AAHW3 - Kilbourn

CJK

2023-06-26

Homework 3 - R Portion

This is in response to question 2. The prompt was:

**2. Build an optimal Lasso, Ridge, or ElasticNet logistic regression model using the attrition data we used for class that includes a “penalty” and “mixture” parameter. I’ll leave it up to you to determine what “optimal” means.**

My first step is to load the attrition data set (note that because this is under a different project in order to connect it to github, I will need to import the dataset again so it’s compatible with git and with this markdown file, even though I have already imported this dataset during the session three lab).

library(readxl)  
WA\_Fn\_UseC\_HR\_Employee\_Attrition <- read\_excel("WA\_Fn-UseC\_-HR-Employee-Attrition.xlsx")  
View(WA\_Fn\_UseC\_HR\_Employee\_Attrition)

Also, the name of the dataset is crazy long, so I will shorten it to just data for now.

data <- WA\_Fn\_UseC\_HR\_Employee\_Attrition

I will call the tidyverse package - I may need other packages later on, but this one will be sufficient to get me started.

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.2 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

And I have already dome some exploratory ‘poking’ at this data during the lab, but I will run a couple of functions just to refamiliarize myself with what I’m working with. First, I will glimpse the data.

glimpse(data)

## Rows: 1,470  
## Columns: 35  
## $ Age <dbl> 41, 49, 37, 33, 27, 32, 59, 30, 38, 36, 35, 2…  
## $ Attrition <chr> "Yes", "No", "Yes", "No", "No", "No", "No", "…  
## $ BusinessTravel <chr> "Travel\_Rarely", "Travel\_Frequently", "Travel…  
## $ DailyRate <dbl> 1102, 279, 1373, 1392, 591, 1005, 1324, 1358,…  
## $ Department <chr> "Sales", "Research & Development", "Research …  
## $ DistanceFromHome <dbl> 1, 8, 2, 3, 2, 2, 3, 24, 23, 27, 16, 15, 26, …  
## $ Education <dbl> 2, 1, 2, 4, 1, 2, 3, 1, 3, 3, 3, 2, 1, 2, 3, …  
## $ EducationField <chr> "Life Sciences", "Life Sciences", "Other", "L…  
## $ EmployeeCount <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …  
## $ EmployeeNumber <dbl> 1, 2, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16,…  
## $ EnvironmentSatisfaction <dbl> 2, 3, 4, 4, 1, 4, 3, 4, 4, 3, 1, 4, 1, 2, 3, …  
## $ Gender <chr> "Female", "Male", "Male", "Female", "Male", "…  
## $ HourlyRate <dbl> 94, 61, 92, 56, 40, 79, 81, 67, 44, 94, 84, 4…  
## $ JobInvolvement <dbl> 3, 2, 2, 3, 3, 3, 4, 3, 2, 3, 4, 2, 3, 3, 2, …  
## $ JobLevel <dbl> 2, 2, 1, 1, 1, 1, 1, 1, 3, 2, 1, 2, 1, 1, 1, …  
## $ JobRole <chr> "Sales Executive", "Research Scientist", "Lab…  
## $ JobSatisfaction <dbl> 4, 2, 3, 3, 2, 4, 1, 3, 3, 3, 2, 3, 3, 4, 3, …  
## $ MaritalStatus <chr> "Single", "Married", "Single", "Married", "Ma…  
## $ MonthlyIncome <dbl> 5993, 5130, 2090, 2909, 3468, 3068, 2670, 269…  
## $ MonthlyRate <dbl> 19479, 24907, 2396, 23159, 16632, 11864, 9964…  
## $ NumCompaniesWorked <dbl> 8, 1, 6, 1, 9, 0, 4, 1, 0, 6, 0, 0, 1, 0, 5, …  
## $ Over18 <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", …  
## $ OverTime <chr> "Yes", "No", "Yes", "Yes", "No", "No", "Yes",…  
## $ PercentSalaryHike <dbl> 11, 23, 15, 11, 12, 13, 20, 22, 21, 13, 13, 1…  
## $ PerformanceRating <dbl> 3, 4, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3, …  
## $ RelationshipSatisfaction <dbl> 1, 4, 2, 3, 4, 3, 1, 2, 2, 2, 3, 4, 4, 3, 2, …  
## $ StandardHours <dbl> 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 8…  
## $ StockOptionLevel <dbl> 0, 1, 0, 0, 1, 0, 3, 1, 0, 2, 1, 0, 1, 1, 0, …  
## $ TotalWorkingYears <dbl> 8, 10, 7, 8, 6, 8, 12, 1, 10, 17, 6, 10, 5, 3…  
## $ TrainingTimesLastYear <dbl> 0, 3, 3, 3, 3, 2, 3, 2, 2, 3, 5, 3, 1, 2, 4, …  
## $ WorkLifeBalance <dbl> 1, 3, 3, 3, 3, 2, 2, 3, 3, 2, 3, 3, 2, 3, 3, …  
## $ YearsAtCompany <dbl> 6, 10, 0, 8, 2, 7, 1, 1, 9, 7, 5, 9, 5, 2, 4,…  
## $ YearsInCurrentRole <dbl> 4, 7, 0, 7, 2, 7, 0, 0, 7, 7, 4, 5, 2, 2, 2, …  
## $ YearsSinceLastPromotion <dbl> 0, 1, 0, 3, 2, 3, 0, 0, 1, 7, 0, 0, 4, 1, 0, …  
## $ YearsWithCurrManager <dbl> 5, 7, 0, 0, 2, 6, 0, 0, 8, 7, 3, 8, 3, 2, 3, …

