DATA2002

Logistic regression

Garth Tarr

THE UNIVERSITY OF SYDNEY

Logistic regression

Evaluating performance

Titanic survival

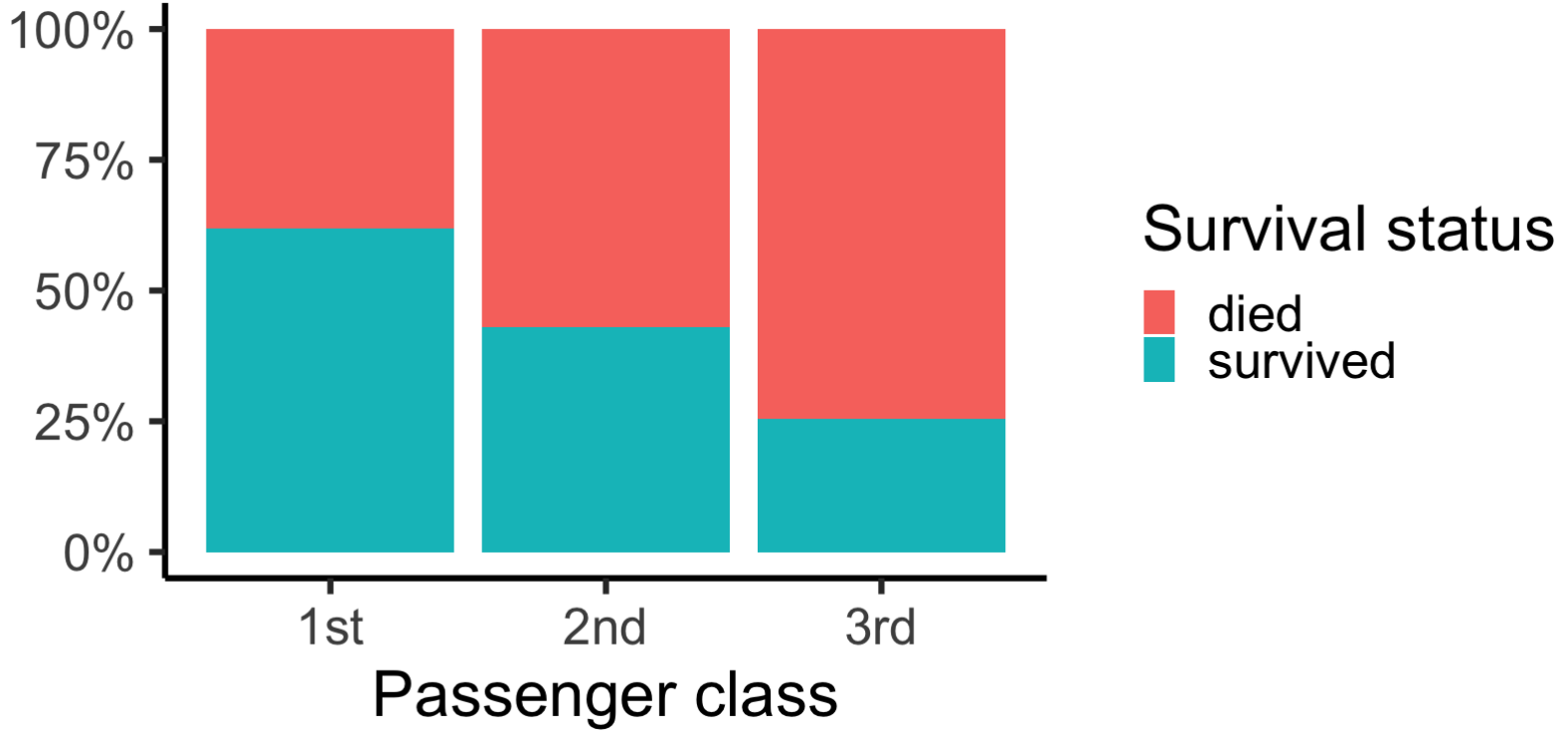
Data on passengers on the RMS Titanic, excluding the crew and some individual identifier variables.

```r
library(tidyverse)
# install.packages("vcdExtra")
data("Titanicp", package = "vcdExtra")
glimpse(Titanicp)
```

