Predicting Binary Events

In this section we will develop methods for predicting binary events. These are events that either occur or don't occur. For example, will a certain customer place a new order in the next month? Clearly, this will either happen or not happen. In order to do this we need to modify our approach from the one we used when discussing linear predictive models. These were models of the mean of a continuous variable, e.g., log price. What we need to do now is to model the mean of a binary variable, e.g., a *probability*. If the predicted probability of the event occurring is high we will predict that it is likely that the event will occur. Since we are modelling probabilities we need to make sure that the predicted probabalities are constrained to be between zero and one. The most popular model for doing this is the **Logit Model**.

All data and code for the material below can be found here:

```
usethis::use_course('https://www.dropbox.com/sh/mln5gafmhu0iayg/AAAj9lbDiRVA9WYo6NL5T
        tqDa?dl=1')
```

Logit Models

Suppose we want to understand what drives some users to click on an online banner ad. In this case we can let Y denote the binary event with

$$Y_i = \begin{cases} 1 & \text{if user i clicks on the ad,} \\ 0 & \text{otherwise.} \end{cases}$$

Suppose the online advertiser randomly changes the image shown on the add from the standard image $X = 0$ to a new test image $X = 1$. We want to predict by how the test image changes the click-rate. Suppose we have data (Y_i, X_i) from a large set of users. Some users were exposed to the standard image while others were exposed to the new image. A logit model will model the click rate as

$$\Pr(Y_i = 1|X_i) = \frac{\exp\{\beta_0 + \beta_X X_i\}}{1 + \exp\{\beta_0 + \beta_X X_i\}},$$

where β_0 and β_X are unknown parameters to be determined by the historical data. Let's make some quick observations. First, this model constrains the predicted probabilities to be between zero and one. Second, we can use this model to see what the change in click-rate will be when we change the image. This is easy - just plug in:

$$\Pr(\text{click}|\text{standard image}) = \Pr(Y_i = 1|X_i = 0) = \frac{\exp\{\beta_0\}}{1 + \exp\{\beta_0\}}.$$

Using the historical data (and the power of R!) we can determine β_0 - we will see how to do this below. Similarly,

$$\Pr(\text{click}|\text{new image}) = \Pr(Y_i = 1|X_i = 1) = \frac{\exp\{\beta_0 + \beta_X\}}{1 + \exp\{\beta_0 + \beta_X\}}.$$

Therefore we get

$$\Delta(\text{click rate}) \equiv \frac{\exp\{\beta_0 + \beta_X\}}{1.0 + \exp\{\beta_0 + \beta_X\}} - \frac{\exp\{\beta_0\}}{1 + \exp\{\beta_0\}}.$$

Calibrating Logit Models on Data

The standard method used to calibrate the β weights of a logit model is Maximum Likelihood. This is based on choosing β to maximize the total probability of the training data. Suppose the training data is $\{Y_i, X_i\}_{i=1}^N$ where X_i are the K predictor variables for person i. Define

$$\mu_i \equiv \sum_{k=1}^{K} \beta_k X_{ik}.$$

The probability of observing Y_i depends on what Y_i is - 1 or 0:

$$\Pr(Y_i = 1|X_i) = \frac{\exp\{\mu_i\}}{1.0 + \exp\{\mu_i\}},$$
$$\Pr(Y_i = 0|X_i) = \frac{1}{1 + \exp\{\mu_i\}}.$$

Therefore we can write the probability of observing Y_i as

$$L_i(\beta) \equiv \left(\frac{\exp\{\mu_i\}}{1.0 + \exp\{\mu_i\}}\right)^{Y_i} \times \left(\frac{1}{1 + \exp\{\mu_i\}}\right)^{1-Y_i}$$

If - in addition - we assume that Y_i is generated independently of all other Y_i's, then we can write the probability of observing the full training sample as

$$L(\beta) = \prod_{i=1}^{N} L_i(\beta).$$

The method of Maximum Likelihood uses a numerical algorithm to find the β that maximizes $L(\beta)$. Technically, since it turns out to be easier to maximize the log of L, it will maximize $\log L(\beta)$.