Then I’ll get a better look at the column names.

colnames(data)

## [1] "Age" "Attrition"   
## [3] "BusinessTravel" "DailyRate"   
## [5] "Department" "DistanceFromHome"   
## [7] "Education" "EducationField"   
## [9] "EmployeeCount" "EmployeeNumber"   
## [11] "EnvironmentSatisfaction" "Gender"   
## [13] "HourlyRate" "JobInvolvement"   
## [15] "JobLevel" "JobRole"   
## [17] "JobSatisfaction" "MaritalStatus"   
## [19] "MonthlyIncome" "MonthlyRate"   
## [21] "NumCompaniesWorked" "Over18"   
## [23] "OverTime" "PercentSalaryHike"   
## [25] "PerformanceRating" "RelationshipSatisfaction"  
## [27] "StandardHours" "StockOptionLevel"   
## [29] "TotalWorkingYears" "TrainingTimesLastYear"   
## [31] "WorkLifeBalance" "YearsAtCompany"   
## [33] "YearsInCurrentRole" "YearsSinceLastPromotion"   
## [35] "YearsWithCurrManager"

And I remember there being no missing data, but I will double check this just to be certain.

apply(is.na(data), 2, sum)

## Age Attrition BusinessTravel   
## 0 0 0   
## DailyRate Department DistanceFromHome   
## 0 0 0   
## Education EducationField EmployeeCount   
## 0 0 0   
## EmployeeNumber EnvironmentSatisfaction Gender   
## 0 0 0   
## HourlyRate JobInvolvement JobLevel   
## 0 0 0   
## JobRole JobSatisfaction MaritalStatus   
## 0 0 0   
## MonthlyIncome MonthlyRate NumCompaniesWorked   
## 0 0 0   
## Over18 OverTime PercentSalaryHike   
## 0 0 0   
## PerformanceRating RelationshipSatisfaction StandardHours   
## 0 0 0   
## StockOptionLevel TotalWorkingYears TrainingTimesLastYear   
## 0 0 0   
## WorkLifeBalance YearsAtCompany YearsInCurrentRole   
## 0 0 0   
## YearsSinceLastPromotion YearsWithCurrManager   
## 0 0

So there’s no missing data, and there are 35 total variables. I will now go into the regularization (shrinking) methods: ridge regression, lasso, and elasticnet. The concept behind regularization is to shrink the coefficient estimates of predictor variables. This is useful if a model has a large number of predictors (and we’re not sure which ones are useful and which ones are not ‘pulling their weight’) since a model will generally perform ‘better’ as the number of predictors increases, regardless of the usefulness of any and all predictors. So if a model has 100 variables but only 8 are responsible for the majority of effective performance of the model, regularization methods will show a significant drop-off in the decrease of effectiveness in the non-useful 92 variables. It is important to remember that the goal is parsimony, wanting to find the best performing model with strong predictive power and good fit that also includes the lowest quantity of predictor variables. The methods of ridge, lasso, and elasticnet vary slightly in their regularization processes, but they all shrink the coefficients of the predictors. Ridge shrinks toward 0 but never gets to 0, lasso can shrink coefficients to 0, and elasticnet combines elements of both ridge and lasso.

