STA4026S – Honours Analytics
Section A – Theory and Application of Supervised Learning
Lecture 3 – Classification Models

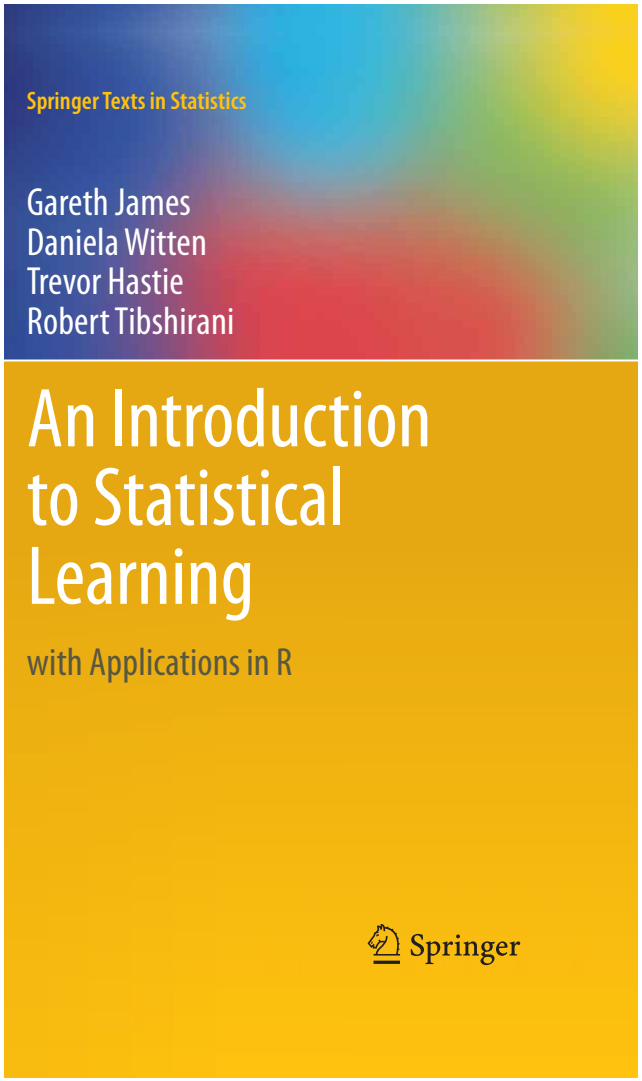
Stefan S. Britz
stefan.britz@uct.ac.za

Department of Statistical Sciences
University of Cape Town

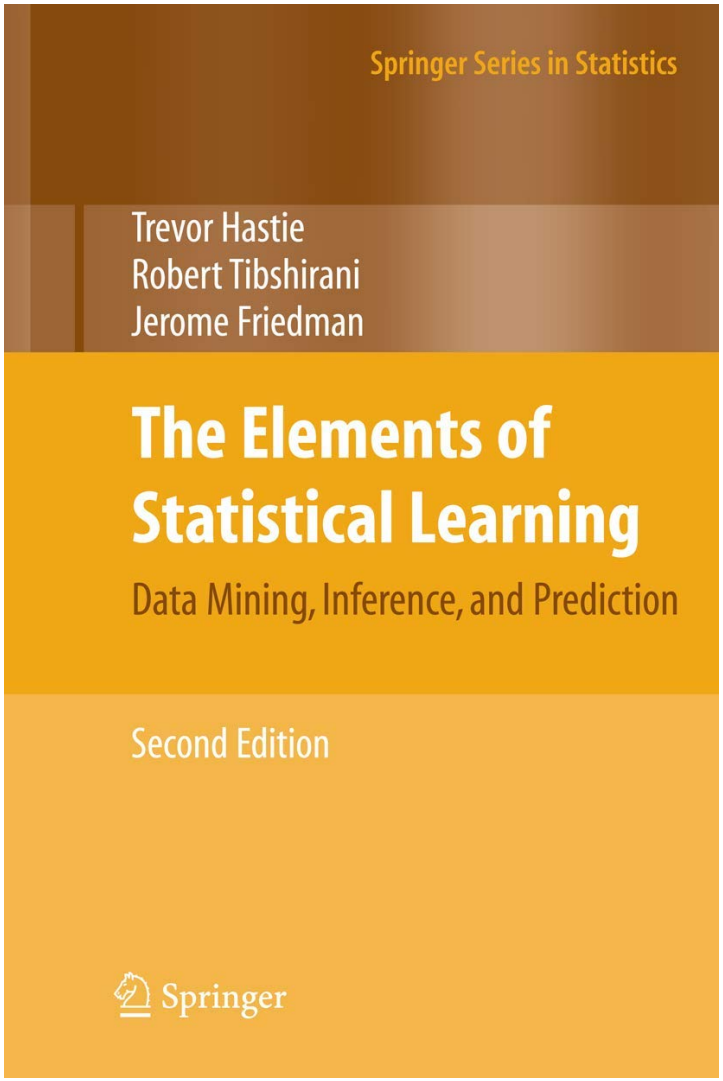
Introduction

- Thus far we have only been considering numeric response variables, generally referred to as regression problems
- We now turn to classification, i.e. supervised learning problems where the response variable is **categorical**
- More specifically, we will focus on methods of evaluating a classification model's performance
- For the sake of illustration we will only consider the **logistic regression** as linear classifier, although other classification models will be introduced in later chapters

Suggested reading



Chapter 4



Chapter 4

Classification

For now, let us consider a **binary** classification task, for example:

- A person can be classified as testing positive or negative for the SARS-CoV-2 virus, based on a range of physiological metrics.
- An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user's IP address, past transaction history, and so forth.
- Email spam filters have to classify incoming emails as either spam or legitimate mail, based on features of the text and sender.

Classification

With regression type problems (quantitative responses) we saw the flexibility and power that the **linear models** class offers.

- Our model equation followed a simple structure:

$$f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$$

- This makes relationships easy to **test** and **interpret**.
- We could apply **variable selection** techniques to identify important predictors for purposes of prediction.
- Using penalised regression techniques (LASSO, Ridge), we could further improve on prediction accuracy and explicitly keep a check on the **complexity** of our model.

Classification

Now define a **binary response variable**:

$$Y = \begin{cases} 1 & \text{if the outcome is 'class A'} \\ 0 & \text{if the outcome is 'class B'} \end{cases}$$

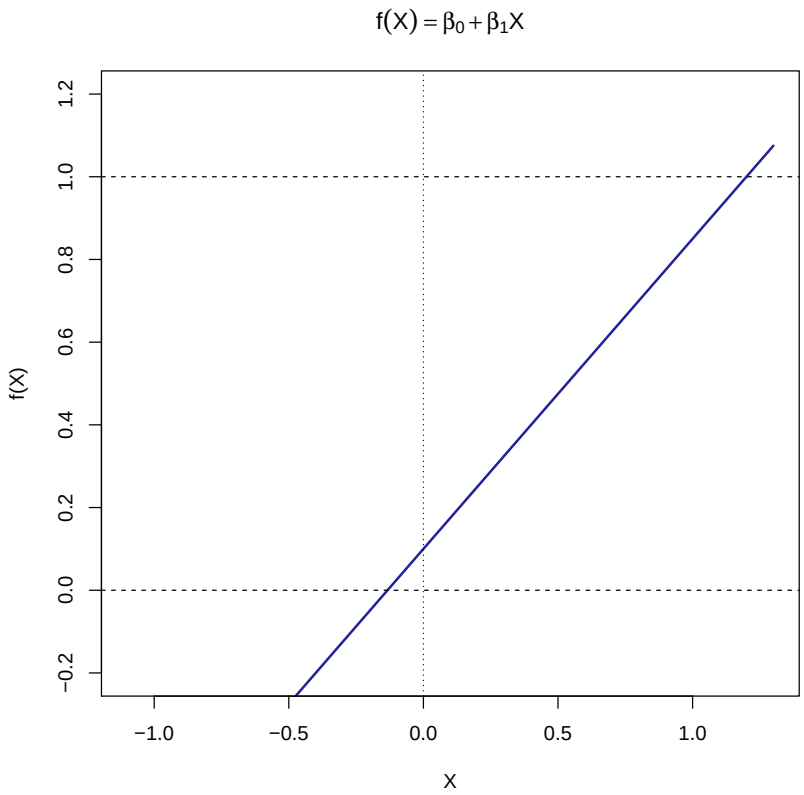
Can we use the same linear model structure for **classification problems**?