Case Study: Titanic

You are about to board the Titanic - would you rather be Jack or Rose? Well, of course we know what happened in the movie - but was this representative of the actual survival rates of rich young women vs. poor young men? And what about old men or women? Let's build a logit model to predict survival probabilities for different traveller segments on the Titanic.

First, we need the actual passenger data for Titanic with information on demographics and survival outcomes. It turns out that this exists - although it is incomplete: We only have survival information on 1309 of the more than

2000 passengers and complete demographics only for 1046 passengers. However, let's see what we can do with what we have. First, load the data and see what variables we have:

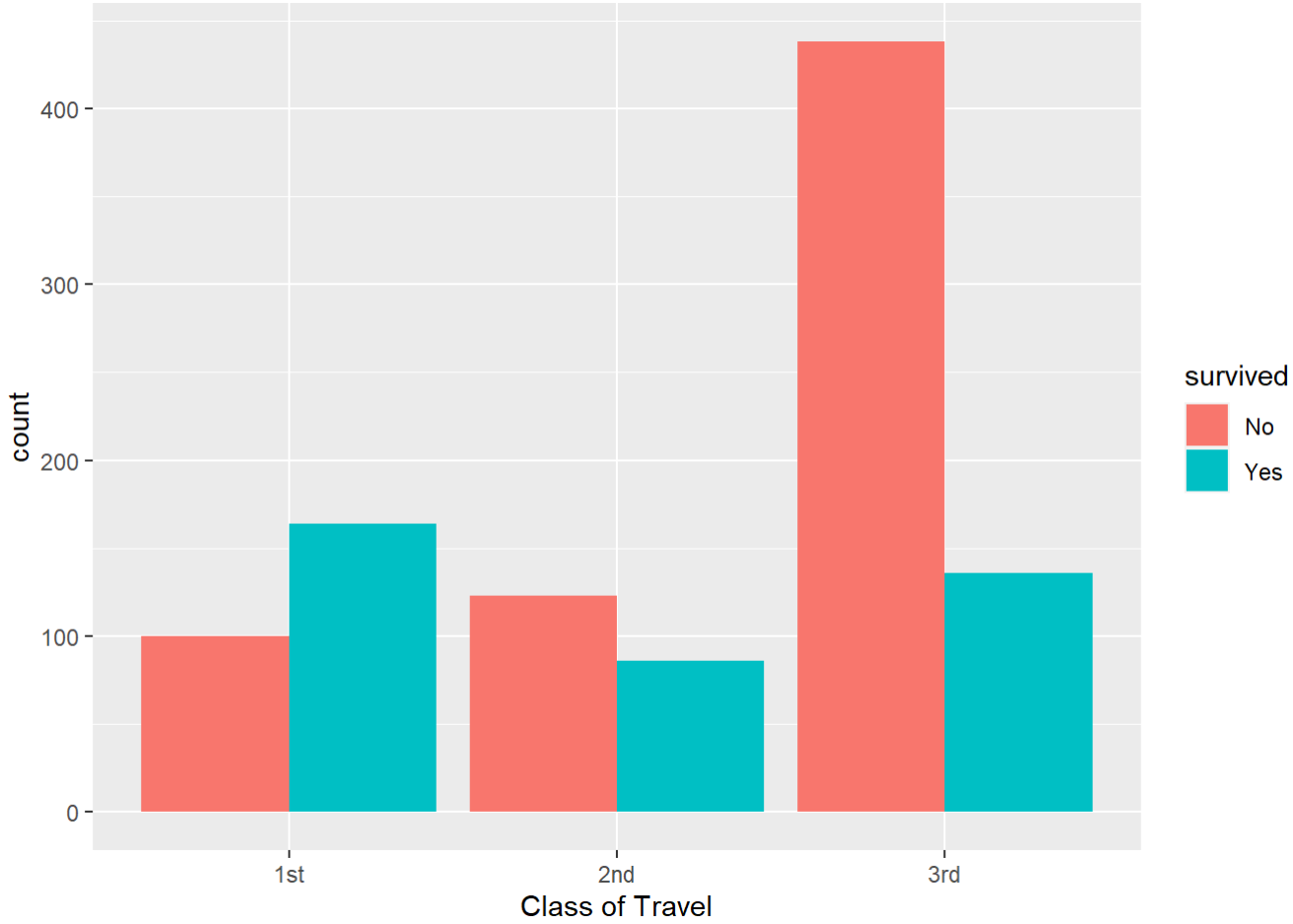
```
## load libraries
library(tidyverse)
library(glmnet)

## get the data
train <- read_csv('data/titanic_train.csv')
validation <- read_csv('data/titanic_validation.csv')

## what's in the data?
head(train)
```

```
## # A tibble: 6 x 15
##      id pclass survived name  sex     age sibsp parch  ticket  fare cabin
##   <dbl> <chr>  <chr>    <chr> <chr> <dbl> <dbl> <dbl>   <dbl> <dbl> <chr>
## 1   472 2nd    Yes      "Kel~ fema~    45     0     0  223596 13.5  <NA>
## 2   370 2nd    No       "Cha~ fema~    29     1     0      NA 26    <NA>
## 3   889 3rd    No       "Joh~ male     34     0     0 3101264  6.50 <NA>
## 4   628 3rd    No       "And~ fema~     9     4     2  347082 31.3  <NA>
## 5   319 1st    No       "Wil~ male     NA     0     0  113510 35    C128
## 6  1115 3rd    No       "Pea~ male     NA     0     0  343271  7    <NA>
## # ... with 4 more variables: embarked <chr>, boat <chr>, body <dbl>,
## #   home.dest <chr>
```