And 35 is a pretty hefty number of predictors (especially when my goal is to be able to tell key business leaders what ‘levers’ to pull to have an influence on attrition), so I will now explore which methods of regularization provide us with the best fitting model to move forward with.

**I will experiment with a ridge regression first.** I did some research on a few blogs, and I did find that ridge is not recommended for logistic models, however, I will proceed with running a ridge just to get the experience with it (and see, comparatively, how it performs with other regularization processes).

The way to create a ridge regression model specification is via the linear\_reg() function and setting the mixture to 0 (this will be very similar to lasso except the mixture will be set to 1, and the elasticnet in which the number will be between 0 and 1 indicating a blend of ridge and lasso). For now, my penalty is set to 0 because I don’t have any knowledge or information at this time to make a more informed selection - that will come later.

I will also call the glmnet and parsnip packages. Note that I adjusted the code to reflect that this is a logistic regression (logistic\_reg) and that my mode should be set to classification. I also will set my seed here. I’m not sure if I need to with regularization, but it should help me with reproducability of my results if I need that).

library(parsnip)  
library(glmnet)

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

## Loaded glmnet 4.1-7

set.seed(31)  
ridge\_spec <- logistic\_reg(mixture = 0, penalty = 0) %>%  
 set\_mode("classification") %>%  
 set\_engine("glmnet")

I will now fit the ridge regression to my data. Just to remind myself, I am currently looking at Attrition as my outcome variable (binary outcome of ‘yes’ or ‘no’) and all other variables as my predictors.

ridge\_fit <- fit(ridge\_spec, Attrition ~ ., data = data)

So I learned something! In the code above, the error stated that ’contrasts can be applied only to factors with 2 or more levels. I remember from exploring this data before that some of the predictors had the exact same responses across all respondents. This may be the problem. I will look at which of these factors fit this description and create a new dataframe with the remaining predictors only.

First, I will look at character (non-qualitative) variables.

library(conflicted)  
conflict\_prefer("filter", "dplyr")

## [conflicted] Will prefer dplyr::filter over any other package.

data %>%  
 select\_if(is.character) %>%  
 map(unique)

## $Attrition  
## [1] "Yes" "No"   
##   
## $BusinessTravel  
## [1] "Travel\_Rarely" "Travel\_Frequently" "Non-Travel"   
##   
## $Department  
## [1] "Sales" "Research & Development" "Human Resources"   
##   
## $EducationField  
## [1] "Life Sciences" "Other" "Medical" "Marketing"   
## [5] "Technical Degree" "Human Resources"   
##   
## $Gender  
## [1] "Female" "Male"   
##   
## $JobRole  
## [1] "Sales Executive" "Research Scientist"   
## [3] "Laboratory Technician" "Manufacturing Director"   
## [5] "Healthcare Representative" "Manager"   
## [7] "Sales Representative" "Research Director"   
## [9] "Human Resources"   
##   
## $MaritalStatus  
## [1] "Single" "Married" "Divorced"  
##   
## $Over18  
## [1] "Y"  
##   
## $OverTime  
## [1] "Yes" "No"

The Over18 variable is a problem - all responses are the same, so there is no variance.

Now I will look at numeric variables

data %>%  
 select\_if(is.numeric) %>%  
 map(~ unique(.) %>% length())

## $Age  
## [1] 43  
##   
## $DailyRate  
## [1] 886  
##   
## $DistanceFromHome  
## [1] 29  
##   
## $Education  
## [1] 5  
##   
## $EmployeeCount  
## [1] 1  
##   
## $EmployeeNumber  
## [1] 1470  
##   
## $EnvironmentSatisfaction  
## [1] 4  
##   
## $HourlyRate  
## [1] 71  
##   
## $JobInvolvement  
## [1] 4  
##   
## $JobLevel  
## [1] 5  
##   
## $JobSatisfaction  
## [1] 4  
##   
## $MonthlyIncome  
## [1] 1349  
##   
## $MonthlyRate  
## [1] 1427  
##   
## $NumCompaniesWorked  
## [1] 10  
##   
## $PercentSalaryHike  
## [1] 15  
##   
## $PerformanceRating  
## [1] 2  
##   
## $RelationshipSatisfaction  
## [1] 4  
##   
## $StandardHours  
## [1] 1  
##   
## $StockOptionLevel  
## [1] 4  
##   
## $TotalWorkingYears  
## [1] 40  
##   
## $TrainingTimesLastYear  
## [1] 7  
##   
## $WorkLifeBalance  
## [1] 4  
##   
## $YearsAtCompany  
## [1] 37  
##   
## $YearsInCurrentRole  
## [1] 19  
##   
## $YearsSinceLastPromotion  
## [1] 16  
##   
## $YearsWithCurrManager  
## [1] 18

EmployeeCount and StandardHours are also problematic. Once more, there is no variance since responses are the same across all respondents.

