Notes and Exercises Chapter 9 – Logistic Regression
Math 217
Saidi

Chapter 9

Logistic regression is a tool for building models when there is a categorical response variable with two levels, e.g., yes and no. Logistic regression is a type of **generalized linear model (GLM)** for response variables where regular multiple regression does not work very well. GLMs can be thought of as a two-stage modeling approach. **F**irst, model the response variable using a probability distribution, such as the binomial or Poisson distribution. Second, we model the parameter of the distribution using a collection of predictors and a special form of multiple regression. Ultimately, the application of a GLM will feel very similar to multiple regression, even if some of the details are different.

Logistic Regression is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

The fundamental equation of generalized linear model is:

$$g(E(y)) = \alpha + \beta x1 + \gamma x2$$

Here, g() is the link function, E(y) is the expectation of target variable and $\alpha + \beta x1 + \gamma x2$ is the linear predictor (α,β,γ to be predicted). The role of link function is to 'link' the expectation of y to linear predictor.

Since probability must always be positive, we'll put the linear equation in exponential form. For any value of slope and dependent variable, exponent of this equation will never be negative.

$$\hat{p} = e^{(\beta_0+\beta(predictor))} \quad \text{------- (b)}$$

To make the probability less than 1, we must divide p by a number greater than p. This can simply be done by:

$$\hat{p} = \frac{e^{(\beta_0+\beta(predictor)}}{e^{(\beta_0+\beta(predictor))}+1} \quad \text{----- (c)}$$

Using (a), (b) and (c), we can redefine the probability as:

$$\hat{p} = \frac{e^{y}}{1+e^{y}} \quad \text{--- (d)}$$

where $\hat{p}$ is the probability of success. *This (d) is the Logit Function.*
If $\hat{p}$ is the probability of success, $1 - \hat{p}$ will be the probability of failure which can be written as:

$\hat{q} = 1 - \hat{p} = 1 - \frac{e^y}{1+e^y}$ --- (e)

where $\hat{q}$ is the probability of failure.

On dividing, (d) / (e), we get, $\left(\frac{\hat{p}}{1-\hat{p}}\right) = e^y$ — odds

Odds of winning

2:1

$\hat{p}(win) = \frac{2}{3}$

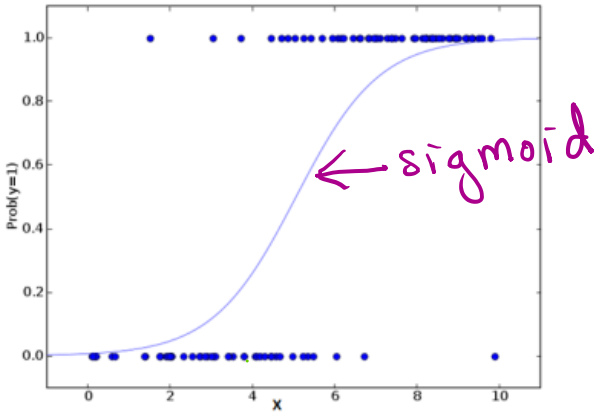
$\hat{p}(lose) = \frac{1}{3}$

After taking log on both sides, we get $log\left(\frac{\hat{p}}{1-\hat{p}}\right) = y$

$log\left(\frac{\hat{p}}{1-\hat{p}}\right)$ is the **link function.** Logarithmic transformation on the outcome variable allows us to model a non-linear association in a linear way.

After substituting value of y, we'll get: $log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \beta_0 + \beta(predictor)$

This is the equation used in Logistic Regression. Here ($\hat{p}$/1-$\hat{p}$) is the odd ratio. Whenever the log of odd ratio is found to be positive, the probability of success is always more than 50%. A typical logistic model plot is shown below. You can see probability never goes below 0 and above 1.



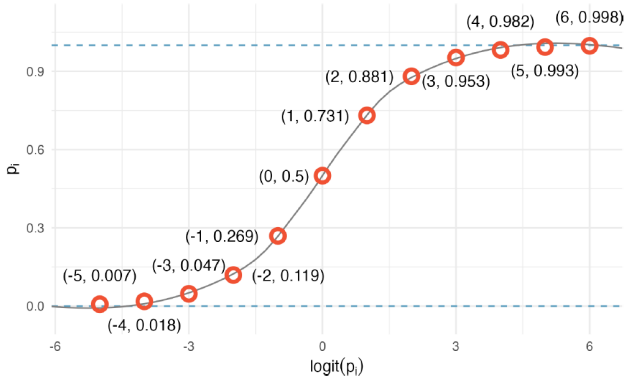


Figure 9.1: Values of p_i against values of $logit(p_i)$.

We will use the **`resume`** dataset from openintro (https://openintro.org/data/index.php?data=resume)

Which resume attributes drive job callbacks?