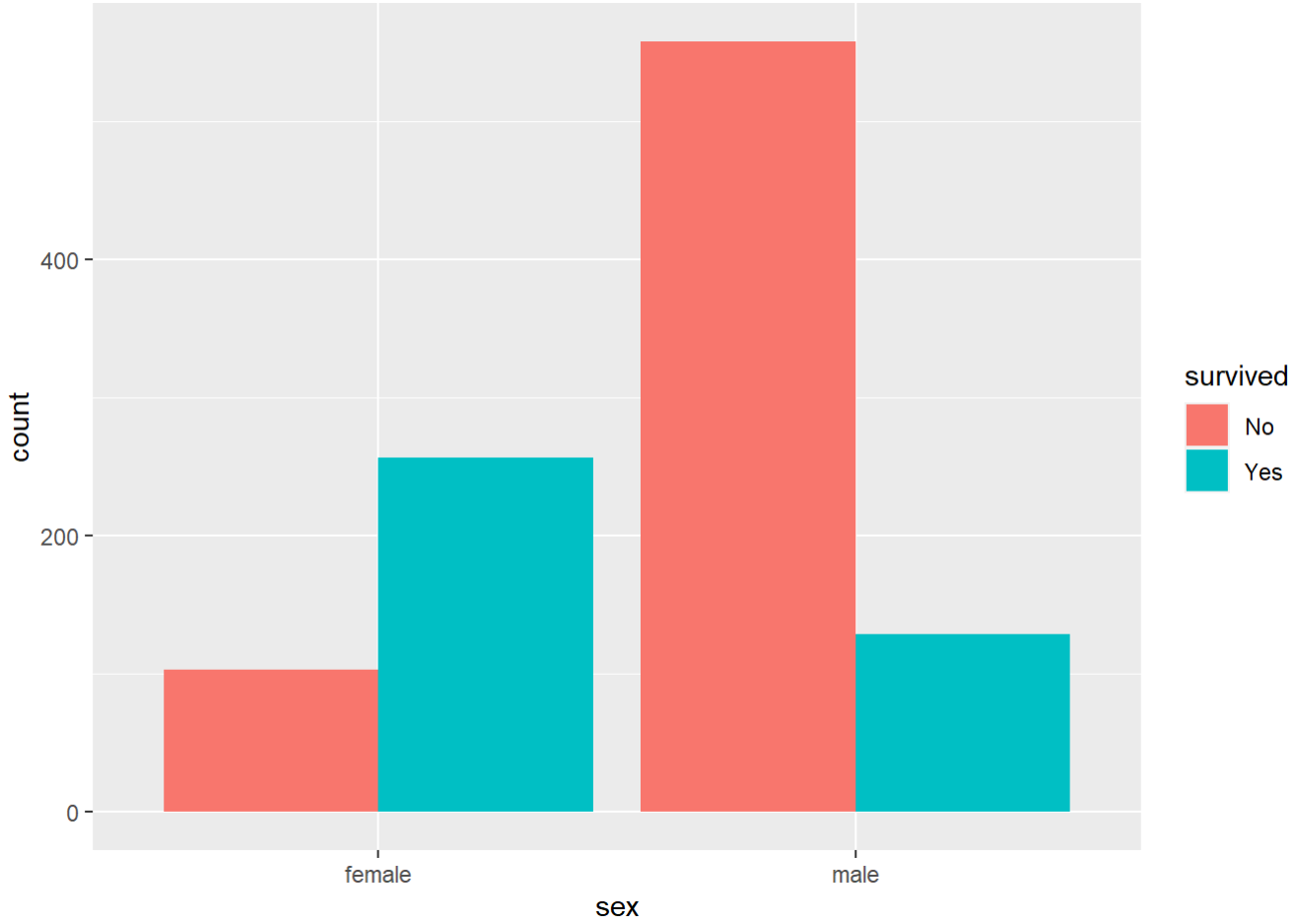
Ok - looks like we have class of travel, survival outcome, name, gender, age. There is also information on number of siblings/spouses aboard (**sibsp**) and number of parents/childen aboard (**parch**). You can also see ticket number and price, cabin number, where the traveller embarked and home destination. For survivors you can see which lifeboard they were on and for non-survivors you can see whether the body was recovered. The data has been split into 1047 passengers in the training sample and 262 in the validation sample.