There are a couple different methods for handling these three variables. Because ridge regression is new to me, I’m not sure if there is a preferred handling method (I experimented with creating a recipe using the step\_zv() function, but I wasn’t sure if that was overcomplicating things). So for now, I used the select function to create a new dataframe called newdata that does not include the three non-variance variables.

newdata <- data %>% select(-Over18,-EmployeeCount,-StandardHours)

I also discovered through trial and error (didn’t leave all the error and wrong codes because they would be a mile long!) that something needed to be adjusted for my outcome variable as well. I received some words of wisdom that my outcome variable needed to be set as a factor instead of a character, so I will do this now.

newdata$Attrition <- factor(newdata$Attrition)

And I will confirm that it is, indeed, a character now.

is.factor(newdata$Attrition)

## [1] TRUE

And now, with my revised dataset, I will try to fit my ridge regression.

ridge\_fit <- fit(ridge\_spec, Attrition ~ ., data = newdata)

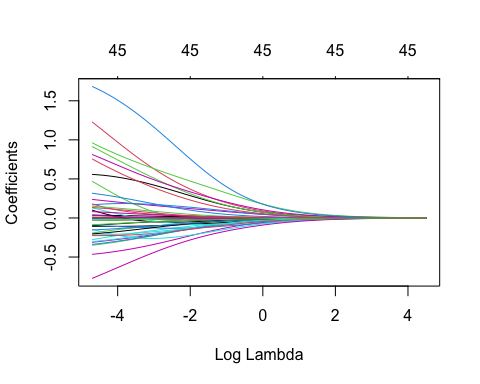
That worked! Now I will use the tidy function to look at the estimates for each variable. Note that the penalty is at 0 for all of them because I haven’t tuned the penalty yet (since I don’t have the info yet to know what it should be tuned to).

tidy(ridge\_fit)

## # A tibble: 46 × 3  
## term estimate penalty  
## <chr> <dbl> <dbl>  
## 1 (Intercept) 2.61 0  
## 2 Age -0.0287 0  
## 3 BusinessTravelTravel\_Frequently 1.23 0  
## 4 BusinessTravelTravel\_Rarely 0.471 0  
## 5 DailyRate -0.000285 0  
## 6 DepartmentResearch & Development -0.202 0  
## 7 DepartmentSales 0.238 0  
## 8 DistanceFromHome 0.0363 0  
## 9 Education 0.0115 0  
## 10 EducationFieldLife Sciences -0.194 0  
## # ℹ 36 more rows

And I will plot the variables. If the ridge specification worked, all variables should trend toward 0.

plot(ridge\_fit$fit, xvar = "lambda")



Good, all coefficients trend toward 0. This is what ridge regression was designed to do, so that’s what I want to see.

Now I will split the data into my training set and my test set. I will use the strata argument to split my data stratified based on Attrition, since it is an unbalanced sample (far more people staying than quitting). I will also set up my data splits for my v-fold cross validation here. I will call the tidymodels package for the splitting and cross-validation fold-making process.

library(tidymodels)

## ── Attaching packages ────────────────────────────────────── tidymodels 1.1.0 ──

## ✔ broom 1.0.5 ✔ rsample 1.1.1  
## ✔ dials 1.2.0 ✔ tune 1.1.1  
## ✔ infer 1.0.4 ✔ workflows 1.1.3  
## ✔ modeldata 1.1.0 ✔ workflowsets 1.0.1  
## ✔ recipes 1.0.6 ✔ yardstick 1.2.0

data\_split <- initial\_split(newdata, strata = "Attrition")  
data\_train <- training(data\_split)  
data\_test <- testing(data\_split)  
data\_fold <- vfold\_cv(data\_train, v = 10)

