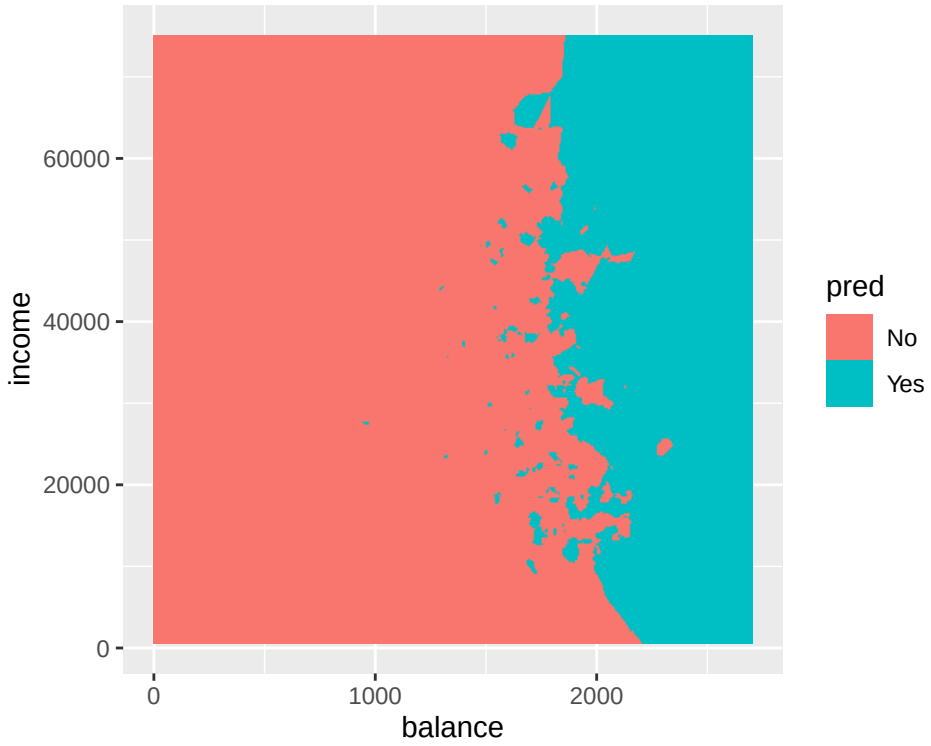
We start with two predictors, so we can plot the decision boundaries as before.
Since the two predictors have different scales, we should standardize them prior to using knn.

```
library(class)
head(Default)

  default student balance income default01
1      No      No     730  44362         0
2      No     Yes     817  12106         0
3      No      No    1074  31767         0
4      No      No     529  35704         0
5      No      No     786  38463         0
6      No     Yes     920   7492         0

xdata <- scale(Default[,3:4])

Default %>%
  ggplot(aes(x = balance, y = income, color = default)) +
  theme_bw() +
  geom_point(alpha = 0.3)
```