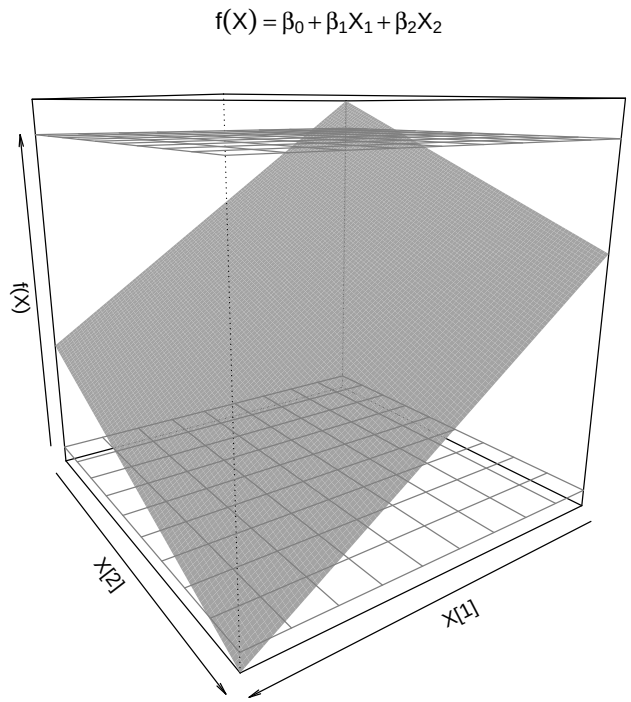
$$Y \approx f(X) = \beta_0 + \beta_1 X,$$

say for one predictor.

Classification

Does it make sense to do this? Our model output has to be a probability!





Left: SLR with one predictor. Right: MLR with two predictors.

Classification

- Using an appropriate model specification and data from some set of p features, define a decision rule based on a p-dimensional **decision boundary**
- This splits the feature space into two prediction regions corresponding to classes A and B respectively
- The classifier yielding this decision boundary is considered **linear** when the boundary itself is linear
- Some of the most widely applied linear classifiers include linear discriminant analysis, naive Bayes, support vector machine (with linear kernel), the perceptron, and **logistic regression**

Logistic Regression

Logistic Regression

We begin by modelling the probability that an observation belongs to some reference class:

$$\Pr(Y = j|\boldsymbol{X} = \boldsymbol{x}) = \begin{cases} p(\boldsymbol{x}) & \text{if } j = 1 \\ 1 - p(\boldsymbol{x}) & \text{if } j = 0 \end{cases} \tag{1}$$

or in other words, $Y \sim \text{Bernoulli}(p)$.

Therefore, $p(\boldsymbol{x})$ represents the probability that an observation is in class A given $\boldsymbol{X} = \boldsymbol{x}$.

Logistic Regression

Now we need to define an appropriate function in order to map the linear function to $p(x) \in [0, 1]$.

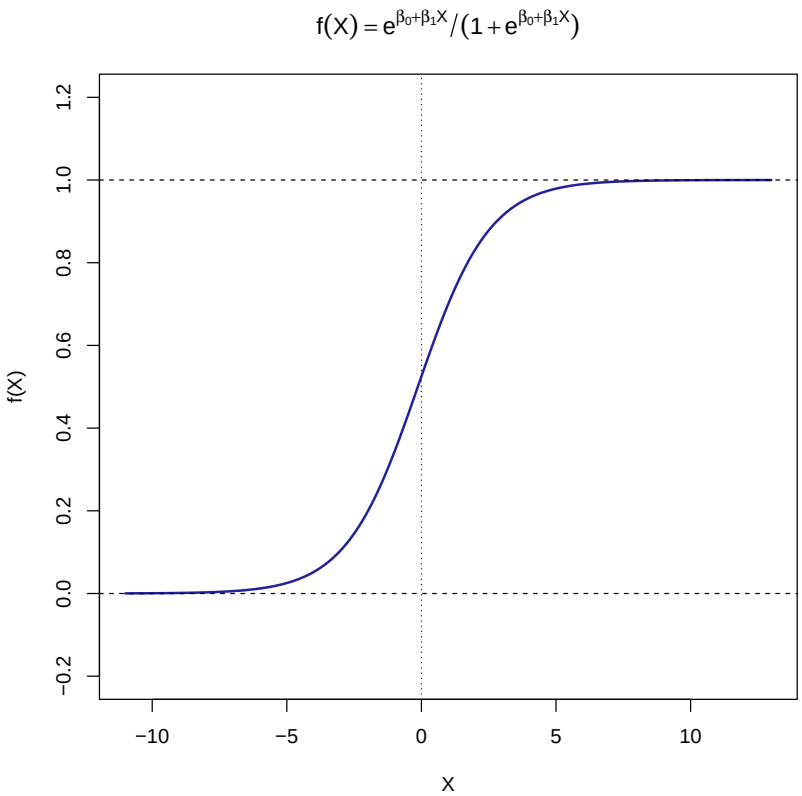
One option is the **logistic function**:

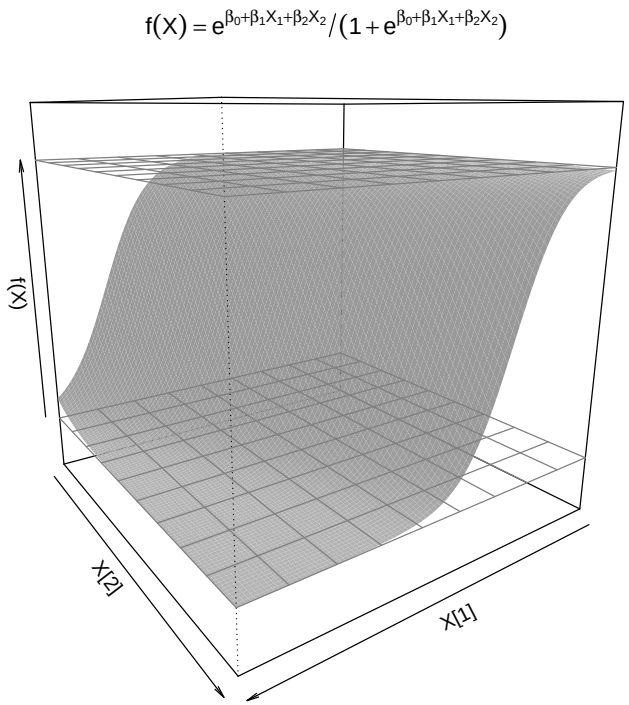
$$p(\boldsymbol{X}) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}} \tag{2}$$

Also referred to as the logistic map or sigmoidal curve, this simple S-shaped function asymptotes to 0 and 1.

Logistic Regression

We employ a **sigmoidal** transform of the linear model equation to ensure our model maps to a set of probabilities.





Left: Logistic with one predictor. Right: Logistic with two predictors.

Log odds

Through some simple manipulation, we can rewrite 2 as follows:

$$\log\left(\frac{p(\boldsymbol{X})}{1-p(\boldsymbol{X})}\right) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j \tag{3}$$

The left-hand is referred to as the **log odds** or **logit**

Here we see that the logistic regression model has a logit that is linear in X

Odds

Consider now the exponent of the log odds, i.e. the **odds**:

$$\text{odds} = \frac{p(\boldsymbol{x})}{1 - p(\boldsymbol{x})} \tag{4}$$

This is useful for interpreting logistic regression models (coefficients) because:

$$\begin{aligned} \text{odds} &= \frac{p(\boldsymbol{X} = \boldsymbol{x})}{1 - p(\boldsymbol{X} = \boldsymbol{x})} = \frac{\Pr(Y = 1 | X = x)}{\Pr(Y = 0 | X = x)} \\ &= \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p) \end{aligned} \tag{5}$$

This is referred to as the odds that $Y = 1$ vs. $Y = 0$ given $\boldsymbol{X}$. "How many times more likely" is $Y = 1$ than $Y = 0$ for this particular $\boldsymbol{x}$?

Note that odds can take on any value between 0 and ∞

Estimation

The most common approach for estimating the parameters $\boldsymbol{\beta}$ is by way of **maximising the likelihood**

The likelihood, a function of the parameters, is simply the joint probability of observing the responses:

$$\begin{aligned} L(\boldsymbol{\beta}) &= \Pr(Y = y_1 | \boldsymbol{X} = \boldsymbol{X}_1) \times \Pr(Y = y_2 | \boldsymbol{X} = \boldsymbol{x}_2) \times \ldots \\ &\quad \times \Pr(Y = y_n | \boldsymbol{X} = \boldsymbol{x}_n) \\ &= \prod_{i=1}^{n} p(\boldsymbol{x}_i)^{y_i} (1 - p(\boldsymbol{x}_i))^{1-y_i} \end{aligned} \tag{6}$$

Estimation

Maximising the likelihood is akin to maximising the log-likelihood:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \log(p(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - p(\boldsymbol{x}_i)) \right] \tag{7}$$

or, equivalently, minimising the so-called **cross-entropy**[1] **error** function:

$$\text{CE}(\boldsymbol{\beta}) = -\ell(\boldsymbol{\beta})$$

As usual, we set $\dfrac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0$, yielding the so-called **score** equations

These equations are nonlinear in $\boldsymbol{\beta}$, hence solving for $\hat{\boldsymbol{\beta}}$ requires an optimisation procedure such as iteratively reweighted least squares

[1] entropy is a measure of disorder in a system

Interpretation

The model parameters (regression coefficients) can also be interpreted in a similar way to the linear model, but with different meaning.