Now I can create a recipe for my ridge regression and ensure that all necessary preprocessing for my training data is taking place, including building dummy variables, normalizing predictors, etc. I will also include the step\_zv (zero variance) step, even though I am working with the dataset that doens’t include the three variables with no variance. It is possible that I chould have split by data using the initial dataset and run this recipe and everything would work out from there, but for now I will leave this as is and ask about this later on.

ridge\_recipe <-   
 recipe(formula = Attrition ~ ., data = data\_train) %>%   
 step\_novel(all\_nominal\_predictors()) %>%   
 step\_dummy(all\_nominal\_predictors()) %>%   
 step\_zv(all\_predictors()) %>%   
 step\_normalize(all\_predictors())

And I will adjust my ridge specification to now be set to ‘penalty = tune()’ so the tune grid knows the penalty parameter should be tuned.

ridge\_spec <- logistic\_reg(mixture = 0, penalty = tune()) %>%  
 set\_mode("classification") %>%  
 set\_engine("glmnet")

And I can create my workflow, adding my recipe and model set to ridge.

ridge\_workflow <- workflow() %>%   
 add\_recipe(ridge\_recipe) %>%   
 add\_model(ridge\_spec)

I will now create a penalty grid to determine the values of the penalty. The code below will create a grid of evenly spaced parameter values. Since I don’t have any prior information about what values will work, I will set the levels to 50 to cast a wider net for this go-around.

penalty\_grid <- grid\_regular(penalty(range = c(-5, 5)), levels = 50)  
penalty\_grid

## # A tibble: 50 × 1  
## penalty  
## <dbl>  
## 1 0.00001   
## 2 0.0000160  
## 3 0.0000256  
## 4 0.0000409  
## 5 0.0000655  
## 6 0.000105   
## 7 0.000168   
## 8 0.000268   
## 9 0.000429   
## 10 0.000687   
## # ℹ 40 more rows

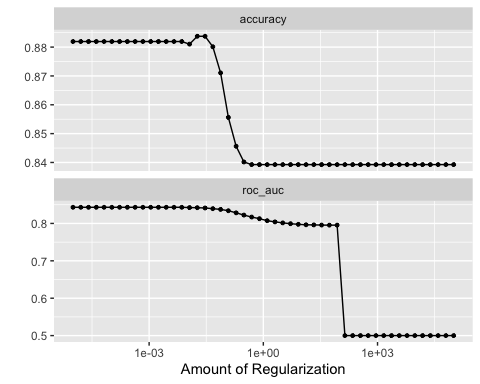
That gives me what I need to fit the model to everything I have specified so far. This is my v-fold cross-validation in action.

tune\_res <- tune\_grid(  
 ridge\_workflow,  
 resamples = data\_fold,   
 grid = penalty\_grid  
)  
tune\_res

## # Tuning results  
## # 10-fold cross-validation   
## # A tibble: 10 × 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [990/111]> Fold01 <tibble [100 × 5]> <tibble [0 × 3]>  
## 2 <split [991/110]> Fold02 <tibble [100 × 5]> <tibble [0 × 3]>  
## 3 <split [991/110]> Fold03 <tibble [100 × 5]> <tibble [0 × 3]>  
## 4 <split [991/110]> Fold04 <tibble [100 × 5]> <tibble [0 × 3]>  
## 5 <split [991/110]> Fold05 <tibble [100 × 5]> <tibble [0 × 3]>  
## 6 <split [991/110]> Fold06 <tibble [100 × 5]> <tibble [0 × 3]>  
## 7 <split [991/110]> Fold07 <tibble [100 × 5]> <tibble [0 × 3]>  
## 8 <split [991/110]> Fold08 <tibble [100 × 5]> <tibble [0 × 3]>  
## 9 <split [991/110]> Fold09 <tibble [100 × 5]> <tibble [0 × 3]>  
## 10 <split [991/110]> Fold10 <tibble [100 × 5]> <tibble [0 × 3]>

I will visualize this more easily with the autoplot() function.

autoplot(tune\_res)



One interesting trend to note is that both accuracy and my ROC area under the curve (AUC) actually decrease as the amount of regularization increases. I want both accuracy and AUC to be as high as possible. This could possibly be due to the fact that ridge regression is not recommended for logistic models, or it could be due to other factors like the regularization increasing bias and/or information loss from the coefficients approaching 0.