This experiment data comes from a study that sought to understand the influence of race and gender on job application callback rates. The study monitored job postings in Boston and Chicago for several months during 2001 and 2002 and used this to build up a set of test cases. Over this time period, the researchers randomly generating resumes to go out to a job posting, such as years of experience and education details, to create a realistic-looking resume. They then

randomly assigned a name to the resume that would communicate the applicant's gender and race. The first names chosen for the study were selected so that the names would predominantly be recognized as belonging to black or white individuals. For example, Lakisha was a name that their survey indicated would be interpreted as a black woman, while Greg was a name that would generally be interpreted to be associated with a white male.

The Dataset

The first names that were used and randomly assigned in this experiment were selected so that they would predominantly be recognized as belonging to Black or White individuals; other races were not considered in this study. While no name would definitively be inferred as pertaining to a Black individual or to a White individual, the researchers conducted a survey to check for racial association of the names; names that did not pass this survey check were excluded from usage in the experiment.
You can find the full set of names that did pass the survey test and were ultimately used in the study in Table 9.1 (textbook). For example, Lakisha was a name that their survey indicated would be interpreted as a Black woman, while Greg was a name that would generally be interpreted to be associated with a White male.

Table 9.2: Descriptions of nine variables from the `resume` dataset. Many of the variables are indicator variables, meaning they take the value 1 if the specified characteristic is present and 0 otherwise.

variable	description
`received_callback`	Specifies whether the employer called the applicant following submission of the application for the job.
`job_city`	City where the job was located: Boston or Chicago.
`college_degree`	An indicator for whether the resume listed a college degree.
`years_experience`	Number of years of experience listed on the resume.
`honors`	Indicator for the resume listing some sort of honors, e.g. employee of the month.
`military`	Indicator for if the resume listed any military experience.
`has_email_address`	Indicator for if the resume listed an email address for the applicant.
`race`	Race of the applicant, implied by their first name listed on the resume.
`sex`	Sex of the applicant (limited to only and in this study), implied by the first name listed on the resume.

The outcome (response variable) we will explore is “received_callback”.

Notation for a logistic regression model.

The outcome variable for a GLM is denoted by Y_i, where the index i is used to represent observation i. In the resume application, Y_i will be used to represent whether resume i received a callback ($Y_i = 1$) or not ($Y_i = 0$).

The Full Regression Model

```
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       -2.53494    0.18531 -13.679  < 2e-16 ***
job_cityChicago   -0.41782    0.11468  -3.643 0.000269 ***
years_experience   0.01962    0.01010   1.942 0.052188 .
honors             0.75131    0.18473   4.067 4.76e-05 ***
military          -0.32473    0.21563  -1.506 0.132087
has_email_address  0.26879    0.11663   2.305 0.021193 *
racewhite          0.43709    0.10810   4.043 5.27e-05 ***
genderm           -0.23905    0.13737  -1.740 0.081814 .
computer_skills   -0.24205    0.13963  -1.733 0.083011 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2726.9  on 4869  degrees of freedom
Residual deviance: 2656.6  on 4861  degrees of freedom
AIC: 2674.6
```

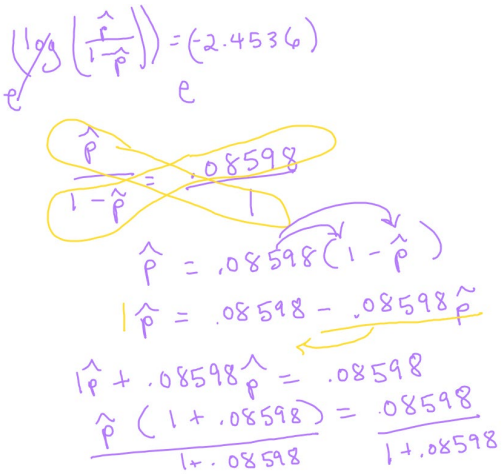
$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = -2.535 - 0.418(job\ city_{chicago}) + 0.0196(years\ experience) + 0.751(honors) - 0.325(military)$$
$$+ 0.269(has\ email\ address) + 0.437(race_{white}) - 0.239(gender_{male}) - 0.242(computer\ skills)$$

Example: Use this model summarized in Table 9.4 to estimate the probability of receiving a callback for: ***a job in Chicago where the candidate lists 14 years' experience, no honors, no military experience, includes an email address, has computer skills, and has a first name that implies they are a White male.***

$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = -2.535 - 0.418 + 0.0196(14) + 0.751(0) - 0.325(0) + 0.269(1) + 0.437(1) - 0.239(1) - 0.242(1)$$