The simplest way is via the odds given in 5

For example, consider the case where X_1 increases by one unit, i.e. from x_1 to $x_1 + 1$. Then:

$$\begin{aligned}
\text{odds}(X_1 = x_1 + 1) &= \exp(\beta_0 + \beta_1(x_1 + 1) + \ldots + \beta_p x_p) \\
&= e^{\beta_1} \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p) \\
&= e^{\beta_1} \text{odds}(X_1 = x_1)
\end{aligned}$$

Therefore, increasing X_1 by one unit (whilst holding all predictors constant), increases the odds of $Y = 1$ by a multiple of e^{β_1}.

Prediction

Prediction for logistic regression is carried out exactly as before. We use parameter estimates to evaluate the probabilities associated with each outcome:

$$\hat{\Pr}(Y = 1|\boldsymbol{X}) = \hat{p}(\boldsymbol{X}) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X_1+...+\hat{\beta}_p X_p}}{1 + e^{\hat{\beta}_0+\hat{\beta}_1 X_1+...+\hat{\beta}_p X_p}} \tag{8}$$

Prediction

Prediction for logistic regression is carried out exactly as before. We use parameter estimates to evaluate the probabilities associated with each outcome:

$$\hat{\Pr}(Y = 1|\boldsymbol{X}) = \hat{p}(\boldsymbol{X}) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X_1+\ldots+\hat{\beta}_p X_p}}{1 + e^{\hat{\beta}_0+\hat{\beta}_1 X_1+\ldots+\hat{\beta}_p X_p}} \tag{8}$$

In order to classify an observation as belonging to one class or another, we need to apply a decision rule to this probability.

For multiclass classification with K different levels, we will classify an observation to the class with the highest estimated probability:

$$\hat{Y} = \underset{k}{\text{argmax}} \left[\hat{p}_k(\boldsymbol{X})\right],$$

where $k = 1, \ldots, K$.

Prediction

In the binary case, this implies that the decision rule has a threshold of 0.5:

$$\hat{Y} = \begin{cases} 1 & \text{if } \hat{p}(\boldsymbol{X}) \geq 0.5 \\ 0 & \text{if } \hat{p}(\boldsymbol{X}) < 0.5 \end{cases} \tag{9}$$

We will return to this decision rule later

- Note that logistic regression produces a **linear decision boundary** since the logit function is linear in the model predictors
- We may still apply linear decision boundaries to non-linear problems if appropriate transforms of the inputs can be found

Example 3 – Default data

- Dataset: Simulated `Default` dataset ($n = 10,000$):
  - `default` : A factor with levels No (0 - blue) and Yes (1 - orange) indicating whether the customer **defaulted** on their debt
  - `student` : A factor with levels No and Yes indicating whether the customer is a student
  - `balance` : The average balance that the customer has remaining on their credit card after making their monthly payment
  - `income` : Income of customer
- Problem: Predicting whether an individual will default on their credit card payment, on the basis of annual income and monthly credit card balance.

Example 3 – Default data

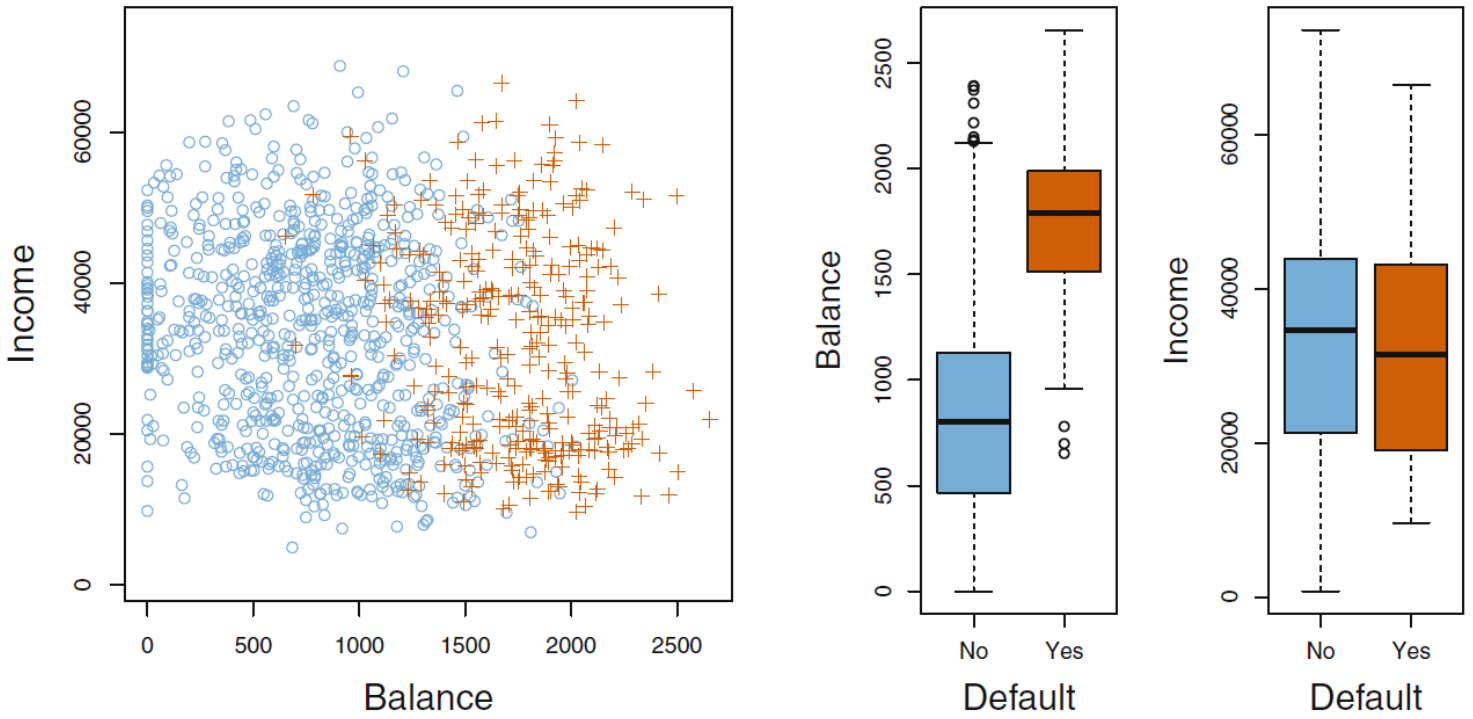


FIGURE 4.1. *The* `Default` *data set.* Left: *The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.* Center: *Boxplots of* `balance` *as a function of* `default` *status.* Right: *Boxplots of* `income` *as a function of* `default` *status.*

Example 3 – Default data Model fit

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-10.87	0.49	-22.08	0.00
studentYes	-0.65	0.24	-2.74	0.01
balance	0.01	0.00	24.74	0.00
income	0.00	0.00	0.37	0.71