Regardless, this helps me visualize the process of regularization, and how changing the value for my penalty will change the metrics of my model. I will also expect the optimal penalty to be quite low (since the higher penalty decreases both accuracy and AUC).

I can collect raw data about these metrics here.

collect\_metrics(tune\_res)

## # A tibble: 100 × 7  
## penalty .metric .estimator mean n std\_err .config   
## <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 0.00001 accuracy binary 0.882 10 0.00619 Preprocessor1\_Model01  
## 2 0.00001 roc\_auc binary 0.843 10 0.0202 Preprocessor1\_Model01  
## 3 0.0000160 accuracy binary 0.882 10 0.00619 Preprocessor1\_Model02  
## 4 0.0000160 roc\_auc binary 0.843 10 0.0202 Preprocessor1\_Model02  
## 5 0.0000256 accuracy binary 0.882 10 0.00619 Preprocessor1\_Model03  
## 6 0.0000256 roc\_auc binary 0.843 10 0.0202 Preprocessor1\_Model03  
## 7 0.0000409 accuracy binary 0.882 10 0.00619 Preprocessor1\_Model04  
## 8 0.0000409 roc\_auc binary 0.843 10 0.0202 Preprocessor1\_Model04  
## 9 0.0000655 accuracy binary 0.882 10 0.00619 Preprocessor1\_Model05  
## 10 0.0000655 roc\_auc binary 0.843 10 0.0202 Preprocessor1\_Model05  
## # ℹ 90 more rows

The same pattern emerges (both accuracy and AUC decreasing with a higher penalty).

Knowing that ridge is probably not what I will ultimately choose, I can still select the optimal penalty (it will probably be quite low) for ridge. I can do this according to either of the two metrics: accuracy or AUC. I chose AUC, primarily because accuracy does not always give the most comprehensive picture of a model’s predictive accuracy, specfically in very imbalanced datasets, which is what I’m dealing with (since significantly more people stay in the organization than leave the organization).

best\_penalty\_auc <- select\_best(tune\_res, metric = "roc\_auc")  
best\_penalty\_auc

## # A tibble: 1 × 2  
## penalty .config   
## <dbl> <chr>   
## 1 0.00001 Preprocessor1\_Model01

I will now apply this best-fit penalty to my workflow.

ridge\_final <- finalize\_workflow(ridge\_workflow, best\_penalty\_auc)  
ridge\_final\_fit <- fit(ridge\_final, data = data\_train)

And I will see how well this performs with my testing set.

augment(ridge\_final\_fit, new\_data = data\_test) %>%  
 roc\_auc(truth = Attrition, .pred\_Yes, event\_level = "second")

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 roc\_auc binary 0.811

After some trial and error with the augment code (I found that the second line of the code needed to be configured differently for the logistic model), I got this to work. I am not sure if I can fully trust the results, but the estimate I got was 0.811. I will compare this to lasso and see how things look there. So in a nutshell, my ridge regression has a mixture of 0, penalty of 0.00001, and a predictive performance metric of 0.811 on my test data.

**Now I will see if the lasso can improve this performance.**

My code for lasso will look quite similar to ridge, except that to specify lasso, I will set my mixture to 1 instead of 0 (1 denotes lasso, 0 denotes ridge, and anything between 0 and 1 denotes an elasticnet).

First things first, I will build my recipe. Once more, I want to normalize my predictors, ensure the setup of dummy variables, etc.

lasso\_recipe <-   
 recipe(formula = Attrition ~ ., data = data\_train) %>%   
 step\_novel(all\_nominal\_predictors()) %>%   
 step\_dummy(all\_nominal\_predictors()) %>%   
 step\_zv(all\_predictors()) %>%   
 step\_normalize(all\_predictors())

Now I will create my lasso specification.