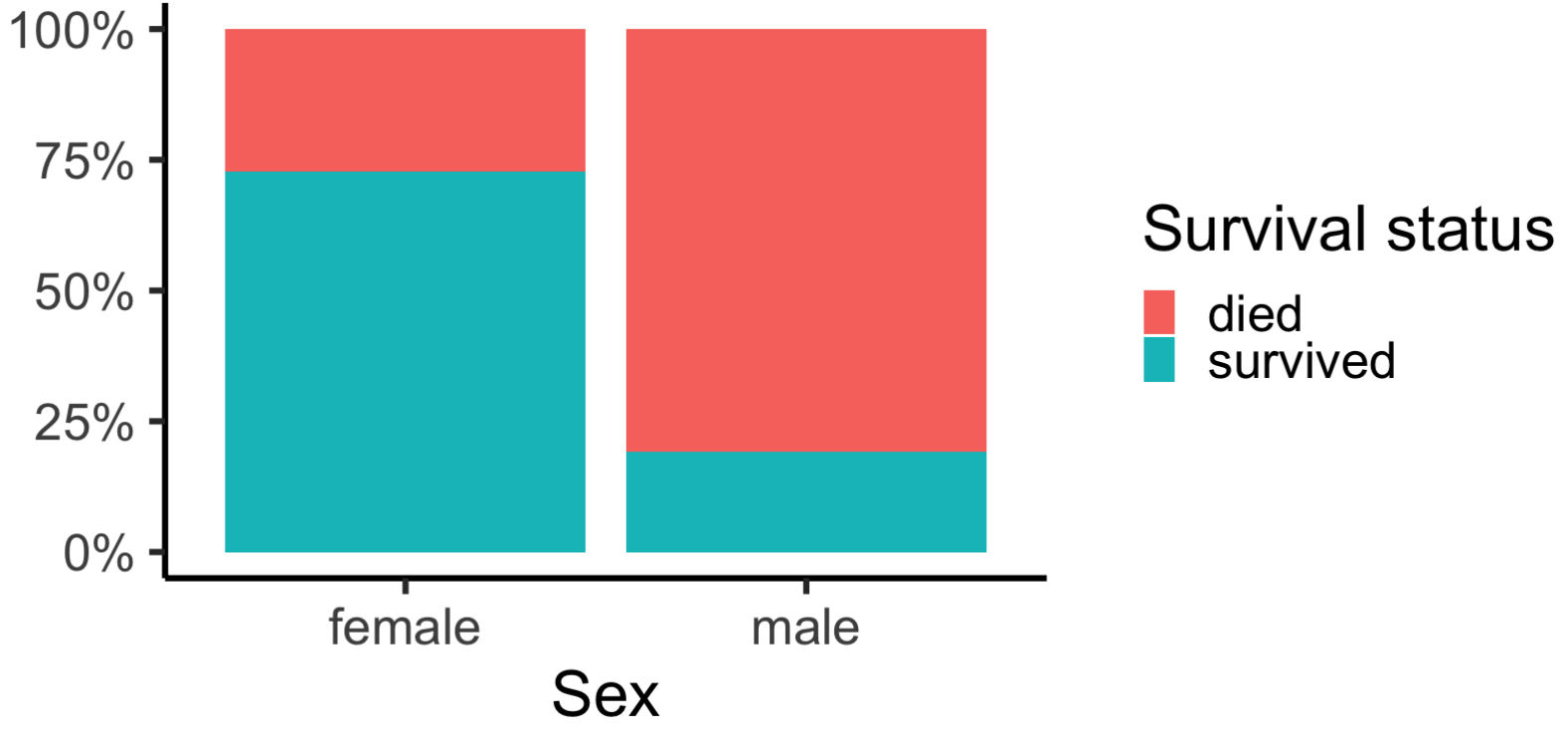
```
## Rows: 1,309
## Columns: 6
## $ pclass   <fct> 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1…
## $ survived <fct> survived, survived, died, died, died, sur…
## $ sex      <fct> female, male, female, male, female, male,…
## $ age      <dbl> 29.0000, 0.9167, 2.0000, 30.0000, 25.0000…
## $ sibsp    <dbl> 0, 1, 1, 1, 1, 0, 1, 0, 2, 0, 1, 1, 0, 0,…
## $ parch    <dbl> 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
```

- **pclass** a factor with levels `1st 2nd 3rd`
- **survived** a factor with levels `died survived`
- **sex** a factor with levels `female male`
- **age** passenger age in years (or fractions of a year, for children), a numeric vector; age is missing for 263 of the passengers
- **sibsp** number of siblings or spouses aboard, integer: `0:8`
- **parch** number of parents or children aboard, integer: `0:6`

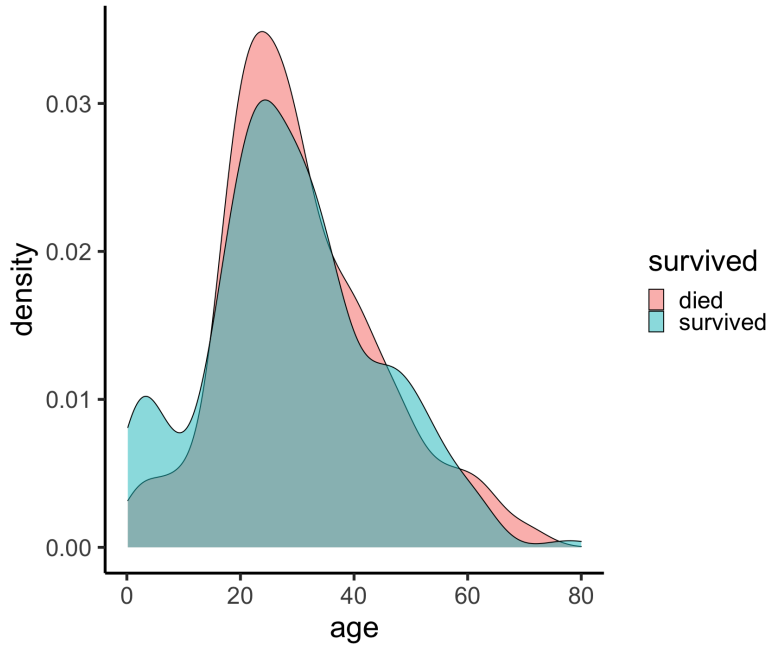
```
Titanicp %>% group_by(survived, pclass) %>% count() %>%
  ggplot(aes(x = pclass, y = n, fill = survived)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  labs(y = "", x = "Passenger class", fill = "Survival status")
```



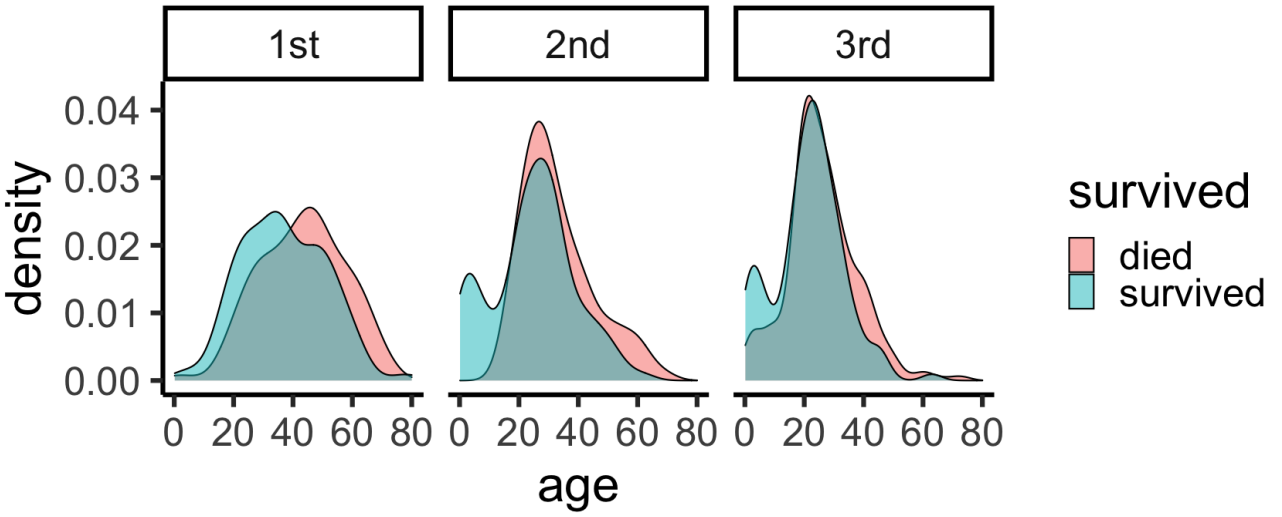
```
Titanicp %>% group_by(survived, sex) %>% count() %>%
  ggplot(aes(x = sex, y = n, fill = survived)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  labs(y = "", x = "Sex", fill = "Survival status")
```



```
Titanicp %>%
  ggplot() +
  aes(x = age, fill = survived) +
  geom_density(alpha = 0.5)
```




```
Titanicp %>% ggplot() +
  aes(x = age, fill = survived) +
  geom_density(alpha = 0.5) +
  facet_grid(~pclass)
```