$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = -2.4536$$

$$\frac{.08598}{1+.08598} \quad \frac{e^{-2.4536}}{1+e^{-2.4536}} = 0.07918 = 7.92\%\ chance\ of\ receiving\ a\ callback$$



chapter 9 logistic regression

Rachel Saidi

Load library tidyverse and resume dataset

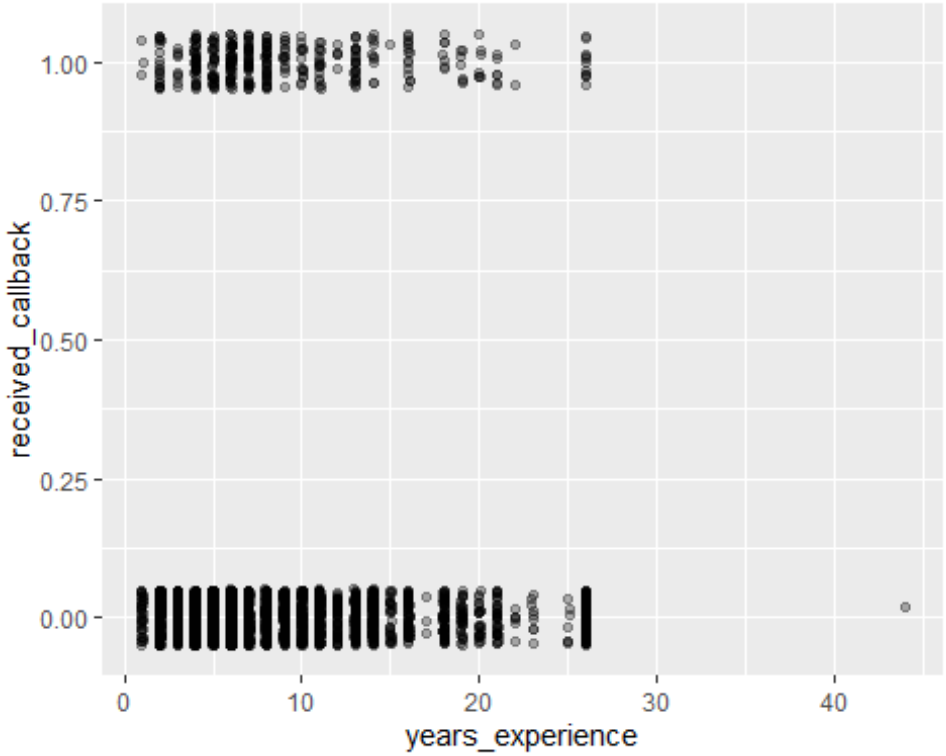
Set your working directory to find the saved dataset and use “head” to view the variables in the dataset

```
library(tidyverse)
library(tidymodels)
library(openintro)
data(resume)
head(resume)
```

```
# A tibble: 6 × 30
  job_ad_id job_city job_industry               job_type   job_fed_contractor
      <dbl> <chr>    <chr>                      <chr>                   <dbl>
1       384 Chicago  manufacturing              supervisor                 NA
2       384 Chicago  manufacturing              supervisor                 NA
3       384 Chicago  manufacturing              supervisor                 NA
4       384 Chicago  manufacturing              supervisor                 NA
5       385 Chicago  other_service              secretary                   0
6       386 Chicago  wholesale_and_retail_trade sales_rep                   0
#
```

Note that in the plot below, we use the **geom_jitter()** function to create the illusion of separation in our data. Because the y value is categorical, all of the points would either lie exactly on “dead” or “alive”, making the individual points hard to see. To counteract this, geom_jitter() will move the points a small random amount up or down.

```
ggplot(data = resume, aes(x = years_experience, y = received_callback)) +
  geom_jitter(width = .05, height = 0.05, alpha = 0.3)
```