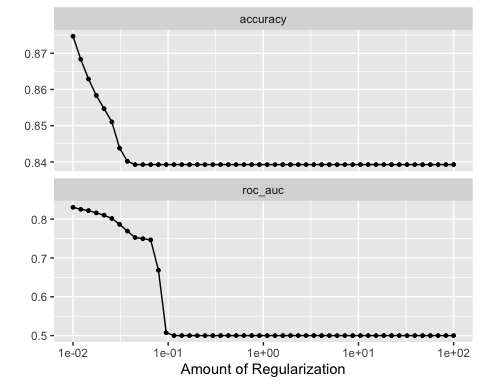
lasso\_spec <-   
 logistic\_reg(penalty = tune(), mixture = 1) %>%   
 set\_mode("classification") %>%   
 set\_engine("glmnet")   
lasso\_workflow <- workflow() %>%   
 add\_recipe(lasso\_recipe) %>%   
 add\_model(lasso\_spec)

And I will set up my penalty grid to determine what the penalty value should be.

penalty\_grid <- grid\_regular(penalty(range = c(-2, 2)), levels = 50)

I will now tune and check out the autoplot to visualize the results.

tune\_res <- tune\_grid(  
 lasso\_workflow,  
 resamples = data\_fold,   
 grid = penalty\_grid  
)  
autoplot(tune\_res)

 Interestingly, the AUC looks quite similar to that of the ridge, but accuracy does seem to drop off much sooner in the lasso than it did in ridge. Currently, I am using AUC as my metric to focus on, so this may not influence the performance as much. Once more, AUC will likely make the most sense as the metric since I’m dealing with an imbalanced dataset.

Now I will make sure the tuning parameter is tuning my penalty based on the most optimal AUC metric.

best\_penalty\_auc <- select\_best(tune\_res, metric = "roc\_auc")  
best\_penalty\_auc

## # A tibble: 1 × 2  
## penalty .config   
## <dbl> <chr>   
## 1 0.01 Preprocessor1\_Model01

And I will finalize fitting my lasso with the tuned penalty to my workflow.

lasso\_final <- finalize\_workflow(lasso\_workflow, best\_penalty\_auc)  
lasso\_final\_fit <- fit(lasso\_final, data = data\_train)

And I can see how well the lasso specification performs.

augment(lasso\_final\_fit, new\_data = data\_test) %>%  
 roc\_auc(truth = Attrition, .pred\_Yes, event\_level = "second")

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 roc\_auc binary 0.796

I got a value of 0.796, which is is actually slightly worse performance-wise than the ridge. Very interesting! I will hope that an elasticnet specification will help improve performance more. So my results from lasso are: mixture of 1, penalty of 0.01, predictive performance of 0.796.

Now last but not least, **I will take a look at elasticnet,** which is a blend of ridge and lasso. Based on what I’ve read, elasticnet is a suitable option for logistic.

The steps will look quite similar to ridge and lasso, but my mixture (in the code below this one) will not be 0 or 1. For now, I will set it to .5 to see how that performs.

elnet\_recipe <-   
 recipe(formula = Attrition ~ ., data = data\_train) %>%   
 step\_novel(all\_nominal\_predictors()) %>%   
 step\_dummy(all\_nominal\_predictors()) %>%   
 step\_zv(all\_predictors()) %>%   
 step\_normalize(all\_predictors())

This will be my elasticnet workflow.

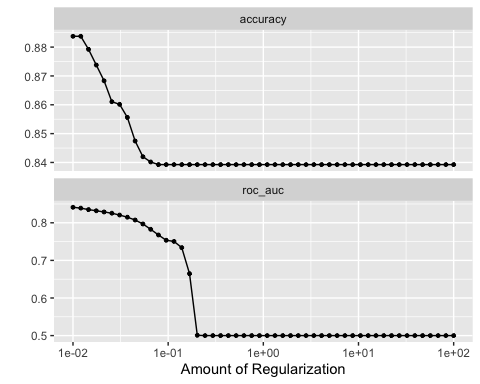
elnet\_spec <-   
 logistic\_reg(penalty = tune(), mixture = .5) %>%   
 set\_mode("classification") %>%   
 set\_engine("glmnet")   
elnet\_workflow <- workflow() %>%   
 add\_recipe(elnet\_recipe) %>%   
 add\_model(elnet\_spec)

And my penalty grid once more. It’s possible I can adjust this to less than 50 levels, but I will leave it at 50 for now to maintain consistency.

penalty\_grid <- grid\_regular(penalty(range = c(-2, 2)), levels = 50)

And I will tune my grid and look at the autoplot.

tune\_res <- tune\_grid(  
 elnet\_workflow,  
 resamples = data\_fold,   
 grid = penalty\_grid  
)  
autoplot(tune\_res)



Now once more, I will use AUC to determine the optimal penalty for my elasticnet specification.

best\_penalty\_auc <- select\_best(tune\_res, metric = "roc\_auc")  
best\_penalty\_auc