To see what might matter for survival, let's quickly make some bar charts focusing on class of travel, age and gender:

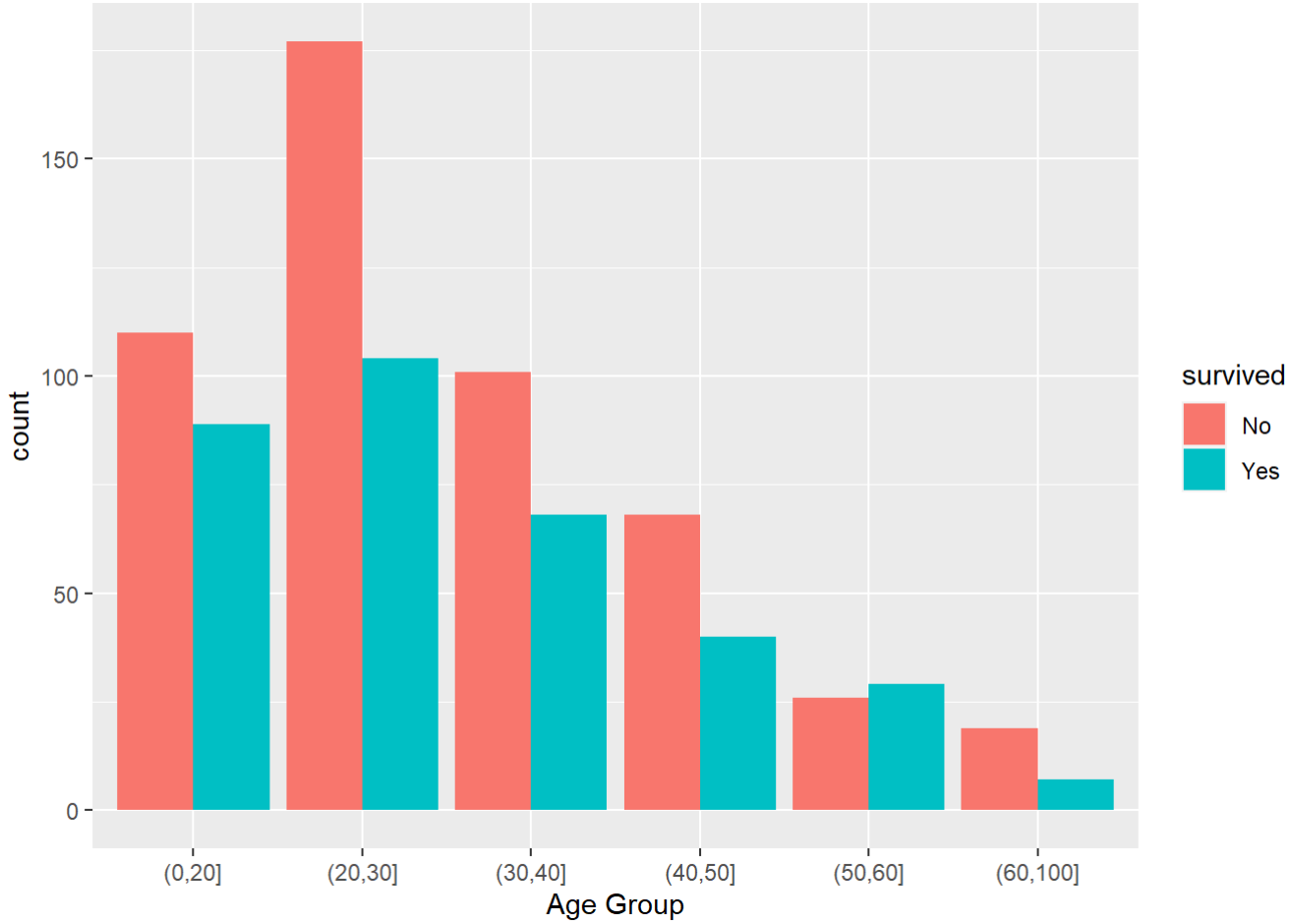
```
## by class of travel
train %>%
  ggplot(aes(x=pclass,fill=survived)) + geom_bar(position='dodge') + xlab('Class of T
        ravel')
```




```
## by gender
train %>%
  ggplot(aes(x=sex,fill=survived)) + geom_bar(position='dodge')
```




```
## by age
train %>%
  filter(!age=='NA') %>%
  mutate(age.f=cut(age,breaks=c(0,20,30,40,50,60,100))) %>%
  ggplot(aes(x=age.f,fill=survived)) + geom_bar(position='dodge') + xlab('Age Group')
```