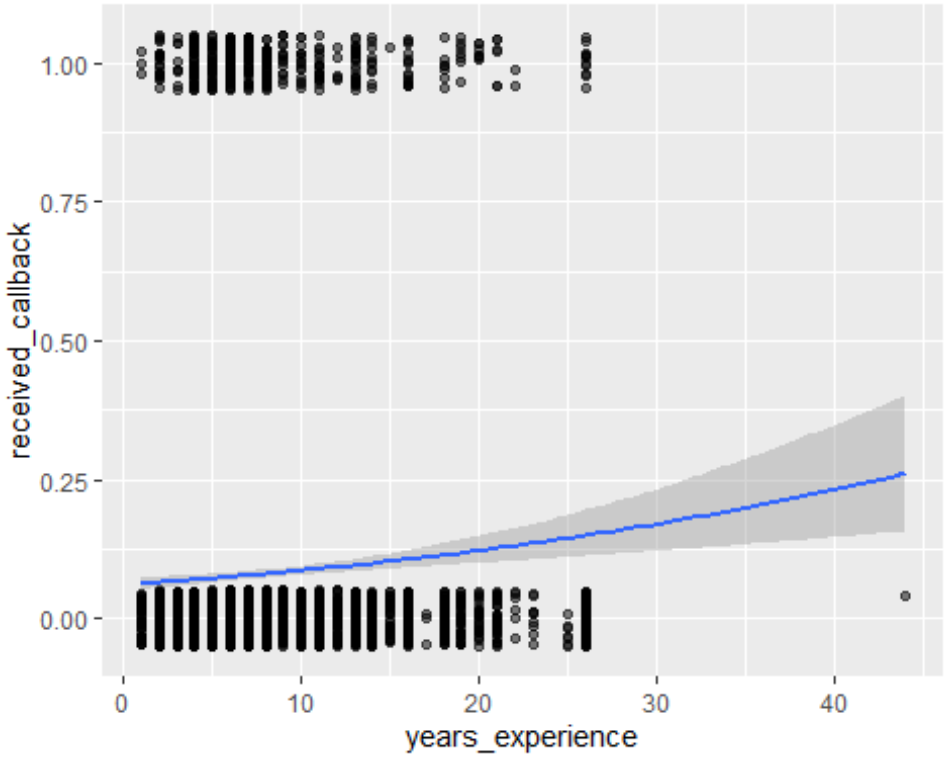


Fit a logistic regression model

Start with the simple premise above, that years experience is associated with the likelihood of receiving a callback.

```
plot_log <- ggplot(data = resume, aes(y = received_callback, x = years_experience)) +
  geom_jitter(width = 0, height = 0.05, alpha = 0.5) +
  geom_smooth(method = "glm", method.args = list(family = "binomial"))
plot_log
```

```
`geom_smooth()` using formula = 'y ~ x'
```



geom_smooth(method = "glm") creates a **sigmoid** curve, like a stretched-out S. This curve fits to the binary values (zeros and ones plotted). It shows the predicted probabilities of receiving a callback, based on years experience. You can see that even a candidate with 20 years of experience has about a 12.5% chance of receiving a callback.

Odds scale

To combat the problem of the scale of the y variable, we can change the scale of the variable on the y-axis. Instead of thinking about the probability of receiving a callback, we can think about the odds. While these two concepts are often conflated, they are not the same. They are however, related by the simple formula below. The odds of a binary event are the ratio of how often it happens, to how often it doesn't happen.

$$odds(\hat{y}) = \frac{\hat{y}}{1-\hat{y}} = exp(\beta_0 + \beta_1 x)$$

Thus, if the probability of receiving a callback is 10%, then the odds of receiving a callback are .1/(1-.1) = 1:9

Odds are commonly used to express uncertainty in a variety of contexts.

Odds are commonly used to express uncertainty in a variety of contexts.

```
# using base r logistic regression
#  model:  y ~ x
fit_log <- glm(data = resume, received_callback ~ years_experience, family = binomial())
summary(fit_log)


Call:
glm(formula = received_callback ~ years_experience, family = binomial(),
```

```
    data = resume)

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      -2.75960    0.09620 -28.687  < 2e-16 ***
years_experience  0.03908    0.00918   4.257 2.07e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2726.9  on 4869  degrees of freedom
Residual deviance: 2710.2  on 4868  degrees of freedom
AIC: 2714.2

Number of Fisher Scoring iterations: 5
```

equation log_odds(received_callback) = -2.7596+0.039(years_experience)

Note the AIC is 2714.2. We will try to improve that later (the smaller, the better).

log_odds(callback_5years experience) = -2.7596 + 0.039(5)

```
exp(-2.456)/(1+exp(-2.456))
```

```
[1] 0.07900089
```

Use the Tidymodels package to create a regression model

Create the training, testing, and validation data by splitting

Use initial_validation_split() function to create a random 3-way split the data into the

```
# convert received_callback to categorical variable
resume$received_callback <- as.factor(resume$received_callback)
set.seed(123)

#initial_split creates a single binary split of the data into a training set and testing
set (the default is 75% training 25% testing)
res_splits<- initial_validation_split(resume, strata = received_callback)

callback_train <- training(res_splits)
validation_set <- validation(res_splits)
callback_test  <- testing(res_splits)
```

Calculate training set proportions by callbacks

```
callback_train |>
  group_by(received_callback) |>
  tally() |>
  mutate(prop = n/sum(n))
```

```
# A tibble: 2 × 3
  received_callback     n   prop
  <fct>             <int>  <dbl>
1 0                  2688 0.920
2 1                   234 0.0801
```

We see that 7.9% of applicants in the training dataset received callbacks.

Calculate training set proportions by callbacks

```
validation_set |>
  group_by(received_callback) |>
  tally() |>
  mutate(prop = n/sum(n))

# A tibble: 2 × 3
  received_callback     n   prop
  <fct>             <int>  <dbl>
1 0                   886 0.910
2 1                    88 0.0903
```