It seems clear that there is some sort of a relationship between survival, sex, class and perhaps even age.

How do we model this?!

Logistic regression

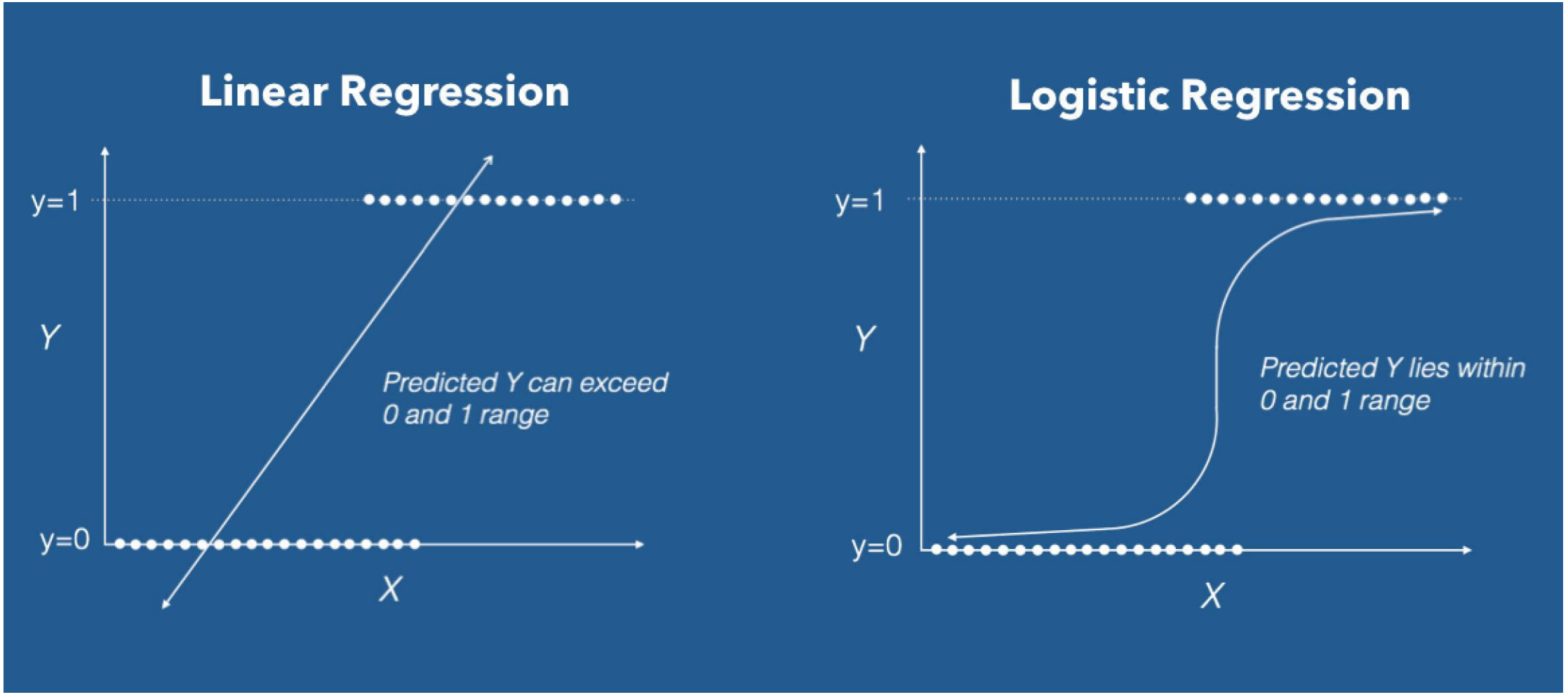
Linear regression

For **linear regression** we have

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon$$

where $\varepsilon \sim N(0, \sigma^2)$. In this case, conditional on the vector of predictor variables ($\boldsymbol{x}$'s), the dependent variable Y also follows a normal distribution, i.e. $Y_i | \boldsymbol{x}_i \sim N(\boldsymbol{x}_i' \boldsymbol{\beta}, \sigma^2)$

If the dependent variable Y is binary, i.e. $Y_i \in \{0, 1\}$, a linear regression doesn't work so well, and we need to use **logistic regression** instead.



Source: https://www.datacamp.com/community/tutorials/logistic-regression-R

Modelling binary data

We need to think what sort of a (conditional) distribution for $Y_i|\boldsymbol{x}_i$ makes more sense for binary data.

Since Y_i is either 0 or 1 it is natural to model it as a Bernoulli random variable:

$$Y_i|\boldsymbol{x}_i \sim \text{Bernoulli}(p(\boldsymbol{x}_i, \boldsymbol{\beta}))$$

where the probability that $Y_i = 1$ is given by some function of our predictors, $p(\boldsymbol{x}_i, \boldsymbol{\beta})$.

We need the probability $p(\boldsymbol{x}_i, \boldsymbol{\beta})$ to be in $[0, 1]$ and for it to depend on the linear combination of our predictors $\boldsymbol{x}_i'\boldsymbol{\beta}$ in some way.