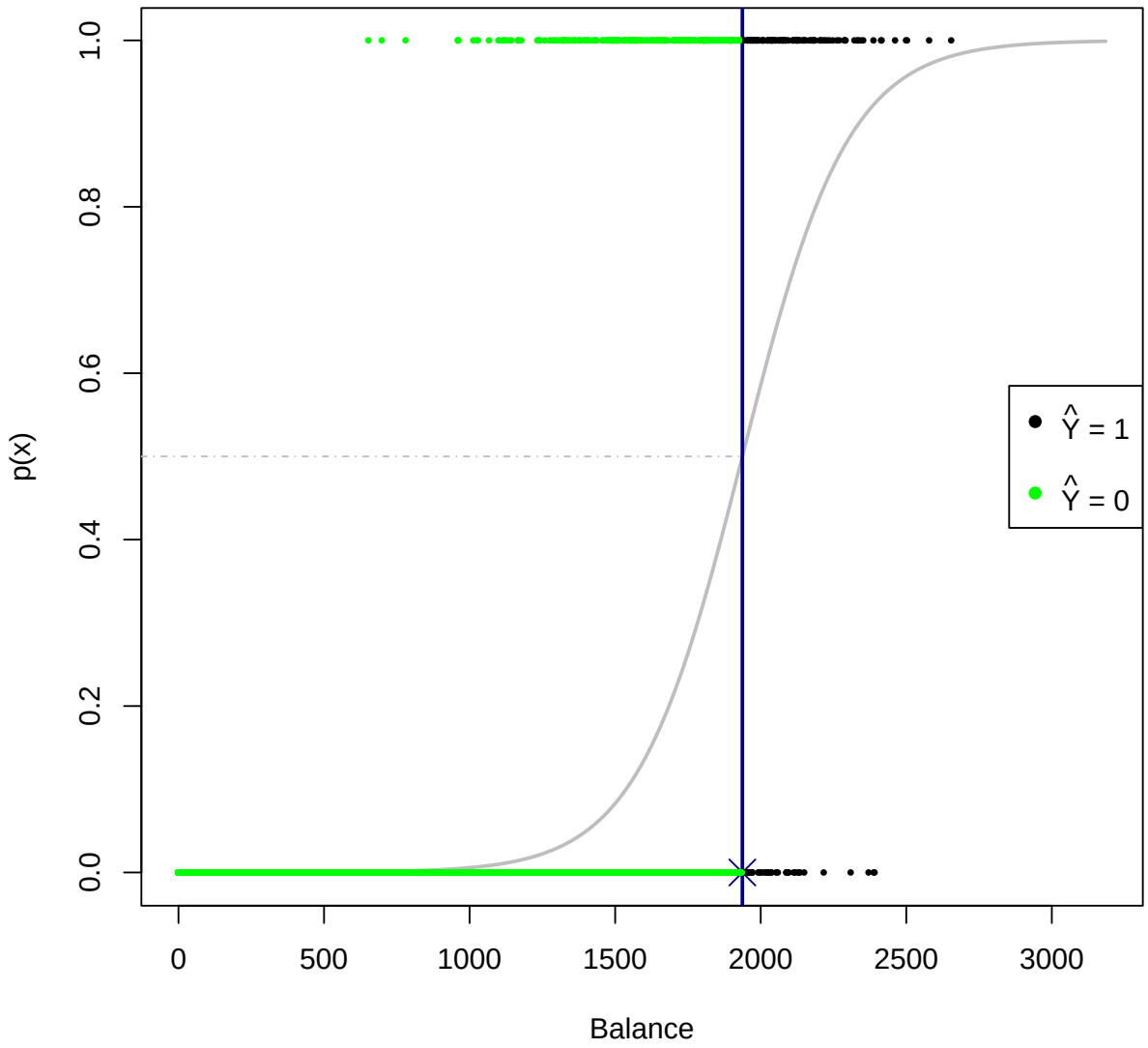
Example 3 – Default data Model interpretation

- The z-statistic in the R table plays the same role as the t-statistic in regression, and is computed in the same way.
- In particular, there is strong evidence to support the conclusion that student status and account balance influence default behaviour in the population.
- A non-student is twice as likely to default compared to a student ($e^{\beta_1} \approx 1/2$), holding balance and income constant.
- A one unit increase in the account balance increases the odds of defaulting by a factor of 1.006. What about a 100 unit increase?
- Income is **not** significantly associated with default behaviour in the population.

Decision boundaries

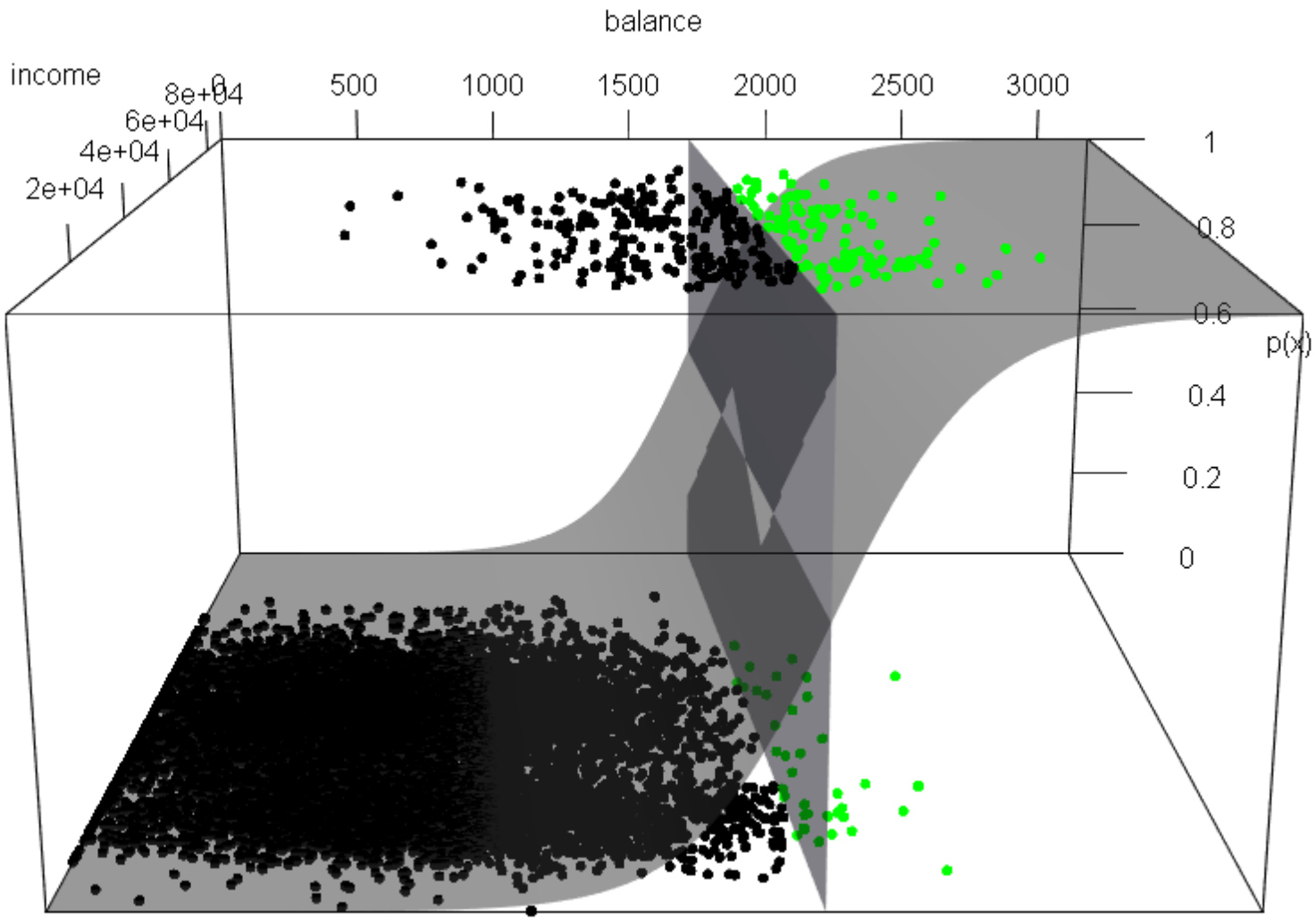
- Consider now the case where $p = 1$, where we only include the `balance` variable
- Apply the decision rule shown in 9, i.e. apply a threshold predicted probability of 0.5
- For the 1-dimensional case, the decision boundary is just a single point