The proportion of callbacks in the validation set is 9.03%

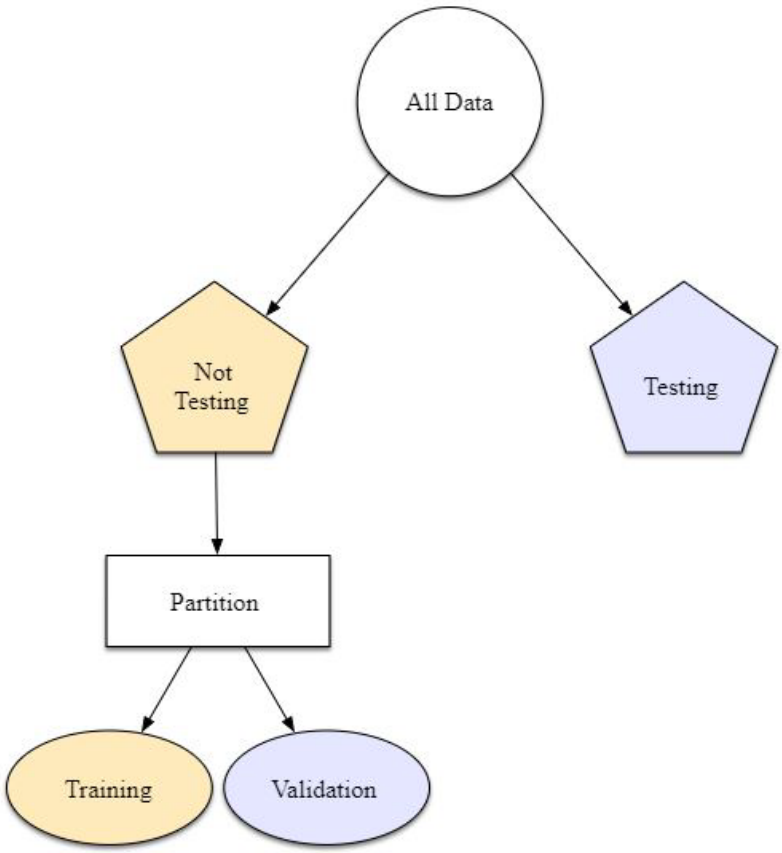
Calculate testing set proportions by callbacks

```
callback_test |>
  group_by(received_callback) |>
  tally() |>
  mutate(prop = n/sum(n))

# A tibble: 2 × 3
  received_callback     n   prop
  <fct>             <int>  <dbl>
1 0                   904 0.928
2 1                    70 0.0719
```

We see that in the testing set, 8.5% of applicants did receive callbacks.

For this case study, rather than using multiple iterations of resampling, let's create a single resample called a validation set. In tidymodels, a validation set is treated as a single iteration of resampling. This will be a split from the 37,500 stays that were not used for testing, which we called hotel_other. This split creates two new datasets:

- the set held out for the purpose of measuring performance, called the validation set
- the remaining data used to fit the model, called the training set.



Fitting Logistic Regression

You can fit any type of model (supported by tidymodels) using the following steps.

1. Call the model function: here we called logistic_reg( ) as we want to fit a logistic regression model.
2. Use set_engine( ) function to supply the family of the model. The “glm” argument as Logistic regression comes under the Generalized Linear Regression family.
3. Use set_mode( ) function and supply the type of model you want to fit. Classify pos vs neg, so it is a classification.
4. Next, you need to use the fit( ) function to fit the model and inside that in the form y~x, you have to provide the formula notation and dataset (callback_train).

plus notation → diabetes ~ ind_variable 1 + ind_variable 2 + …….so on

```
logit_fit <-
  logistic_reg(mode = "classification") |>
  set_engine(engine = "glm") |>
  set_mode("classification") |>
  fit(received_callback ~ years_experience, data = callback_train)
tidy(logit_fit)

# A tibble: 2 × 5
  term             estimate std.error statistic   p.value
  <chr>               <dbl>     <dbl>     <dbl>     <dbl>
```

```
1 (Intercept)       -2.70      0.123     -21.9  6.98e-106
2 years_experience   0.0312    0.0120      2.59 9.57e-  3
```

Odds Ratio

The interpretation of coefficients in the log-odds term does not make much sense if you need to report it in your article or publication. That is why the concept of odds ratio was introduced.

The ODDS is the ratio of the probability of an event occurring to the event not occurring. When we take a ratio of two such odds it called Odds Ratio.

you can get directly the odds ratios of the coefficient by supplying the exponentiate = True inside the tidy( ) function.

```
tidy(logit_fit, exponentiate = TRUE)
```

```
# A tibble: 2 × 5
  term             estimate std.error statistic   p.value
  <chr>               <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)        0.0675    0.123     -21.9  6.98e-106
2 years_experience   1.03      0.0120      2.59 9.57e-  3
```

Model Prediction

Test Data Class Prediction

The very next step is to generate the test predictions that we could use for model evaluation. To generate the class prediction (pos/ neg) we can use the predict function and supply the trained model object, test dataset and the type which is here “class” as we want the class prediction, not probabilities.

To get predicted values we use the function **predict**.

```
pred_class <- predict(logit_fit,
                      new_data = callback_test,
                      type = "class")

pred_prob <- predict(logit_fit,
                      new_data = callback_test,
                      type = "prob")

joined_pred_results <- callback_test |>
  select(received_callback) |>
  bind_cols(pred_class, pred_prob)
```

Model performance

yardstick is a package to get metrics on model performance. By default on classification problem yardstick::metrics returns accuracy and kappa.

https://yardstick.tidymodels.org/

```
library(yardstick)
metrics(joined_pred_results, truth = received_callback, estimate = .pred_class)
```

```
# A tibble: 2 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.928
2 kap      binary         0
```

According to Wikipedia, *accuracy* is how close a given set of measurements (observations or readings) are to their true value.

The accuracy of this model is 92.8%

Accuracy is calculated by:

$$\frac{\text{True Positive + True Negatives}}{\text{True Positive + True Negatives + False Positives + False Negatives}}$$

kappa

Kappa measures the model's reliability. It represents the extent to which the data collected in the study are correct representations of the variables measured.