A common choice is the **logistic function**,

$$p(\boldsymbol{x}_i, \boldsymbol{\beta}) = \frac{\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}_i'\boldsymbol{\beta})}.$$

A nice property of this choice is that,

- $\boldsymbol{x}_i'\boldsymbol{\beta} > 0$ implies that $p(\boldsymbol{x}_i, \boldsymbol{\beta}) > 0.5$, so $Y_i = 1$ is most likely
- $\boldsymbol{x}_i'\boldsymbol{\beta} < 0$ implies that $p(\boldsymbol{x}_i, \boldsymbol{\beta}) < 0.5$, so $Y_i = 0$ is most likely

Logistic regression

- A logistic regression model begins with,

$$Y_i|\boldsymbol{x}_i \sim \text{Bernoulli}\left(\frac{\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i'\boldsymbol{\beta})}\right).$$

- If we had a new observation vector $\boldsymbol{x}_0$ and we knew the $\boldsymbol{\beta}$ vector, we could calculate the probability that the corresponding $Y = 1$:

$$P(Y=1|\boldsymbol{x}_0) = \frac{\exp(\boldsymbol{x}_0'\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_0'\boldsymbol{\beta})}.$$

- If this probability is greater than 0.5, we would make the prediction $\hat{Y} = 1$, otherwise we would predict $\hat{Y} = 0$.
- BUT we don't know $\boldsymbol{\beta}$, we need to estimate the coefficient vector (just like in linear regression).
- Estimating $\hat{\boldsymbol{\beta}}$ can be done using iteratively reweighted least squares (there's no closed form solution).

Odds

We introduced **odds** in module 1, they are an alternative way of quantifying the probability of an event.

For some event E,

$$\text{odds}(E) = \frac{P(E)}{1 - P(E)}.$$

If we are told the odds of E are a to b, then

$$\text{odds}(E) = \frac{a}{b} = \frac{a/(a+b)}{b/(a+b)},$$

which implies $P(E) = a/(a+b)$.

Odds feature in **logistic regression**.

- Start by converting survival to 0/1 (numeric) variable

```
x = Titanicp %>% mutate(survived = ifelse(survived == "survived", 1, 0))
glimpse(x)
```

```
## Rows: 1,309
## Columns: 6
## $ pclass   <fct> 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1st, 1…
## $ survived <dbl> 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1,…
## $ sex      <fct> female, male, female, male, female, male,…
## $ age      <dbl> 29.0000, 0.9167, 2.0000, 30.0000, 25.0000…
## $ sibsp    <dbl> 0, 1, 1, 1, 1, 0, 1, 0, 2, 0, 1, 1, 0, 0,…
## $ parch    <dbl> 0, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
```

- We treat `survived` and `died` as successes and failures from a Bernoulli (binomial) distribution where the probability of success is given by a transformation of a linear model of the predictors.

Fit a logistic regression model

```
glm1 = glm(survived ~ pclass + sex + age, family = binomial, data = x)
summary(glm1)
```

```
##
## Call:
## glm(formula = survived ~ pclass + sex + age, family = binomial,
##     data = x)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6399  -0.6979  -0.4336   0.6688   2.3964
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.522074   0.326702  10.781  < 2e-16 ***
## pclass2nd   -1.280570   0.225538  -5.678 1.36e-08 ***
## pclass3rd   -2.289661   0.225802 -10.140  < 2e-16 ***
## sexmale     -2.497845   0.166037 -15.044  < 2e-16 ***
## age         -0.034393   0.006331  -5.433 5.56e-08 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Checking for significance

Before we start to interpret our model and make predictions, we might want to know if we can drop any of the variables from the model. Like in the linear regression context, this is equivalent to testing $H_0\colon \beta = 0$ against the alternative $H_0\colon \beta \neq 0$.

Example, let's test if the coefficient for age is significantly different to zero.

- Hypotheses: $H_0\colon\ \beta_{\text{age}} = 0$ vs $H_1\colon\ \beta_{\text{age}} \neq 0$
- Test statistic: $T = \frac{\hat{\beta}_{\text{age}} - \beta_{\text{age}}}{\text{SE}(\hat{\beta}_{\text{age}})} \sim N(0,1)$
- Observed test statistic: $t_0 = \frac{-0.034393}{0.006331} = -5.433$
- p-value: $2P(T \geq |t_0|) = 2P(Z \geq 5.433) < 0.0001$
- Conclusion: the p-value is very small (much smaller than 0.05) therefore we reject the null hypothesis and conclude that age is a significant predictor for survival.

Write down the fitted model