Decision boundaries



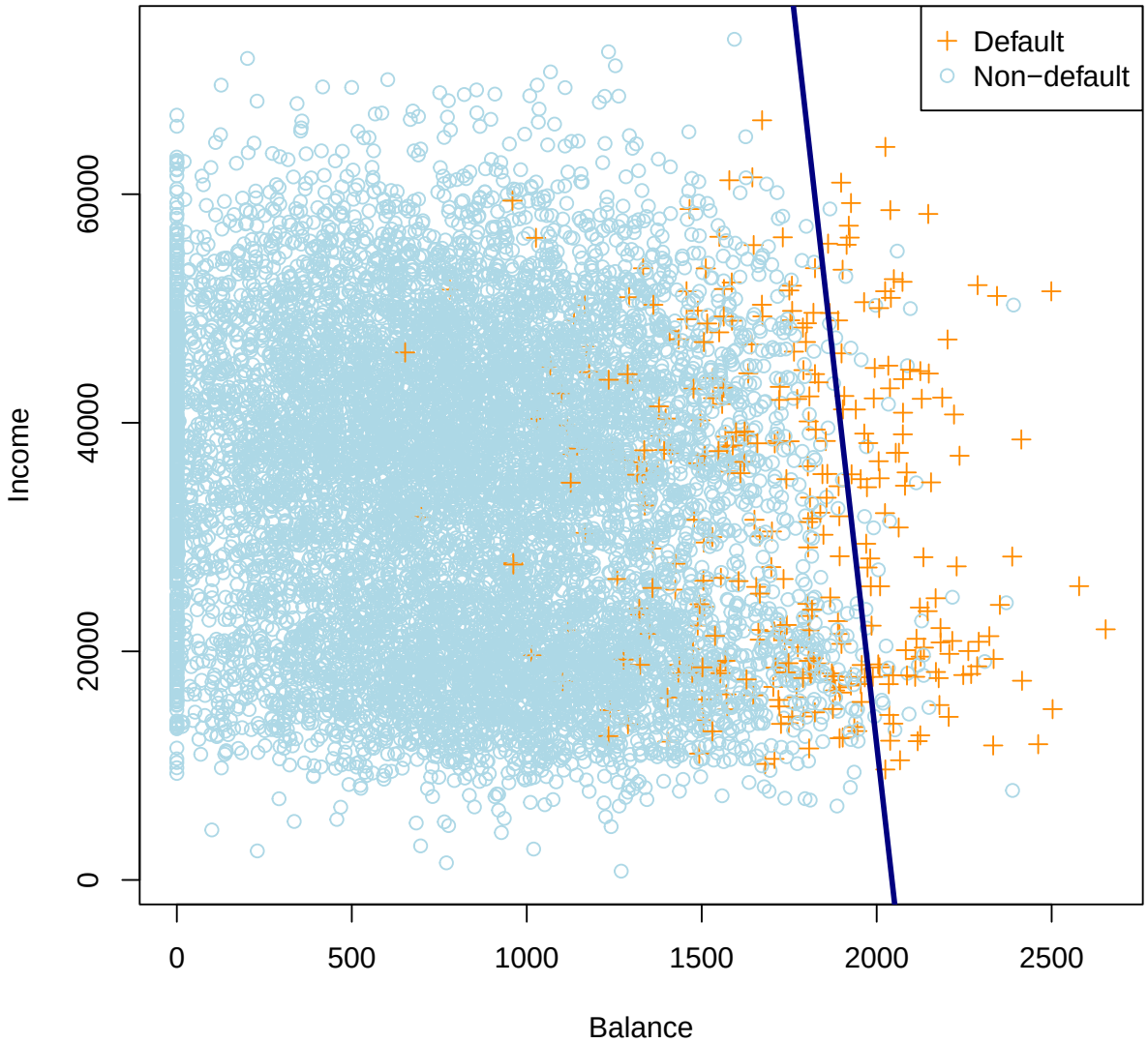
Decision boundaries

The linearity of the decision boundary is better illustrated for $p = 2$, so let's add the `income` variable to the model.



Decision boundaries

Here we can see that the contour on the sigmoid surface corresponding to a constant threshold is straight line, yielding the linear decision boundary:



Decision boundaries

Note that the decision boundary for the decision rule threshold $\hat{p}(\boldsymbol{X}) \geq 0.5$ corresponds to the set of points in the parameters space for which the odds are 1 (or, equivalently, the log-odds are zero)

Using this fact allows us to easily determine the boundary

We can visualise the decision boundary up until $p = 3$, please see the notes for an illustration

Now that we have sufficiently explored the logistic regression's decision boundaries, we turn our attention to measuring the accuracy of the classifications resulting from these decision boundaries.

Model Evaluation

Confusion Matrix

The **accuracy** of our model can be measured by the number of incorrectly classified observations:

$$\text{Error} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left(Y_i \neq \hat{Y}_i\right) = \frac{\#\text{ of misclassifications}}{N}$$

Confusion Matrix

The **accuracy** of our model can be measured by the number of incorrectly classified observations:

$$\text{Error} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left(Y_i \neq \hat{Y}_i\right) = \frac{\#\text{ of misclassifications}}{N}$$

A useful way of representing the observed vs. predicted classes, is by way of a **confusion matrix**. Looking at the `Default` data again:

```
table(yhat, y, dnn = c('predicted label', 'true label'))
```

```
                 true label
predicted label    No   Yes
            No   9627   228
            Yes    40   105
```

Here the off-diagonals represent **misclassification**, and the error rate is calculated as $\frac{228+40}{10000} = 0.0268$. **Is this good?**

Confusion Matrix Example 1

Hypothetical scenario 1:

Suppose the bank's president is irrational, and their "gut feeling algorithm" tells them to classify everyone as 'No':

	true label	
predicted label	No	Yes
No	9667	333
Yes	0	0

They look at their classification error of 3.33%, see that it's basically the same as yours of 2.68%, and fire you for "wasting precious company resources"...

Wait, what just happened?!

Confusion Matrix Example 2

Hypothetical scenario 2:

Suppose two companies each produce a device that immediately classifies someone as either having the Covid-19 disease or not. They are tested on 50 healthy and 50 infected patients, and the results are as follows:

Company A

	true label	
predicted label	No	Yes
No	50	10
Yes	0	40

Company B

	true label	
predicted label	No	Yes
No	40	0
Yes	10	50

Both companies have a 10% error rate. However, Company A will inform 20% of infected people that they do not have the disease, whilst Company B will inform 20% of healthy people that they have Covid-19!

Classification Error(s)

Clearly, all errors are not the same (think of the repercussions of each!), so we define the following errors:

1. False Positive (FP): Classify $\hat{Y} = 1$ when $Y = 0$.
2. False Negative (FN): Classify $\hat{Y} = 0$ when $Y = 1$.

Classification Error(s)

Clearly, all errors are not the same (think of the repercussions of each!), so we define the following errors:

1. False Positive (FP): Classify $\hat{Y} = 1$ when $Y = 0$.
2. False Negative (FN): Classify $\hat{Y} = 0$ when $Y = 1$.

These quantities, together with the True Positives (TP) and True Negatives (TN), make up the **confusion matrix**:

		True State	
		Negative	Positive
Prediction	Negative	TN	FN
	Positive	FP	TP
		Tot. Neg.	Tot. Pos.

Classification Performance Measures

We can now define several metrics to describe the performance of a classification algorithm across various aspects, some of which have several equivalent names:

- True Positive Rate = TPR = Sensitivity = Recall = $\frac{TP}{Tot.Pos.}$
- True Negative Rate = TNR = Specificity = $\frac{TN}{Tot.Neg.}$
- Positive Predictive Value = PPV = Precision = $\frac{TP}{TP+FP}$
- Negative Predictive Value = NPV = $\frac{TN}{TN+FN}$

Classification Performance Measures

The complements of these values yield the corresponding errors

Some other useful metrics include:

- The harmonic mean of precision and sensitivity (F1 Score), which is a popular performance metric, especially in computer vision:

  $$\text{F1 Score} = 2\frac{PPV \times TPR}{PPV + TPR}$$

- $\text{Balanced Accuracy} = \frac{TPR + TNR}{2}$

Logistic Regression – Errors & Accuracy

Since we now know that classification errors are much more nuanced than just looking at the misclassification rate, can we measure how much more powerful your logistic regression model was than the president's "gut feeling algorithm"? We note that

Logistic Regression – Errors & Accuracy

Since we now know that classification errors are much more nuanced than just looking at the misclassification rate, can we measure how much more powerful your logistic regression model was than the president's "gut feeling algorithm"? We note that

- We were actually being naive about our choice of decision threshold when attempting to map probabilities to classifications.
- By setting $\hat{Y} = 1$ if $\hat{f}(X) > 0.5$, we are attempting to approximate the Bayes classifier which minimises the total error rate.
- This implicitly assumes that the two error types (FP & FN) are equally bad.
- But here we are specifically interested in classifying **default** events.
- We may thus modify the threshold level in order to accommodate the asymmetric cost of misclassification.