Hmm…it's pretty clear that passengers travelling on 3rd class didn't fare well compared to 1st and 2nd class passengers, and 2nd class passengers were worse off compared to 1st class. We also see a huge gender effect - many more female passengers survived. It is a bit harder to judge the age effect. However, it seems like the survival rate is much lower for 20-30 year olds than any of the other categories.

Based on these summaries, it seems reasonable to try out a logit model with class of travel, gender and age as predictor variables. The syntax for setting up a logit model is more or less the same as the one we used for linear predictive models:

```
## remove observations with missing age, define age groups and survival outcome
train <- train %>%
  filter(!age=='NA') %>%
  mutate(age.f=cut(age,breaks=c(0,20,30,40,50,100)),
         lsurv = survived=='Yes',
         pclass = factor(pclass),
         sex = factor(sex))

validation <- validation %>%
  filter(!age=='NA') %>%
  mutate(age.f=cut(age,breaks=c(0,20,30,40,50,100)),
         lsurv = survived=='Yes',
         pclass = factor(pclass),
         sex = factor(sex))

## define logit model: class of travel, gender, age group ----------------------
logitTitanicA <- glm(lsurv~pclass+sex+age.f,
                     data=train,
                     family=binomial(link="logit"))
```

This sets up a logit model for predicting the event that **survived** is equal to **Yes**. You can see the individual effects by looking at the results:

```
summary(logitTitanicA)
```

```
##
## Call:
## glm(formula = lsurv ~ pclass + sex + age.f, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2703  -0.7083  -0.4575   0.6751   2.4494
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     2.8952     0.3043   9.514  < 2e-16 ***
## pclass2nd      -1.2294     0.2527  -4.866 1.14e-06 ***
## pclass3rd      -2.2532     0.2484  -9.070  < 2e-16 ***
## sexmale        -2.4493     0.1858 -13.186  < 2e-16 ***
## age.f(20,30]   -0.3971     0.2401  -1.654 0.098131 .
## age.f(30,40]   -0.4714     0.2777  -1.698 0.089568 .
## age.f(40,50]   -1.1414     0.3273  -3.487 0.000488 ***
## age.f(50,100]  -1.1430     0.3703  -3.087 0.002024 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1129.41  on 837  degrees of freedom
## Residual deviance:  792.11  on 830  degrees of freedom
## AIC: 808.11
##
## Number of Fisher Scoring iterations: 4
```