## # A tibble: 1 × 2  
## penalty .config   
## <dbl> <chr>   
## 1 0.01 Preprocessor1\_Model01

And I will finalize fitting my elasticnet with the tuned penalty to my workflow.

elnet\_final <- finalize\_workflow(elnet\_workflow, best\_penalty\_auc)  
elnet\_final\_fit <- fit(elnet\_final, data = data\_train)

And I can see how well the elasticnet specification performs.

augment(elnet\_final\_fit, new\_data = data\_test) %>%  
 roc\_auc(truth = Attrition, .pred\_Yes, event\_level = "second")

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 roc\_auc binary 0.807

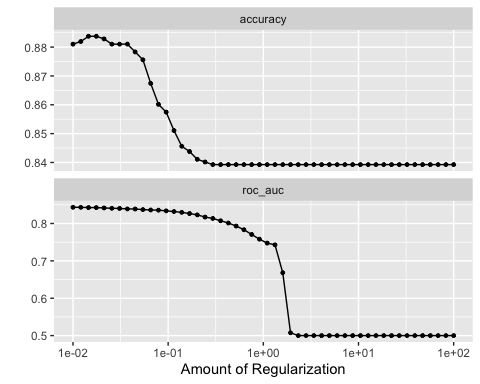
The result is 0.807. Slightly better than lasso, slightly worse than ridge.

I have copied the code above and will run through a few iterative processes, adjusting the value set for the mixture, just to see how this shifts the results (for the sake space, I will only leave up the code that correponds to the best fitting mixture value, but I will note the values that I tried below).

elnet\_spec <-   
 logistic\_reg(penalty = tune(), mixture = .05) %>%   
 set\_mode("classification") %>%   
 set\_engine("glmnet")   
elnet\_workflow <- workflow() %>%   
 add\_recipe(elnet\_recipe) %>%   
 add\_model(elnet\_spec)

penalty\_grid <- grid\_regular(penalty(range = c(-2, 2)), levels = 50)

tune\_res <- tune\_grid(  
 elnet\_workflow,  
 resamples = data\_fold,   
 grid = penalty\_grid  
)  
autoplot(tune\_res)



best\_penalty\_auc <- select\_best(tune\_res, metric = "roc\_auc")  
best\_penalty\_auc

## # A tibble: 1 × 2  
## penalty .config   
## <dbl> <chr>   
## 1 0.01 Preprocessor1\_Model01

elnet\_final <- finalize\_workflow(elnet\_workflow, best\_penalty\_auc)  
elnet\_final\_fit <- fit(elnet\_final, data = data\_train)

augment(elnet\_final\_fit, new\_data = data\_test) %>%  
 roc\_auc(truth = Attrition, .pred\_Yes, event\_level = "second")

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 roc\_auc binary 0.811

mixture = 0.65; penalty = 0.01; estimate (performance) = 0.806

mixture = 0.4; penalty = 0.012; estimate (performance) = 0.806

mixture = 0.25; penalty = 0.01; estimate (performance) = 0.810

mixture = 0.05; penalty = 0.01; estimate (performance) = 0.811

It was interesting to see how adjusting the value for the mixture would change the results (but only slightly). I also noted that the grids would produce somehwat different results as well - notably that there was an uptick in accuracy for some of the above specifications, including the chosen specificaion of a mixture of 0.05.

Based on all of the above information gleaned from looking at ridge, lasso, and elasticnet specifications, **I have deemed that elasticnet is the optimal option for regularization for my logistic model.** Although the performance was comparable for elasticnet and ridge, I have chosen elasticnet as a better option since ridge is not recommended for classification of a binary outcome variable (logistic).

**My final results for optimal regularization of my data are as follows:**

**Elasticnet**

**Mixture of 0.05**

**Penalty of 0.01**

**Performance of 0.811**

Also, I did some sleuthing to see if there was a method for determining which variables would be removed for ‘underperforming’ in either a lasso or elasticnet specification, but I was not able to get any of the code to work. I will note this as a question to ask later, and I will continue exploring potential options to find these results in the meantime.

Additionally, I attempted to find a workable code to gather confusion matrices for the different regularization options, but I unfortunately was not able to find one that worked. The conf\_mat() function did not work, but I could be inputting the variables incorrectly. I think this would help me better determine the performance of each option, so I hope to learn more about how to access this information in the near future.