```
glm1
```

```
##
## Call:  glm(formula = survived ~ pclass + sex + age, family = binomial,
##     data = x)
##
## Coefficients:
## (Intercept)    pclass2nd    pclass3rd      sexmale
##     3.52207     -1.28057     -2.28966     -2.49784
##         age
##    -0.03439
##
## Degrees of Freedom: 1045 Total (i.e. Null);  1041 Residual
##   (263 observations deleted due to missingness)
## Null Deviance:        1415
## Residual Deviance: 982.5     AIC: 992.5
```

$$\text{logit}(p) = 3.5 - 1.3\,\text{pclass2nd} - 2.3\,\text{pclass3rd} - 2.5\,\text{sexmale} - 0.03\,\text{Age}$$

What's this logit function?

The **logit** function is our **link** from a linear combination of the predictors to the probability of the outcome being equal to 1.

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

- It's the log-odds!
- Our estimated coefficients are therefore interpreted as changes in the **log-odds**.
- I.e. we can write out fitted model as:

$$\log\left(\frac{p}{1-p}\right) = 3.5 - 1.3\,\text{pclass2nd} - 2.3\,\text{pclass3rd} - 2.5\,\text{sexmale} - 0.03\,\text{Age}$$

The logistic function (introduced earlier) is the inverse of the logit function.

Interpreting our coefficients

$$\log\left(\frac{p}{1-p}\right) = 3.5 - 1.3\,\text{pclass2nd} - 2.3\,\text{pclass3rd} - 2.5\,\text{sexmale} - 0.03\,\text{age}$$

- **Intercept**: the log-odds of survival for an individual travelling in 1st class who is female and aged zero years old.
- Holding sex and age constant, the `pclass2nd` coefficient represents the **difference** in the log-odds between someone travelling in 1st class and someone travelling in 2nd class. In this case, it's **negative**, so we're saying that your odds of survival were lower if you travelled in second class, relative to those who travelled in first class.
- Holding class and age constant, the `sexmale` coefficient represents the **difference** in the log-odds between males and females. It is **negative,** so we can say that if you were a male, your odds of survival were **lower** than if you were a female.
- The `age` coefficient is also negative, which implies that older people had lower odds of survival than younger people. Specifically, on average, for each additional year older you are, the log-odds of survival decreased by 0.03, holding class and sex constant.

What do our predictions mean?

$$\log\left(\frac{p}{1-p}\right) = 3.5 - 1.3\,\text{pclass2nd} - 2.3\,\text{pclass3rd} - 2.5\,\text{sexmale} - 0.03\,\text{age}$$

We can predict the log-odds for a newborn male travelling in first class:

- `pclass2nd = 0`, `pclass3rd = 0`, `sexmale = 1`, `age = 0`

$$\log\left(\frac{p}{1-p}\right) = 3.5 - 1.3 \times 0 - 2.3 \times 0 - 2.5 \times 1 - 0.03 \times 0 = 3.5 - 2.5 = 1$$

The log-odds of survival for a newborn male travelling in first class is estimated to be 1.

```
new_data = data.frame(pclass = "1st", sex = "male", age = 0)
predict(glm1, newdata = new_data, type = "link")
```

```
##        1
## 1.024229
```

Can we work out the estimated probability of survival for a newborn male travelling in first class?

$$
\begin{aligned}
\log\left(\frac{p}{1-p}\right) &= 1 \\
\left(\frac{p}{1-p}\right) &= \exp(1) \\
p &= \exp(1) - p\exp(1) \\
p + p\exp(1) &= \exp(1) \\
p &= \frac{\exp(1)}{1+\exp(1)} \approx 0.73
\end{aligned}
$$

```
new_data = data.frame(pclass = "1st", sex = "male", age = 0)
predict(glm1, newdata = new_data, type = "response")
```

```
##         1
## 0.7357956
```

Note that we've used the **logistic** function to transform back to obtain an estimate of the **probability** (from the output of our model which is an estimate of the log-odds).

Outputting your model coefficients

The **sjPlot** package has some nice functions for outputting regression models.

```
library(sjPlot)
tab_model(glm1, transform = NULL)
```

	survived		
Predictors	*Log-Odds*	*CI*	*p*
(Intercept)	3.52	2.90 – 4.18	**<0.001**
pclass [2nd]	-1.28	-1.73 – -0.84	**<0.001**
pclass [3rd]	-2.29	-2.74 – -1.85	**<0.001**
sex [male]	-2.50	-2.83 – -2.18	**<0.001**
age	-0.03	-0.05 – -0.02	**<0.001**
Observations	1046		
R^2 Tjur	0.376		

Without the `transform = NULL` parameter, it will exponentiate the coefficients:

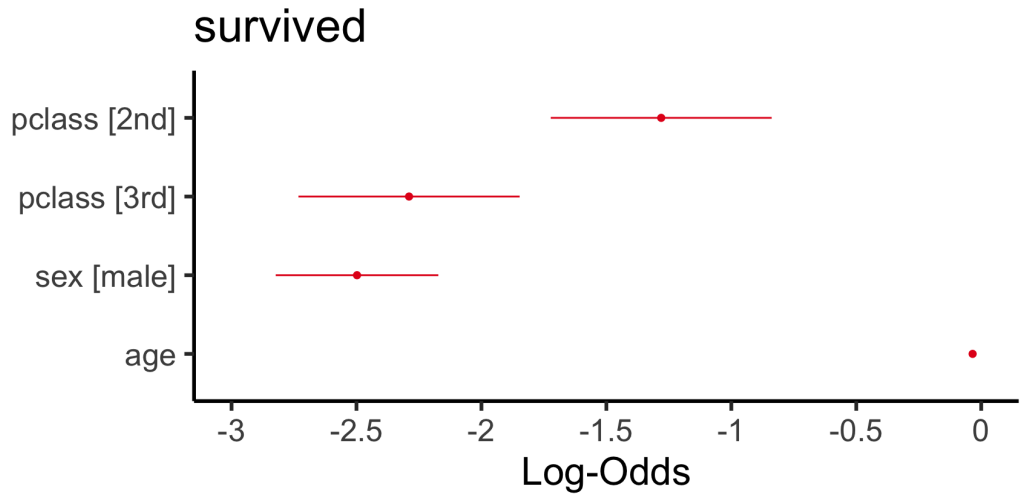
```
tab_model(glm1, show.ci = FALSE,
          show.r2 = FALSE)
```

	survived	
Predictors	*Odds Ratios*	*p*
(Intercept)	33.85	**<0.001**
pclass [2nd]	0.28	**<0.001**
pclass [3rd]	0.10	**<0.001**
sex [male]	0.08	**<0.001**
age	0.97	**<0.001**
Observations	1046	

Visualising your model coefficients

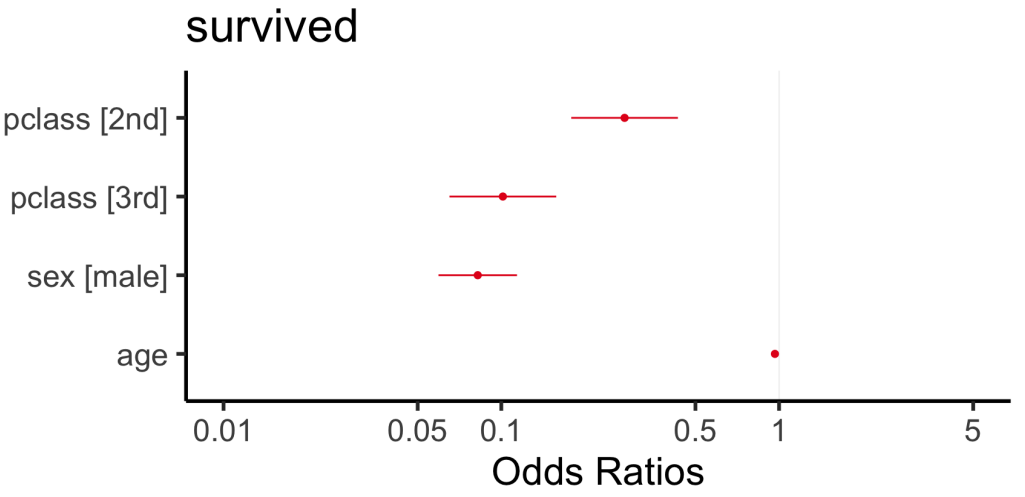
Log-odds scale