Since kap = 0, the model does not appear to be a good representation.

The null model

It will be useful to have our null model stored as a model object. We can create such an object using tidymodels by specifying a logistic regression model with **no explanatory variables**.

```
null_mod <-
  logistic_reg(mode = "classification") |>
  set_engine("glm") |>
  fit(received_callback ~ 1, data = callback_train)
tidy(null_mod)
```

```
# A tibble: 1 × 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)    -2.44    0.0682     -35.8 5.96e-281
```

Compute the accuracy of the null model

```
pred <- callback_train |>  # use the training dataset
  select(received_callback, years_experience) |> #select only these columns
  bind_cols(predict(null_mod, new_data = callback_train, type = "class"))|>
  rename(null_mod = .pred_class)

accuracy(pred, received_callback, null_mod)
```

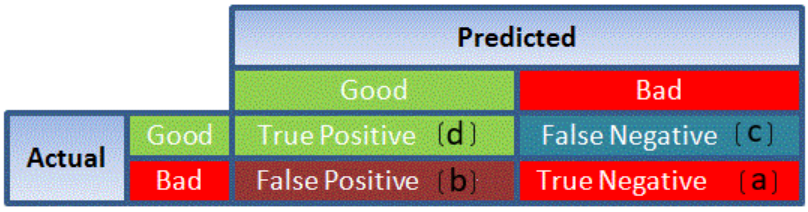
```
# A tibble: 1 × 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.920
```

The accuracy estimate of this null model is almost the same as the prior model: = 92.0%

Create a confusion matrix

This is a two-way table that counts how often our model made the correct prediction. Note that there are two different types of mistakes that our model can make: predicting a high income when the income was in fact low (a Type I error), and predicting a low income when the income was in fact high (a Type II error).

Confusion Matrix is nothing but a tabular representation of Actual vs Predicted values. This helps us to find the accuracy of the model and avoid overfitting. This is how it looks like:



From the confusion matrix, **Specificity and Sensitivity** can be derived as illustrated below:

$$\text{True Negative Rate (TNR), specificity} = \frac{A}{A+B}$$
$$\text{False Positve Rate (FPR), } 1-\text{specificity} = \frac{B}{A+B}$$

} sum to 1

$$\text{True Positive Rate (TPR), sensitivity} = \frac{D}{C+D}$$
$$\text{False Negative Rate (FNR)} = \frac{C}{C+D}$$

} sum to 1

Specificity and Sensitivity plays a crucial role in deriving ROC curve.

This is a two-way table that counts how often our model made the correct prediction. Note that there are two different types of mistakes that our model can make: predicting receiving a callback when a callback was not received (a Type I error), and predicting not receiving a callback when the callback was received (a Type II error).

```
confusion_null <- pred |>
  conf_mat(truth = received_callback, estimate = null_mod)
confusion_null
```

```
          Truth
Prediction    0    1
         0 2688  234
         1    0    0
```

Note that the null model predicts that *NONE* do not receive a callback (receives_callback = 0), so it makes many *Type II errors* (false negatives) and *NONE* receive a callback (receives_callback = 1) *Type I errors* (false positives).

Build Logit Models and Predict

Start by building the logit model using training data

Beating the null model shouldn't be hard. Our first attempt will be to employ a simple logistic regression model. First, we'll fit the model using only one explanatory variable: years_experience. It stands to reason that more experienced people are more likely to get a job offer.

This is the same as the first model we built in base R at the beginning.

```
mod_log_1 <-
  logistic_reg(mode = "classification") |>
  set_engine("glm") |>
  fit(received_callback ~ years_experience, data = callback_train)
tidy(mod_log_1)
```

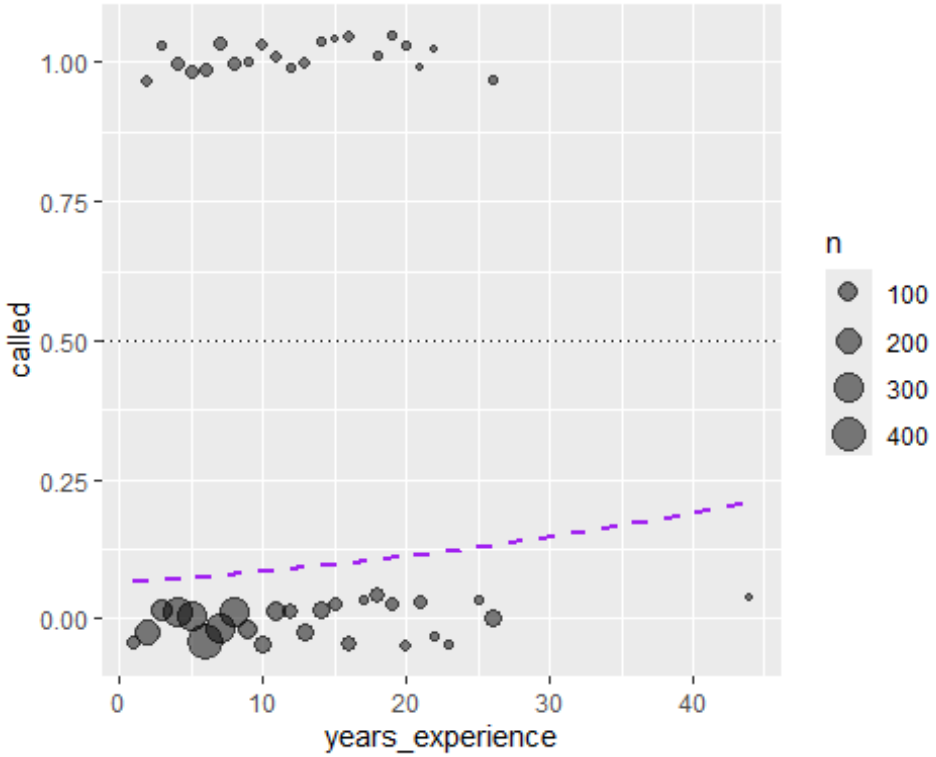
```
# A tibble: 2 × 5
  term             estimate std.error statistic   p.value
  <chr>               <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)       -2.70      0.123     -21.9  6.98e-106
2 years_experience   0.0312    0.0120      2.59 9.57e-  3
```