Logistic Regression – Errors & Accuracy

Let's see the effect of changing the threshold level from 0.5 to 0.2:

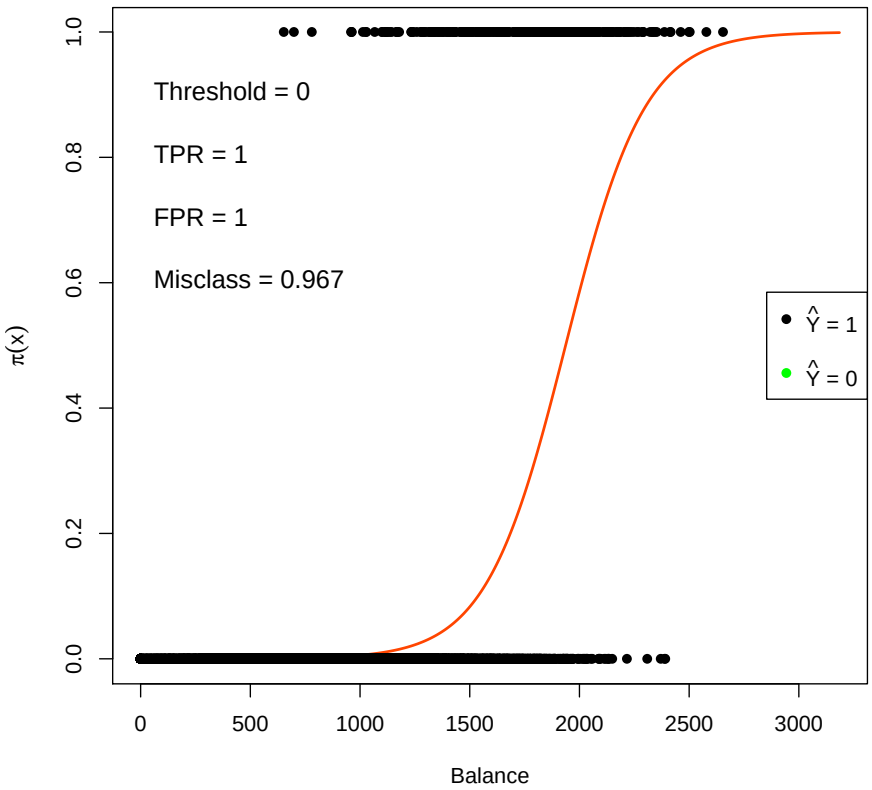
```
yhat2 <- ifelse(phat >= 0.2, 'Yes', 'No')
table(yhat2, Y, dnn=c('predicted label', 'true label'))
```

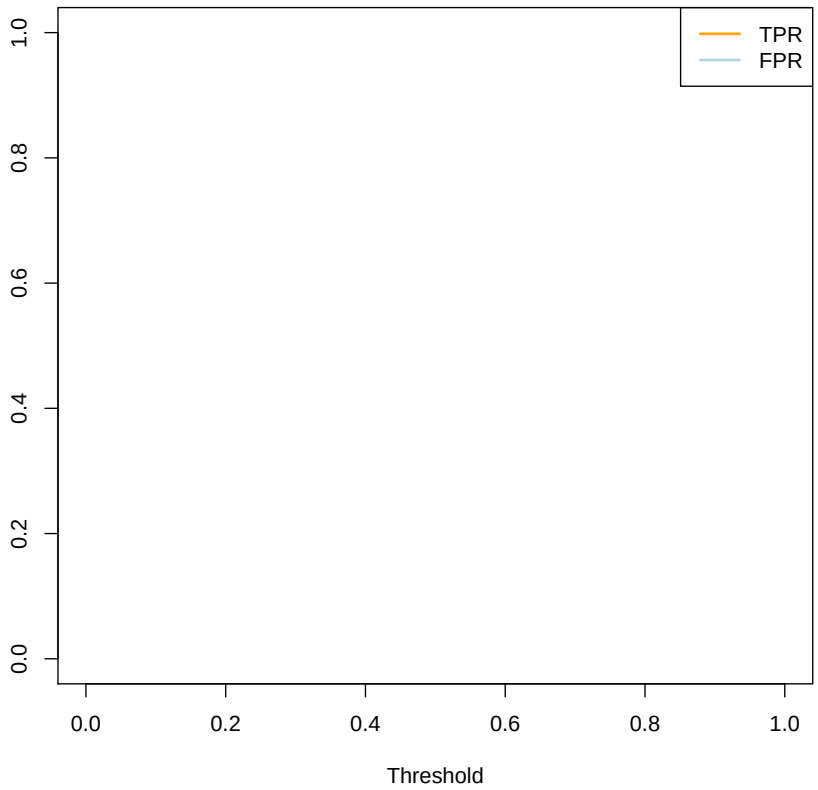
```
                 true label
predicted label    No   Yes
             No  9390   130
             Yes  277   203
```

- We have improved the classification accuracy for defaulters (TPR = Sensitivity): $0.315 \rightarrow 0.61$
- But at the cost of increased misclassification of non-defaulters (FPR = $1-$Specificity): $0.004 \rightarrow 0.029$
- We also note that this increased the classification error Error: $0.027 \rightarrow 0.041$

Logistic Regression – Errors & Accuracy

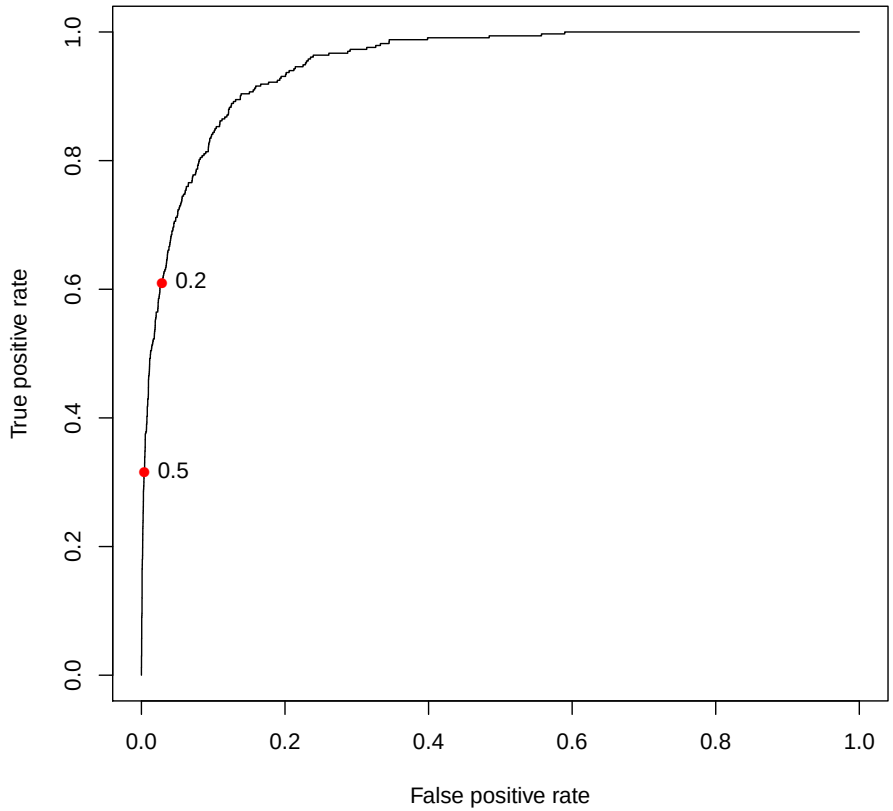
- So is logistic regression a good classifier for this problem?
- To determine this, we can vary the threshold, and analyse how this affects these two metrics, namely
  TPR = Sensitivity & FPR = $1-$Specificity





ROC Curve

- The most common way of displaying how TPR and FPR change as the threshold changes, is by plotting each on an axis for different values of the threshold.
- This is known as the Receiver Operating Characteristic (ROC) curve.



ROC Curve

- The ROC curve thus gives us an indication of the performance of a classifier over the entire range of decision rules.
- We can, therefore, compare different classifiers in a more rigorous way.
- A completely random classifier would (on average) lie on the diagonal of the ROC curve.
- A classifier constantly predicting one category would lie exactly on the diagonal.
- Can a classifier lie below the diagonal? How?

ROC Curve

- A perfect classifier would perfectly predict all positive cases and never falsely identify an observation as positive.
- Therefore, an ideal classifier would lie close to the top left corner of the graph.
- Consequently, when comparing models we seek ones that lie closer to this ideal.
- The way to identify this is by way of calculating the **area under the curve (AUC)**.
- The AUC is simply a value between 0.5 and 1 (technically 0 and 1) measuring a binary classification model's ability to distinguish between positive and negative responses.

Regularisation

Logistic regression regularisation

Previously, we considered the lasso, aka L_1 regularisation as a variable selection method for a linear (or any parameterised) regression model.

This was predicated on penalising the loss function, the RSS, which is **minimised** in order to find the estimated regression coefficients.

We can apply the same reasoning to logistic regression.

Although, since we estimate the regression coefficients by **maximising** the likelihood, the penalty term is subtracted, instead of added.