```
plot_model(glm1, transform = NULL)
```



Odds scale

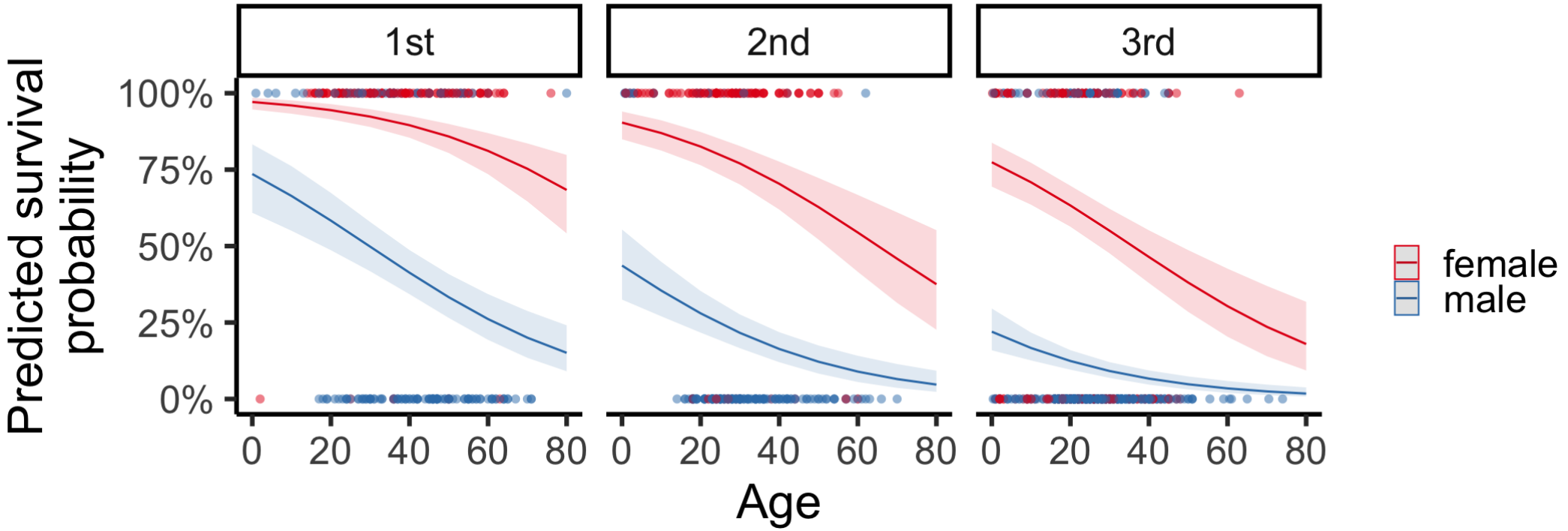
```
plot_model(glm1)
```



Note the coefficient for `age` is on a different scale to the categorical variables.

Visualising predictions

```
plot_model(glm1, type = "pred", terms = c("age", "sex", "pclass"), show.data = TRUE) +
  labs(title = "", y = "Predicted survival\nprobability", x = "Age", colour = "")
```



Evaluating performance

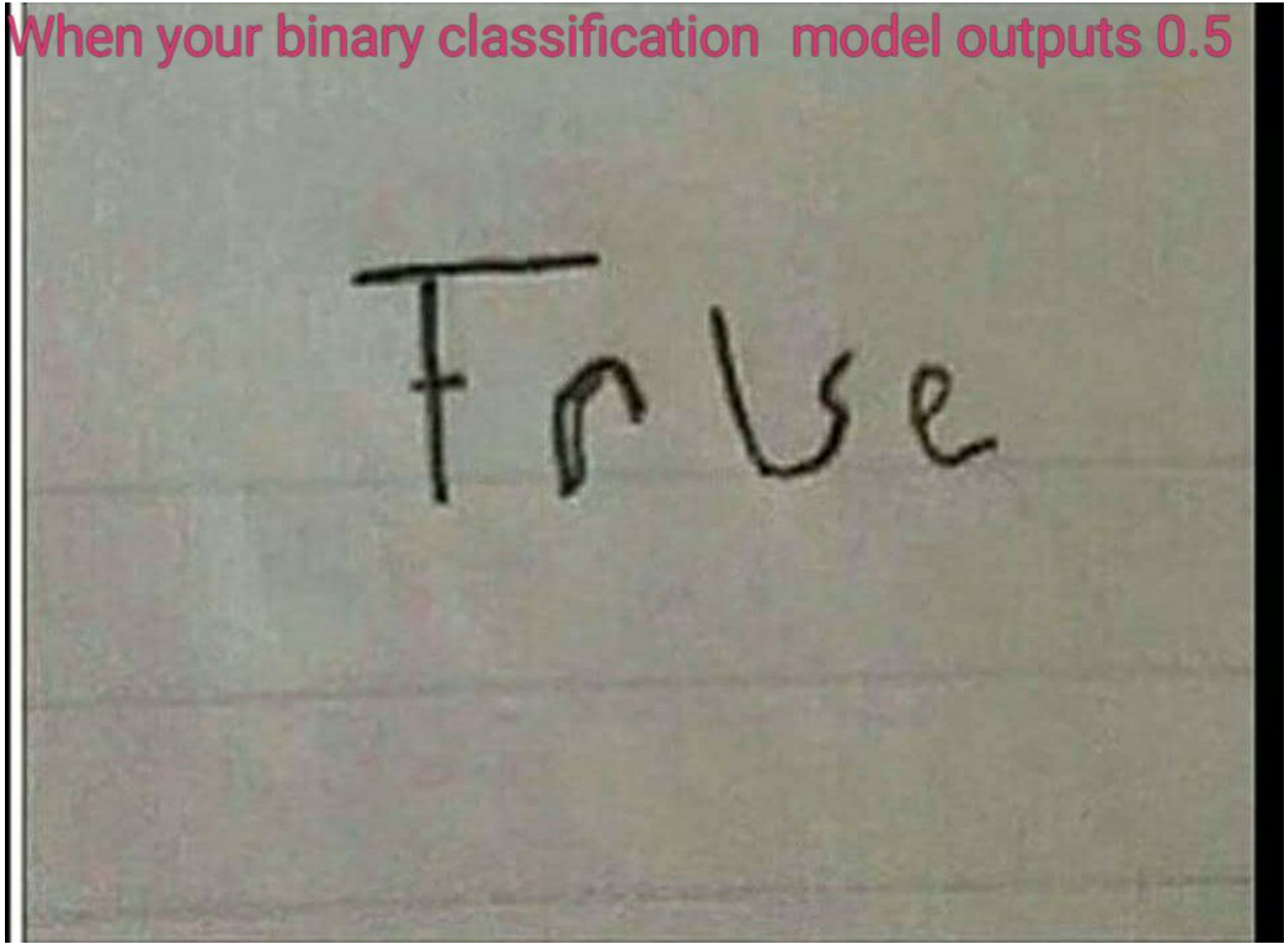
Making predictions

We can make predictions by rounding our predicted probability to 0 or 1.