Illustrate the predicted probability

Illustrate how the predicted probability of dying varies in our simple logistic regression model with respect to a person's age. Notice the sigmoid curve (in purple) - this is the shape of a logistic curve.

```
train_plus <- callback_train |>
  mutate(called = as.integer(received_callback == 1))

ggplot(train_plus, aes(x = years_experience, y = called)) +
  geom_count(
    position = position_jitter(width = 0, height = 0.05),
    alpha = 0.5 ) +
  geom_smooth(method = "glm", method.args = list(family = "binomial"),
              color = "purple", lty = 2, se = FALSE) +
  geom_hline(aes(yintercept = 0.5), linetype = 3)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



This still looks very similar to the original plot.

ROC - Receiver Operating Characteristics

Receiver Operating Characteristics Curve traces the percentage of true positives accurately predicted by a given logit model as the prediction probability cutoff is lowered from 1 to 0. For a good model, as the cutoff is lowered, it should mark more of actual 1's as positives and lesser of actual 0's as 1's. So for a good model, the curve should rise steeply, indicating that the TPR (Y-Axis) increases faster than the FPR (X-Axis) as the cutoff score decreases.

AUC - Area under the curve

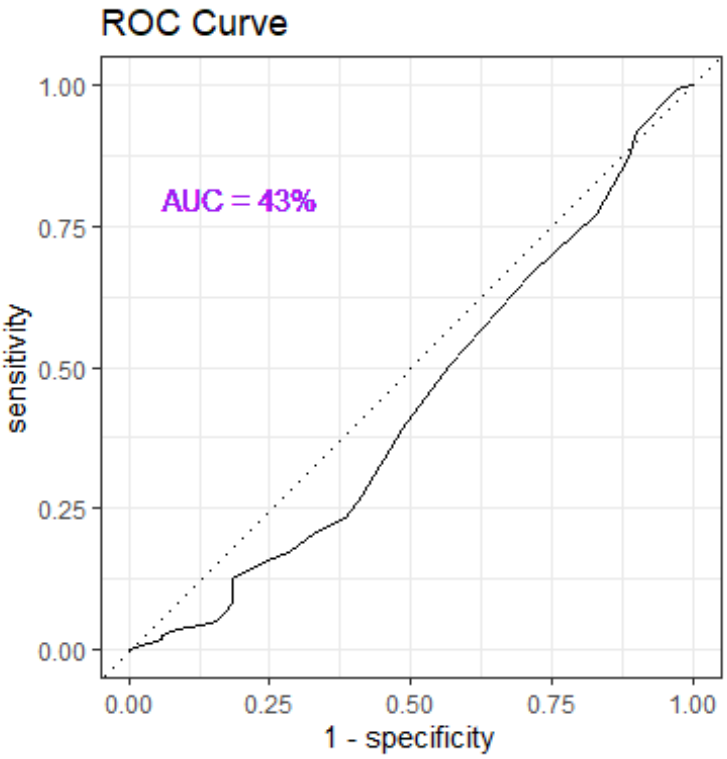
The greater the area under the ROC curve, better the predictive ability of the model

```
roc_auc(bind_cols(callback_test, pred_prob), truth = received_callback, .pred_1)

# A tibble: 1 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.434
```

The AUC estimate is 43.3% - this is VERY low, meaning the model is not great for predicting probabilities.