Let $\boldsymbol{\beta} = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_p \end{bmatrix}$. Now, using 2, the log-likelihood in 7 can be written as follows:

Logistic regression regularisation

$$
\begin{aligned}
\ell(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \left[y_i \log(p(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - p(\boldsymbol{x}_i)) \right] \\
&= \sum_{i=1}^{n} \left[y_i \log \left(\frac{e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i}}{1 + e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i}} \right) + \log \left(1 - \frac{e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i}}{1 + e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i}} \right) \right. \\
&\quad \left. - y_i \log \left(1 - \frac{e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i}}{1 + e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i}} \right) \right] \\
&= \sum_{i=1}^{n} \left[y_i \left(\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i \right) - y_i \log \left(1 + e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i} \right) - \log \left(1 + e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i} \right) \right. \\
&\quad \left. + y_i \log \left(1 + e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i} \right) \right] \\
&= \sum_{i=1}^{n} \left[y_i \left(\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i \right) - \log \left(1 + e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i} \right) \right]. \qquad (10)
\end{aligned}
$$

Logistic regression regularisation

Therefore, L_1 regularised logistic regression parameters are given by:

$$\hat{\boldsymbol{\beta}}_L = \underset{\boldsymbol{\beta}}{\text{argmax}} \left\{ \sum_{i=1}^{n} \left[y_i \left(\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i \right) - \log \left(1 + e^{\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i} \right) \right] - \lambda \sum_{j=1}^{p} |\beta_j| \right\}.$$

As in the linear regression model, the predictors are generally standardised and the intercept term is not penalised.

Once again, the optimsation methods used to solve for the coefficients are beyond the scope of this course.

Example 4 – Heart failure data

- We have medical records of 299 patients who experienced heart failure, with data collected during the patients' follow-up period
- Each patient profile contains 12 clinical features
- The goal is to predict the 13^{th} variable, namely death event (binary)
- Data are slightly imbalanced, with 33.9% of cases resulting in death
- Setting aside 20% of the data for testing, and fit the saturated logistic model to the training data

Example 4 – Heart failure data

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	15.52	7.09	2.19	0.03
age	0.06	0.02	3.05	0.00
anaemia1	0.19	0.39	0.49	0.63
creatinine_phosphokinase	0.00	0.00	1.63	0.10
diabetes1	0.34	0.39	0.89	0.37
ejection_fraction	-0.07	0.02	-4.01	0.00
high_blood_pressure1	-0.05	0.40	-0.12	0.90
platelets	-0.00	0.00	-0.84	0.40
serum_creatinine	0.66	0.20	3.28	0.00
serum_sodium	-0.11	0.05	-2.22	0.03
sex1	-0.81	0.46	-1.76	0.08
smoking1	0.13	0.46	0.27	0.79
time	-0.02	0.00	-6.00	0.00

Example 4 – Heart failure data

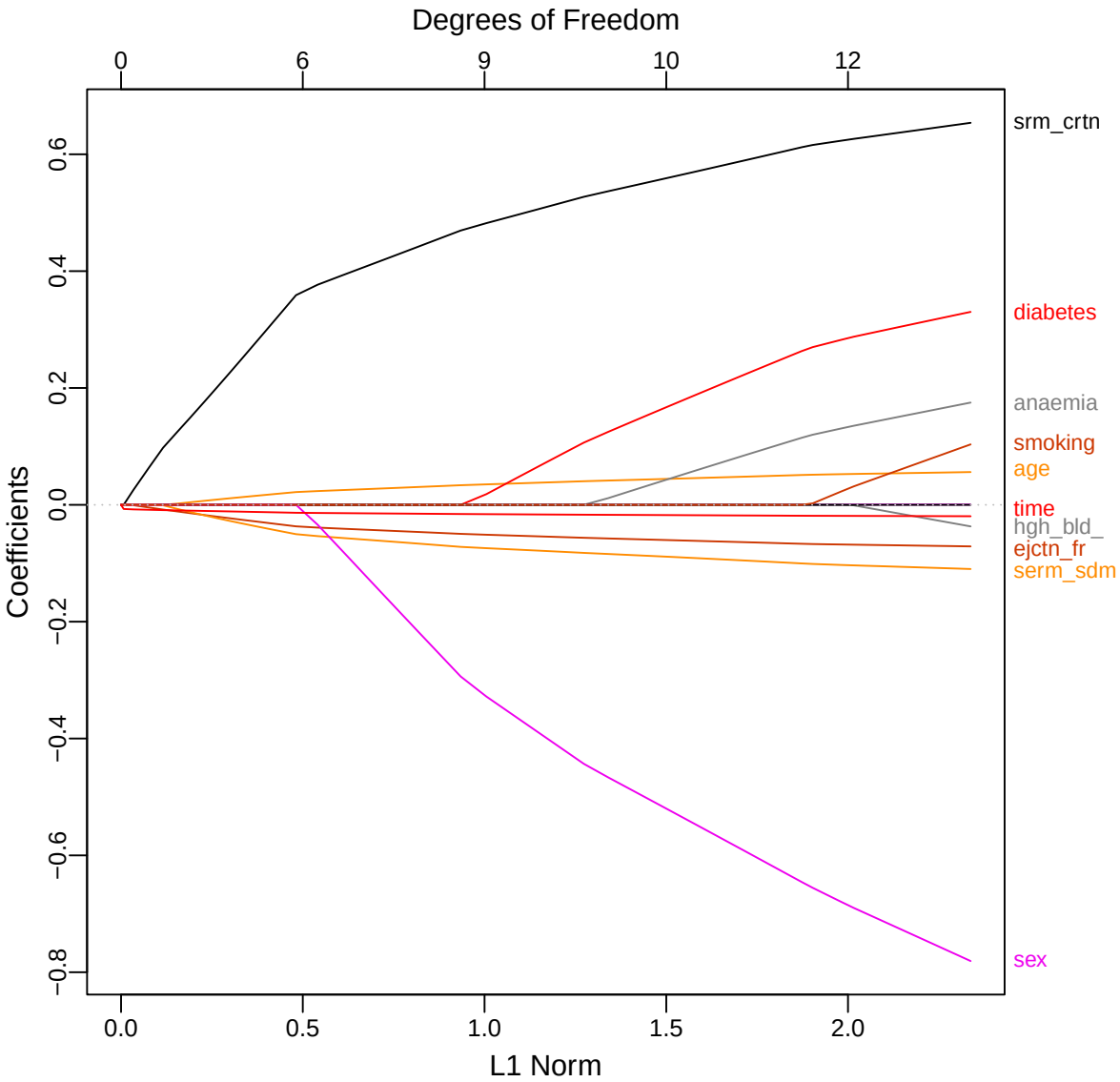
A few of the predictors are not statistically significant in this model at any reasonable significance level and should be removed

Most notably

- `high_blood_pressure1` (hypertension indicator)
- `smoking1` (smoking indicator)
- `anaemia1` (hemoglobin indicator)