```
x = x %>% drop_na() %>%
  mutate(pred_prob = predict(glm1, type = "response"),
         pred_surv = round(pred_prob))
head(x, n = 10)
```

```
##    pclass survived    sex     age sibsp parch pred_prob pred_surv
## 1     1st        1 female 29.0000     0     0 0.9258533         1
## 2     1st        1   male  0.9167     1     2 0.7296211         1
## 3     1st        0 female  2.0000     1     2 0.9693290         1
## 4     1st        0   male 30.0000     1     2 0.4981081         0
## 5     1st        0 female 25.0000     1     2 0.9347616         1
## 6     1st        1   male 48.0000     0     0 0.3482715         0
## 7     1st        1 female 63.0000     1     0 0.7949948         1
## 8     1st        0   male 39.0000     0     0 0.4213810         0
## 9     1st        1 female 53.0000     2     0 0.8454345         1
## 10    1st        0   male 71.0000     0     0 0.1950240         0
```

Predicted probabilities close to 0.5 are hard to classify!!



Evaluating (in-sample) performance

How many passengers did we correctly classify?

Resubstitution error rate

The **resubstitution error rate** is the proportion of observations we predict **incorrectly** when we try to predict all the points we used to fit the model.

$$\frac{1}{n}\sum_{i=1}^{n}(y_i \neq \hat{y}_i)$$

```
mean(x$survived != x$pred_surv)
```

```
## [1] 0.2151052
```

We failed to correctly classify 21.5% of the observations.

Evaluating (in-sample) performance

Confusion matrix

We can examine how our model predicted all the data points, using `confusionMatrix` from the **caret** package. Note, that we are required to put in factor inputs.

```
library(caret)
confusion.glm = confusionMatrix(
  data = as.factor(x$pred_surv),
  reference = as.factor(x$survived))
confusion.glm$table
```

```
##           Reference
## Prediction   0   1
##          0 520 126
##          1  99 301
```

Looking at the table output and reading vertically, we can assess model performance.

- Out of the 520+99=619 deaths in our data set, the model successfully predicts 520.
- Out of the 126+301=427 survivors in our data set, the model correctly predicts 301.

Evaluating (in-sample) performance