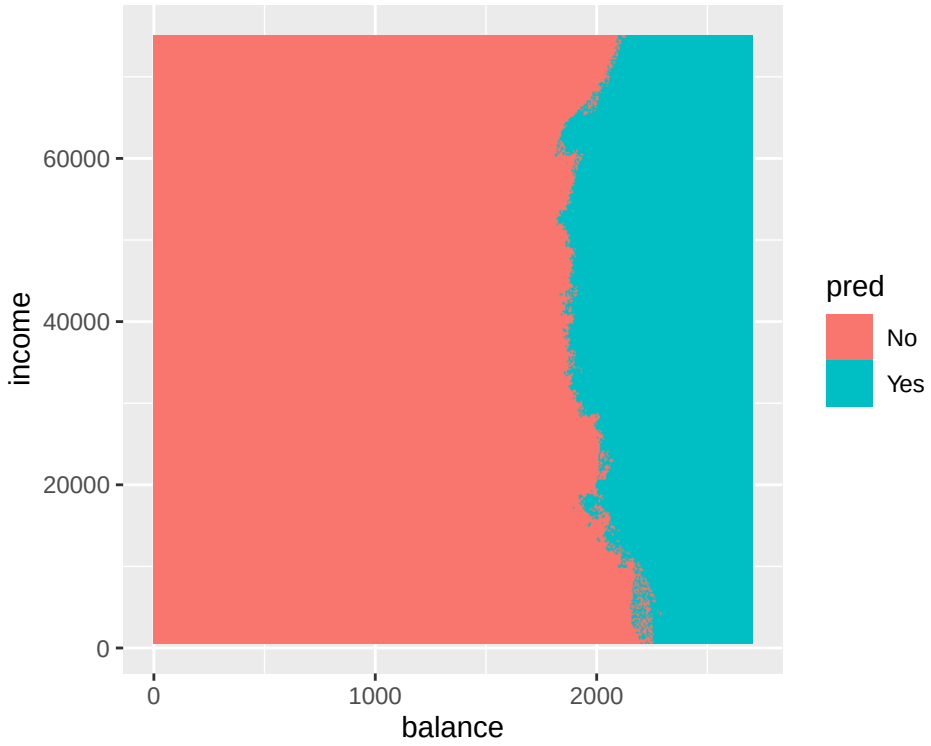

```
pred <- knn(xdata, xdata, Default[,1], k = 3)
```

```
grid <- expand.grid(
  balance = seq(0, 2700, length = 400),
  income = seq(700, 75000, length = 400)
)
means <- attr(xdata, "scaled:center")
sds <- attr(xdata, "scaled:scale")
grids <- scale(grid, center = means, scale = sds)
grid$pred <- knn(xdata, grids, Default[,1], k = 3)

grid %>%
  ggplot(aes(x = balance, y = income, fill = pred)) +
  geom_raster()
```



You can see the decision boundary is irregular. If we re-calculated knn with higher k the results should be smoother:

```
grid$pred <- knn(xdata, grids, Default[,1], k = 30)


grid %>%
  ggplot(aes(x = balance, y = income, fill = pred)) +
  geom_raster()
```



What do you think will happen if we increase k to 1000 say?

Next we calculate the confusion matrix

```
table(Default$default, pred)
```

```
     pred
        No  Yes
  No  9611   56
  Yes  172  161
```

- The overall mis-classification rate is 2.28%
- The percentage of true defaulters that are mis classified is 51.65%
- And 0.58% is the percentage of non-defaulters that are misclassified.

This gives a slightly better overall error rate and error rate for defaulters than logistic regression, but has a higher error rate for non-defaulters.

Because we used just $k = 3$ nearest neighbours, the bias of this fit will be low. However we suspect the variability is high.

The knn algorithm in R requires quantitative predictors. It is possible to construct a version using categorical predictors.

The response for knn must be categorical, there is no limitation on the number of categories

The methods are better compared by using a training set of data to calculate the fit and a test set to validate it.

```
set.seed(2024)
indTrain <- sample(nrow(Default), round(.8 * nrow(Default)))
indTest <- (1:nrow(Default))[-indTrain]

f1 <- glm(default ~ balance + income + student,
          family = "binomial",
          data = Default[indTrain,])
pred1 <- predict(f1, Default[indTest,], type = "response")
pred1 <- factor(ifelse(pred1 < .5, "No", "Yes"))
tab1 <- table(Default$default[indTest], pred1)
tab1

     pred1
        No  Yes
  No  1927    8
  Yes   42   23

xdata <- scale(Default[indTrain,3:4])
means <- attr(xdata,"scaled:center")
sds<- attr(xdata,"scaled:scale")
xdataTest <- scale(Default[indTest,3:4], center=means, scale=sds)

pred2 <- knn(xdata, xdataTest, Default[indTrain,1], k=3)
tab2 <-table(Default$default[indTest], pred2)
tab2

     pred2
        No  Yes
```

```
  No  1913   22
  Yes   42   23
```

The logistic error rate is 0.025

The logistic type 1 error rate is 0.646

The knn error rate is 0.032

The knn type 1 error rate is 0.646

This tells us that knn does not perform as well in terms of the out-of-sample error rate when compared to the logistic regression model, however it has the same type 1 error rate.

The value of k could be chosen by **cross-validation**.

KNN versus other methods

- Unlike the other methods, KNN is non parametric meaning it makes no assumptions about the shape of the decision boundary.
- When the decision boundary is highly non-linear, KNN is likely to beat other methods.
- But, KNN does not tell us which predictors are important.