In a moment we will use R to automatically generate preditions for different traveller segments, but before we do that it might be instructive to see how to do this manually. Suppose we wanted the predicted survival probability for 25 year old males, travelling on 3rd class. We can piece together the β effects for this segment by simply picking out the relevant ones from the table above - remembering to always include the intercept:

$$2.8952 - 2.2532 - 2.4493 - 0.3971 \approx -2.20$$

Then - applying the logit formula - we get a predicted survival rate of

$$\Pr(\text{survival}|\text{25 year, 3rd class, male}) = \exp(-2.20)/(1.0 + \exp(-2.20)) \approx 10\%$$

Those are bad odds - only 1 in 10 survived. Let's calculate this for all segments. To do this we create a new data frame where each row is a segment for which we want a predicted survival rate:

```
## create prediction array
predDF <- expand.grid(pclass=levels(train$pclass),
                      sex=levels(train$sex),
                      age.f=levels(train$age.f))

## make predictions
predDF$Prob <- predict(logitTitanicA,
                       newdata = predDF,
                       type = "response")
```

The data frame **predDF** contains all possible combinations of our three predictor variables. Sometimes - when you have many predictor variables - you might not be interested in all combinations and you would only use a subset of all possibilities. Here we will use all of them (30 in total). The second command use the logit model to predict the probability ("response") for each of the rows in **predDF**.

Let's check that our manual calculation above was correct:

```
predDF %>%
  filter(sex=='male', pclass=='3rd')
```

```
##   pclass  sex    age.f       Prob
## 1    3rd male   (0,20] 0.14096462
## 2    3rd male  (20,30] 0.09935783
## 3    3rd male  (30,40] 0.09290324
## 4    3rd male  (40,50] 0.04979937
## 5    3rd male (50,100] 0.04972038
```

These are the predicted survival rates for males travelling on 3rd class. We can see that we got it right above.

At this point it might be fun to visualize the predictions for all segments:

```
## plot predictions
predDF %>%
  ggplot(aes(x=age.f,y=Prob,group=sex,color=sex)) + geom_line(linetype='dotted') +
  geom_point() +
  ylab('Survival Probability') + xlab('Age Group') +
  facet_wrap(~pclass) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title=element_text(size=14))+
  ggtitle('Predicted Survival Probability by Class of Travel, Gender and Age')
```

What would you conclude about demographics and survival rates based on this?

Is the model above reasonable? Maybe - but one objection we could raise is the way we have modelled the impact of gender and class of travel. Notice that by design, the effect of gender is the same for every class of travel. This is a consequence of the **additivity** assumption implicit in the model above. Specifically, the difference in the sum of β's for males and females on 1st class is identical to the difference in the sum of β's for males and females on 3rd class. We can change this by adding an **interaction** in the model:

```
logitTitanicB <- glm((survived=='Yes')~pclass*sex+age.f,
                     data=train,
                     family=binomial(link="logit"))

summary(logitTitanicB)
```

```
##
## Call:
## glm(formula = (survived == "Yes") ~ pclass * sex + age.f, family = binomial(link =
"logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8218  -0.6939  -0.5519   0.5027   2.2748
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)         4.4107     0.6350   6.946 3.75e-12 ***
## pclass2nd          -2.0623     0.6902  -2.988 0.002809 **
## pclass3rd          -4.2510     0.6375  -6.668 2.59e-11 ***
## sexmale            -4.1950     0.6192  -6.774 1.25e-11 ***
## age.f(20,30]       -0.4481     0.2346  -1.910 0.056086 .
## age.f(30,40]       -0.5038     0.2764  -1.823 0.068309 .
## age.f(40,50]       -1.2082     0.3475  -3.477 0.000507 ***
## age.f(50,100]      -1.2865     0.4129  -3.116 0.001832 **
## pclass2nd:sexmale   0.7453     0.7503   0.993 0.320571
## pclass3rd:sexmale   2.7342     0.6665   4.102 4.09e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1129.41  on 837  degrees of freedom
## Residual deviance:  758.17  on 828  degrees of freedom
## AIC: 778.17
##
## Number of Fisher Scoring iterations: 6
```