```
roc_data <- roc_curve(bind_cols(callback_test, pred_prob), truth = received_callback, .pr
ed_1)

roc_data |>
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path() +
  geom_abline(lty = 3) +
  coord_equal()+
```

```
  ggtitle("ROC Curve")+
  theme_bw()+
  geom_text(x=0.2, y=.8, label="AUC = 43%", color ="purple")
```



The Full Regression Model

Computing the full logistic regression model will feel similar to the process for computing the full multiple linear regression model. We will use **GLM** to replace **LM** and we need to include **family = binomial()**

```
log_mod_full <- glm(data=resume, received_callback ~ job_city + college_degree + years_ex
perience + honors + military + has_email_address + race + gender + computer_skills + resu
me_quality, family = binomial())
summary(log_mod_full)


Call:
glm(formula = received_callback ~ job_city + college_degree +
    years_experience + honors + military + has_email_address +
    race + gender + computer_skills + resume_quality, family = binomial(),
    data = resume)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       -2.47003    0.33302  -7.417 1.20e-13 ***
job_cityChicago   -0.42156    0.11493  -3.668 0.000245 ***
college_degree    -0.06682    0.12244  -0.546 0.585214
years_experience   0.01896    0.01037   1.828 0.067475 .
honors             0.75741    0.18539   4.085 4.40e-05 ***
military          -0.32315    0.21597  -1.496 0.134577
has_email_address  0.25501    0.23523   1.084 0.278320
```

```
racewhite          0.43659    0.10811   4.038 5.38e-05 ***
genderm           -0.22604    0.13996  -1.615 0.106289
computer_skills   -0.24265    0.14064  -1.725 0.084476 .
resume_qualitylow -0.01140    0.23602  -0.048 0.961473
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2726.9  on 4869  degrees of freedom
Residual deviance: 2656.3  on 4859  degrees of freedom
AIC: 2678.3

Number of Fisher Scoring iterations: 5
```

Performance of Logistic Regression Model

To evaluate the performance of a logistic regression model, we must consider few metrics: AIC and p-values.

AIC (Akaike Information Criteria) – The analogous metric of adjusted R^2 in logistic regression is AIC. AIC is the measure of fit which penalizes model for the number of model coefficients. Therefore, we always prefer model with **minimum** AIC value.

Assessing our full model

5. The AIC is 2678
6. The largest p-values come from college_degree and resume_quality

Remove college_degree and resume_quality and re-run the model

```
log_mod2 <- glm(data=resume, received_callback ~ job_city + years_experience + honors + m
ilitary + has_email_address + race + gender + computer_skills, family = binomial())
summary(log_mod2)


Call:
glm(formula = received_callback ~ job_city + years_experience +
    honors + military + has_email_address + race + gender + computer_skills,
    family = binomial(), data = resume)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       -2.53494    0.18531 -13.679  < 2e-16 ***
job_cityChicago   -0.41782    0.11468  -3.643 0.000269 ***
years_experience   0.01962    0.01010   1.942 0.052188 .
honors             0.75131    0.18473   4.067 4.76e-05 ***
military          -0.32473    0.21563  -1.506 0.132087
has_email_address  0.26879    0.11663   2.305 0.021193 *
racewhite          0.43709    0.10810   4.043 5.27e-05 ***
genderm           -0.23905    0.13737  -1.740 0.081814 .
computer_skills   -0.24205    0.13963  -1.733 0.083011 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 2726.9  on 4869  degrees of freedom
Residual deviance: 2656.6  on 4861  degrees of freedom
AIC: 2674.6

Number of Fisher Scoring iterations: 5
```

The AIC decreased to 2675 and all p-values are relatively low. This could possibly be our best model.

VIF - Variance Inflation Factor

Like in case of linear regression, we should check for multicollinearity in the model.

One way to measure multicollinearity is the variance inflation factor (VIF), which assesses how much the variance of an estimated regression coefficient increases if your predictors are correlated.

A VIF between 5 and 10 indicates high correlation that may be problematic. And if the VIF goes above 10, you can assume that the regression coefficients are poorly estimated due to multicollinearity.

```
#install.packages("car")  #car package will calculate the VIF
library(car)
vif(log_mod2)
```

```
         job_city  years_experience           honors          military
         1.162480          1.176885          1.057478          1.189503
has_email_address              race           gender   computer_skills
         1.204211          1.001827          1.126755          1.159797
```

All X variables in the model have VIF well below 4, so log_mod2 model does not appear to have multicollinearity.

Example for predicting

Use this model(log_mod2) to estimate the probability of receiving a callback for: a job in Chicago where the candidate lists 14 years' experience, no honors, no military experience, includes an email address, has computer skills, and has a first name that implies they are a White male.

```
# the log odds value of the above characteristics
-2.535-0.418+0.0196*14+0.269+0.437-0.239-0.242
```

```
[1] -2.4536
```

```
# calculate the probability using the above value
exp(-2.4536)/(1+exp(-2.4536))
```

```
[1] 0.07917569
```

The probability of receiving a callback for a candidate looking at a job in Chicago with 14 years' experience, no honors, no military experience, included an email address, has computer skills, and has a first name that implies they are White male is 7.92%

Overall

Although this tutorial goes through many concepts that are involved with logistic regression, one of the take-aways is that there are many components involved in building the model and then testing the model.

Homework Chapter 9

1. Review section 9.5 (the chapter review)
2. Required problems from textbook section 9.6 exercises: 3,4,5,6,8
3. Optional Unit 3 Tutorial Regression Modeling:

   9 - Logistic regression