```
confusion.glm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 520 126
##          1  99 301
##
##                Accuracy : 0.7849
##                  95% CI : (0.7587, 0.8094)
##     No Information Rate : 0.5918
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.5504
##
##  Mcnemar's Test P-Value : 0.08304
##
##             Sensitivity : 0.8401
##             Specificity : 0.7049
##          Pos Pred Value : 0.8050
##          Neg Pred Value : 0.7525
##              Prevalence : 0.5918
```

The **accuracy** is 1 minus the resubstitution error rate.

Some of the other performance metrics will be familiar to you from module 1.

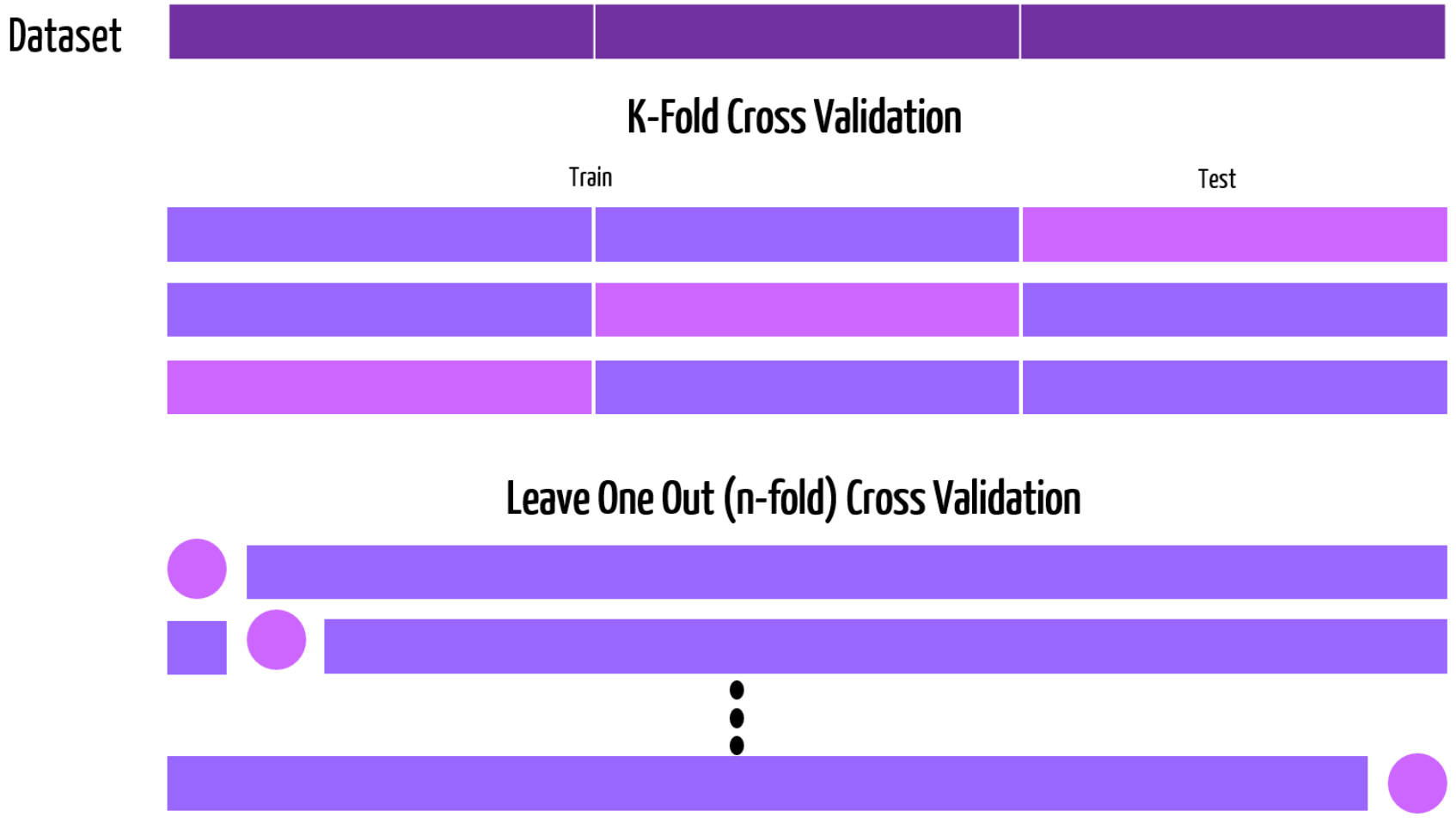
- sensitivity
- specificity
- positive predictive value
- negative predictive value

Building reliable and accurate models can be tricky

Evaluating out of sample performance

- Often, we want to see how well our model can predict new data points. However, it is often impossible to get completely new data.
- Like with linear regression we can perform k-fold cross validation to evaluate out of sample performance.
- We split our data into training and testing sets to evaluate performance, treating the testing data as new data points.
- For k-fold CV, we split our data into k-folds. The first fold is treated as a testing set, and the method is fit on the remaining $k-1$ folds.
- The misclassification error rate is then computed on the observations in the held-out fold.
- This procedure is repeated k times; each time, a different group of observations is treated as a testing set.
- The **CV error rate** is then calculated as the average of these k error rates.

How many folds?



How many folds?

- Larger k can take longer to run (computationally more expensive)
- Larger k reduces bias in predictions (you use more data to build your training model)
- If k is too small you might not have enough training observations to build a sensible model
- But there's a variance issue with the test error rate for models with large k. As k increases the variance increases (using almost identical data each time to make predictions, which doesn't give a very precise assessment of how independent models on new data would perform). For more details see James, Witten, Hastie, et al. (2017; p. 183).

Generally, k between 5 and 10 is selected.

Cross validation for Titanic data

```
x_full = x %>% drop_na() %>%
  select(pclass, survived, sex, age)
nrow(x_full)
```

```
## [1] 1046
```

```
nrow(x_full)/5
```

```
## [1] 209.2
```

```
fold_id = c(1, rep(1:5, each = 209))
table(fold_id)
```

```
## fold_id
##   1   2   3   4   5
## 210 209 209 209 209
```

```
x_full$fold_id = sample(fold_id, replace = FALS
```

```
head(x_full)
```

```
##   pclass survived    sex     age fold_id
## 1    1st        1 female 29.0000       3
## 2    1st        1   male  0.9167       4
## 3    1st        0 female  2.0000       4
## 4    1st        0   male 30.0000       5
## 5    1st        0 female 25.0000       5
## 6    1st        1   male 48.0000       4
```

Cross validation for Titanic data

```
cv_error = vector("numeric", length = 5)
for(j in 1:5){
  train = x_full %>% filter(fold_id != j)
  fit = glm(survived ~ pclass + sex + age, data = train)
  test = x_full %>% filter(fold_id == j)
  pred = round(predict(fit, newdata = test, type = "response"))
  cv_error[j] = mean(pred != test$survived)
}
cv_error
```

```
## [1] 0.2333333 0.2344498 0.2009569 0.2822967 0.1578947
```

```
mean(cv_error)
```

```
## [1] 0.2217863
```

```
1-mean(cv_error) # accuracy
```

```
## [1] 0.7782137
```

Cross validation for Titanic data using caret

```r
library(caret)
train(factor(survived) ~ pclass + sex + age,
      data = x_full,
      method = "glm",
      family = "binomial",
      trControl = trainControl(
        method = "cv", number = 5,
        verboseIter = FALSE
      ))
```

```
## Generalized Linear Model
##
## 1046 samples
##    3 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 837, 836, 838, 837, 836
## Resampling results:
##
##   Accuracy   Kappa
##   0.7820303  0.5441975
```

References

Baumer, B. S., D. T. Kaplan, and N. J. Horton (2017). *Modern Data Science with R*. Boca Raton: Chapman and Hall/CRC. URL: https://mdsr-book.github.io/index.html.

James, G., D. Witten, T. Hastie, and R. Tibshirani (2017). *An Introduction to Statistical Learning: With Applications in R*. New York: Springer. URL: https://www-bcf.usc.edu/~gareth/ISL/.

Jed Wing, M. K. C. from, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, et al. (2018). *caret: Classification and Regression Training*. R package version 6.0-80. URL: https://CRAN.R-project.org/package=caret.