The gender effect now depends on class of travel: In 1st class, the gender effect for males and females is -4.19. For 3rd class it is -4.19+2.73 = -1.46. What does this mean? It means that the relative difference in survival rates for males and females is much bigger in 1st class than 3rd class. We can see this better by visualizing the implied survival rates:

```
predDF$Prob <- predict(logitTitanicB,
                       newdata = predDF,
                       type = "response")

predDF %>%
  ggplot(aes(x=age.f,y=Prob,group=sex,color=sex)) + geom_line(linetype='dotted') +
  geom_point() +
  ylab('Survival Probability') + xlab('Age Group') +
  facet_wrap(~pclass) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title=element_text(size=14))+
  ggtitle('Predicted Survival Probability by Class of Travel, Gender and Age')
```

Compare this to the previous plot.

One thing we haven't done is to see how well our model actually predicts survival rates for the "validation" passengers. We can use the predict function on the test data to do this:

```
validation <- validation %>%
  mutate(ProbRepA = predict(logitTitanicA,newdata = validation,type = "response"),
         ProbRepB = predict(logitTitanicB,newdata = validation,type = "response"),
         SurvPredA = ProbRepA > 0.5,
         SurvPredB = ProbRepB > 0.5)
```

We first get the predicted survival probabilities for each validation passenger and then use them to predict survival if the probability is above 50% (otherwise no survival is predicted). We can now tally up our predictions and see how we did. This is usefully presented in a table called a *confusion matrix*. This table simply counts how many times we got a prediction right or wrong:

```
TabPredA <- table(validation$lsurv,validation$SurvPredA)
TabPredB <- table(validation$lsurv,validation$SurvPredB)
```

Let's look at the first table:

```
TabPredA
```

```
##
##         FALSE TRUE
##   FALSE    98   20
##   TRUE     24   66
```

Remember that "TRUE" means survival. There were a total of 90 validation passengers who survived. This is the sum of the second row in the table. The columns indicate our predictions of the fate of these passengers. Model A correctly predicted that 66 of these 90 passengers survived. It wrongly predicted that 24 of them died.

Similarly - looking at the first row - there were a total of 118 validation passengers who died. We correcly predict 98 of these deaths. We can think of the diagonal of the table - normalized by the total - as the overall accuracy of the model. This is

```
AccuracyA <- (TabPredA[1,1]+TabPredA[2,2])/sum(TabPredA)
AccuracyA
```

```
## [1] 0.7884615
```

Let's compare this to the second model:

```
AccuracyB <- (TabPredB[1,1]+TabPredB[2,2])/sum(TabPredB)

TabPredB
```

```
##
##          FALSE TRUE
##   FALSE   111    7
##   TRUE     33   57
```

```
AccuracyB
```

```
## [1] 0.8076923
```

The second model has higher accuracy. It achieves this by being better at predicting non-survival compared to Model A. Note that Model B is slightly worse than A at predicting survival.

Python Version

There are several Python libraries available for training logit classifiers. Here we will use `scikit-learn`. We load the required libraries and the data. Furthermore, knowing what we know from above, we also drop passengers with missing age information:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

## get titanic.data ------------------------------------------------------------
train = pd.read_csv('data/titanic_train.csv')
validation = pd.read_csv('data/titanic_validation.csv')
train = train.dropna(subset=['age'])
validation = validation.dropna(subset=['age'])

allDF = pd.concat([train, validation])
nTrain = train.shape[0]
nAll = allDF.shape[0]
```

Next we define age bins like above and create the feature matrix using the `get_dummies` function from pandas for both the training and validation data:

```
allDF['age_f'] = pd.cut(allDF["age"],bins=[0,20,30,40,50,100])

Xdf = allDF[['pclass','sex','age_f']].copy()
X = pd.get_dummies(data=Xdf, drop_first=True)
y = allDF['survived']

X_train = X[0:nTrain]
y_train = y[0:nTrain]
X_valid = X[nTrain:nAll]
y_valid = y[nTrain:nAll]
```

Now we are ready to train our classifier. We initialize a `LogisticRegression` object and specify that we wish to train the model without any penalty/regularization term (the default is an L2 penalty so we need to override this). We then train the model using only the training data:

```
model = LogisticRegression(penalty = "none")
model.fit(X_train, y_train)
```

```
## LogisticRegression(penalty='none')
```

We can inspect the trained weights for the classifier:

```
# weights
model.intercept_
```

```
## array([2.89516993])
```

```
model.coef_
```

```
## array([[-1.22938318, -2.25319286, -2.44926147, -0.39705775, -0.47139706,
##         -1.14135252, -1.14302393]])
```

This is identical to what we got above in the R version. Finally, we can predict on the validation data and summarize the performance:

```
## predictions on validation
y_pred = model.predict(X_valid)
y_pred_prob = model.predict_proba(X_valid)

## confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
## [[98 20]
##  [24 66]]
```

```
cr = classification_report(y_valid, y_pred)
print(cr)
```

```
##               precision    recall  f1-score   support
##
##           No       0.80      0.83      0.82       118
##          Yes       0.77      0.73      0.75        90
##
##     accuracy                           0.79       208
##    macro avg       0.79      0.78      0.78       208
## weighted avg       0.79      0.79      0.79       208
```

The confusion matrix is identical to the one we got above using R. Here we have also calculated a set of additional performance metrics using the `classification_report` function. These metrics are useful when more detailed insights into the model's performance is needed. Precision is the fraction of predicted positives (here survivals) that are correct while recall is the fraction of actual positives that the model predicts correctly. These metrics can be relevant when prediction errors have differential costs to the decision maker. For example, if false positives (an actual dead person is predicted to have survived) are very costly, then precision should be monitored. On the hand, if false negatives (an actual survivor is predicted to have died) are more costly, then recall is important. The F1 score is a balance of these two considerations as is defined as 2*(precision*recall)/(precision + recall).

Case Study: Donors Choose

Let's return to the DonorsChoose data. The objective is to try to predict which projects gets funded. Here we are using a subset of the full data. There are 60,000 projects in the training data and we will use this data to predict funding success of the 40,000 projects in the test data. Let's read in the data and apply a few transformations:

```
library(tidyverse)
library(broom)
library(forcats)
library(lubridate)

## get data

projectsTrain <- read_csv('data/projects_train.csv')
projectsVal <- read_csv('data/projects_validation.csv')

projectsTrain <- projectsTrain %>%
  mutate(cost.group = cut(total_price_excluding_optional_support,
                          breaks = c(0,200,300,400,500,600,700,800,900,1000,1500,2000
        ,500000)),
         month = as.character(month(date_posted,label = T)),
         nStudents = cut(students_reached,
                         breaks=c(0,10,20,30,50,100,200,10000),
                         include.lowest = T))

projectsVal <- projectsVal %>%
  mutate(cost.group = cut(total_price_excluding_optional_support,
                          breaks = c(0,200,300,400,500,600,700,800,900,1000,1500,2000
        ,500000)),
         month = as.character(month(date_posted,label = T)),
         nStudents = cut(students_reached,
                         breaks=c(0,10,20,30,50,100,200,10000),
                         include.lowest = T))
```

Next, we specify a logit model including the main characteristics of each project:

```
logitProjects <- glm(funding_status=='completed'~
                       primary_focus_subject +
                       grade_level +
                       poverty_level +
                       resource_type +
                       eligible_double_your_impact_match +
                       cost.group +
                       nStudents +
                       eligible_almost_home_match +
                       month +
                       teacher_teach_for_america +
                       teacher_prefix +
                       school_charter +
                       school_magnet +
                       school_state,
                     data=projectsTrain,
                     family=binomial(link="logit"))

logitProjectsStats <- tidy(logitProjects)
```

Let’s see how well we predict the validation projects:

```
logitProjectsStats <- tidy(logitProjects)

projectsVal$ProbCompleteLogit=predict(logitProjects,newdata = projectsVal,type = "res
        ponse")
projectsVal$PredStatusLogit=if_else(projectsVal$ProbCompleteLogit > 0.5,'Pred.complet
        ed','Pred.expired')

confMat1 <- table(projectsVal$funding_status,projectsVal$PredStatusLogit)
prec1 <- sum(diag(confMat1))/sum(confMat1)

prec1
```

```
## [1] 0.672725
```