We will use L_1 regularisation to perform the variable selection

Example 4 – Heart failure data

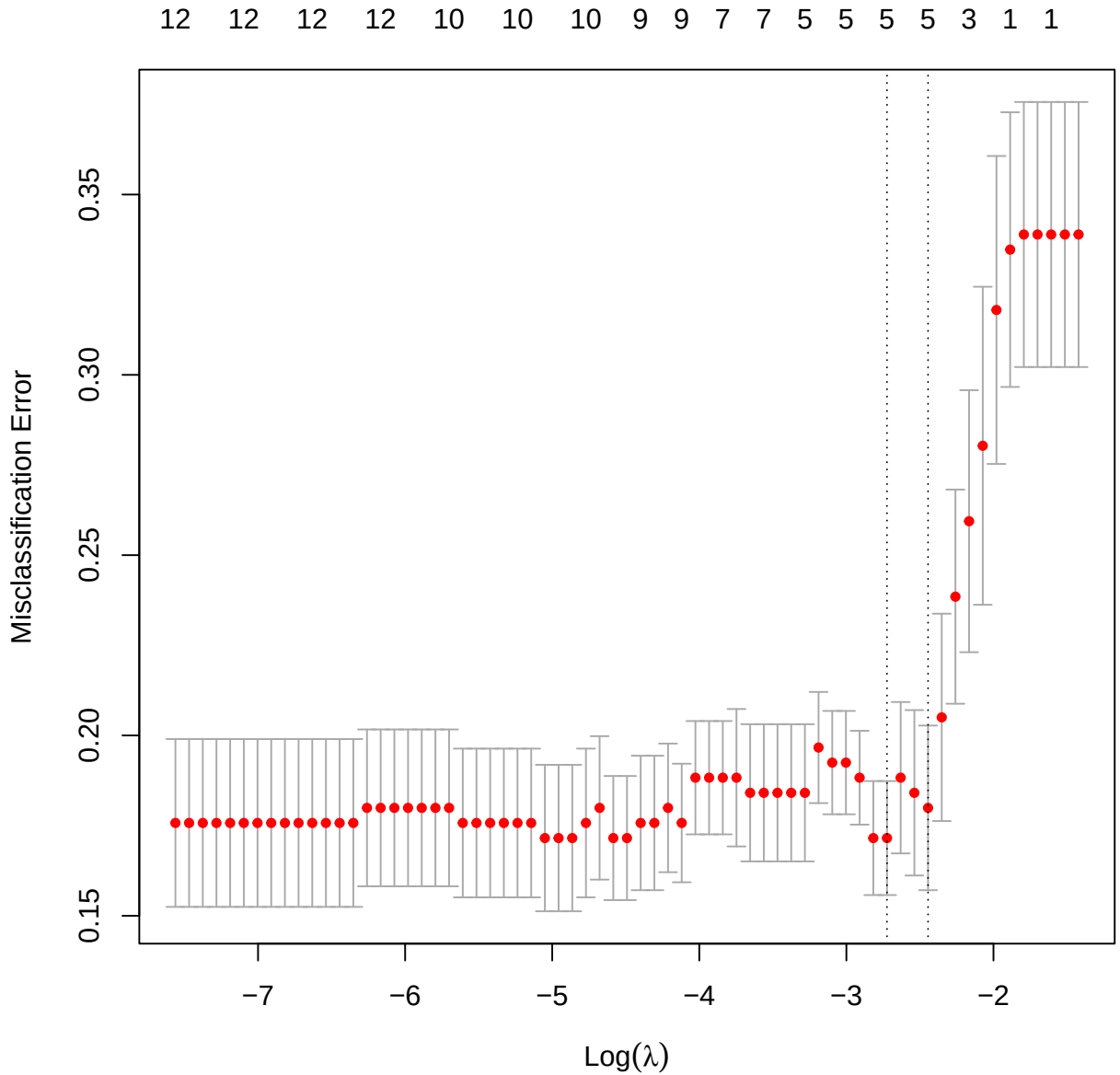


Example 4 – Heart failure data

Again we observe how how the regression coefficients are reduced to zero as the L_1 norm constraint decreases

To decide on the penalty to apply – and by extension the number of variables to drop – we again use 10-fold cross-validation with **classification error** as loss function.

Example 4 – Heart failure data



Example 4 – Heart failure data

- The best model according to cross-validated classification accuracy contains five features
- **However**, this model displays high variance!
- Setting different seeds will yield different results
- The selected features in this instance are the ones with non-zero coefficients

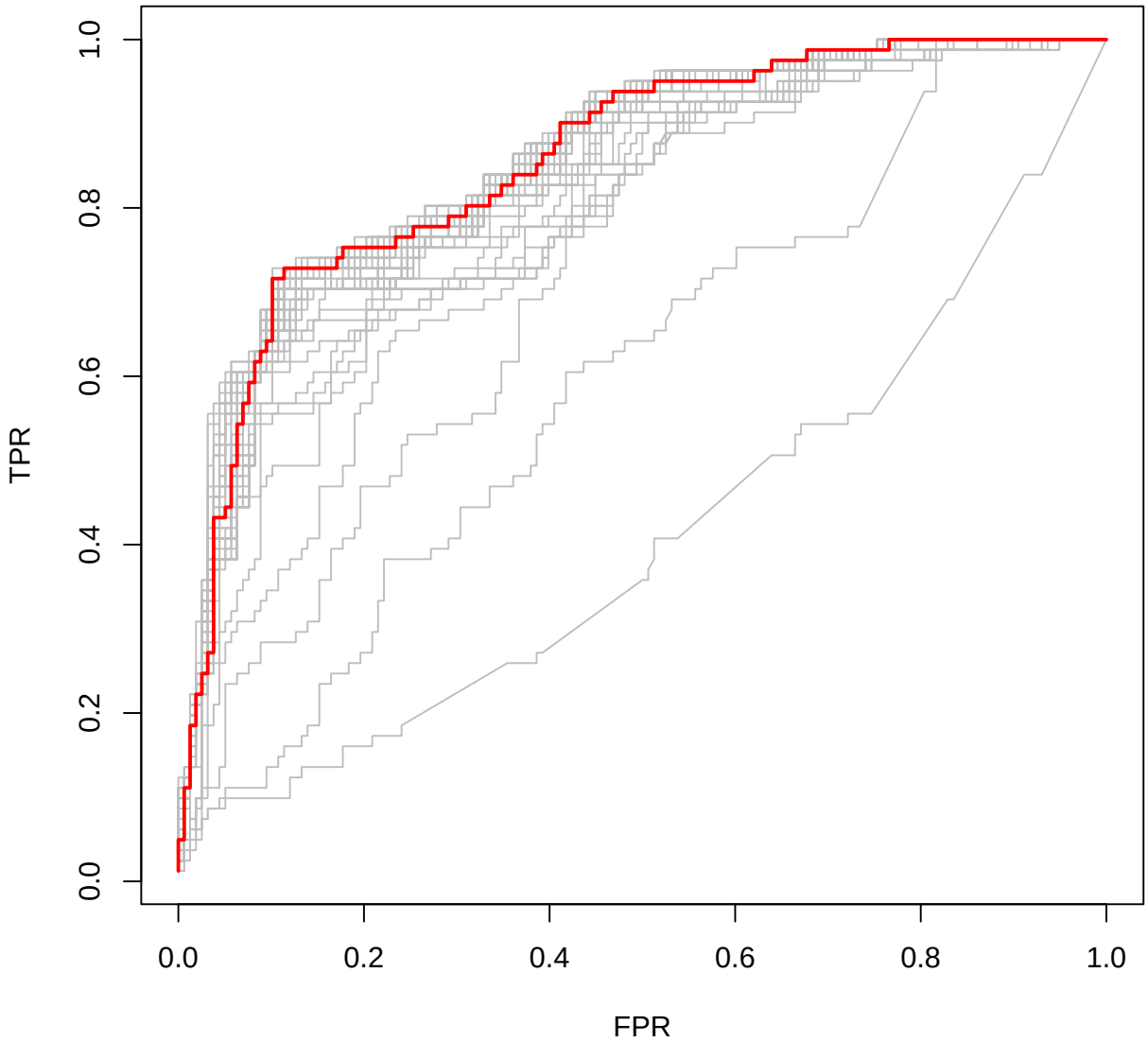
Example 4 – Heart failure data

	$\hat{\boldsymbol{\beta}}_{\text{Min}}$
X.Intercept.	3.646
age	0.009
anaemia1	.
creatinine_phosphokinase	.
diabetes1	.
ejection_fraction	-0.020
high_blood_pressure1	.
platelets	.
serum_creatinine	0.200
serum_sodium	-0.023
sex1	.
smoking1	.
time	-0.011

Example 4 – Heart failure data

- We select this model
- All the included variables were significant at $\alpha = 0.07$ in the saturated model
- `sex` was also borderline significant in the full model and had the largest absolute coefficient (although the largest standard error too)
- If we want to include this variable for physiological reasons, we can relax the regularisation penalty
- We can also optimise according to a different metric, for example the ROC AUC
- For the following plot we varied the penalty and measured the CV ROC AUC, with the curve yielding the highest value highlighted

Example 4 – Heart failure data



Example 4 – Heart failure data

- This model corresponds to a relatively small penalty, only dropping four variables from the model (retaining `sex`)
- The other two included variables have very small coefficients
- We now have three logistic regression models of varying complexity:
  - The full ("vanilla") model, containing some unnecessary variables
  - The 5-variable model (regularised according to CV accuracy)
  - The 8-variable model (regularised according to CV AUC)
- We must now select a model to apply to the test set
- If we want to optimise classification accuracy, that would be the 5-variable model
- For illustration, apply all three model to the test set

Example 4 – Heart failure data

	Accuracy	Recall	Spec.	Prec.	F1	AUC
Vanilla LR	0.833	0.800	0.844	0.632	0.706	0.899
L1 LR (CV accuracy)	0.867	0.667	0.933	0.769	0.714	0.944
L1 LR (CV AUC)	0.850	0.800	0.867	0.667	0.727	0.939

Example 4 – Heart failure data

	Accuracy	Recall	Spec.	Prec.	F1	AUC
Vanilla LR	0.833	0.800	0.844	0.632	0.706	0.899
L1 LR (CV accuracy)	0.867	0.667	0.933	0.769	0.714	0.944
L1 LR (CV AUC)	0.850	0.800	0.867	0.667	0.727	0.939

- Both approaches to the regularisation improved the results from the baseline vanilla logistic regression across most metrics
- Recall is the exception, which is arguably most important!
- The differences in results were relatively small, partly due to small sample size
- Note again high variability! A better approach is to repeat the CV, say, 100 times and track the distribution of results
- Next up: going non-linear