DS 6030 HW04 Classification

Ben Wilson

‘09/21/2022

Contents


```
#set up R
knitr::opts_chunk$set(echo = TRUE)
data.dir = 'https://mdporter.github.io/DS6030/data/' # data directory
library(R6030)     # functions for SYS-6030
library(tidyverse) # functions for data manipulation
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-4
```

```
library(yardstick)
```

```
## For binary classification, the first factor level is assumed to be the event.
## Use the argument 'event_level = "second"' to alter this as needed.
##
## Attaching package: 'yardstick'
##
## The following object is masked from 'package:readr':
##
##     spec
```

Crime Linkage

Load data for R

```
#load data
linkage_test <- read.csv("C:\\Users\\brwil\\Desktop\\SY MSDS\\DS 6030 Stat Learning\\Week 5\\linkage_te

linkage_train <- read.csv("C:\\Users\\brwil\\Desktop\\SY MSDS\\DS 6030 Stat Learning\\Week 5\\linkage_t
```

Setting for R

```
#-- Settings
K = 10            # number of CV folds
M = 20             # number of simulations
n.folds = 10
```

Establish x,y values

```
#create glmnet matrix
Split = sample(c(rep(0, 0.7 * nrow(linkage_train)),
                 rep(1, 0.3 * nrow(linkage_train))))

Split.train = linkage_train[Split == 0, ]
Split.test = linkage_train[Split == 1, ]

X.train = makeX(Split.train %>% select(-y))
Y.train = Split.train$y

X.test = makeX(Split.test %>% select(-y))
Y.test = Split.test$y

#create folds
fold = sample(rep(1:n.folds, length=nrow(X.train)))
```

1a. Fit a penalized linear regression model to predict linkage. Use a lasso, ridge, or elasticnet penalty (your choice).

Report the value of alpha used (if elasticnet) Report the value of lambda used Report the estimated coefficients

Identify optimal alpha

```r
#loop lambda values
models <- list()
for (i in 0:20) {
  name <- paste0("alpha", i/20)


  models[[name]] <-
    cv.glmnet(X.train, Y.train, type.measure="mse", alpha=i/20,
              family="gaussian")
}

#predict results
results <- data.frame()
for (i in 0:20) {
  name <- paste0("alpha", i/20)

  ## Use each model to predict 'y' given the Testing dataset
  predicted <- predict(models[[name]],
                       s=models[[name]]$lambda.1se, newx=X.test)

  ## Calculate the Mean Squared Error...
  mse <- mean((Y.test - predicted)^2)

  ## Store the results
  temp <- data.frame(alpha=i/20, mse=mse, name=name)
  results <- rbind(results, temp)
}

#print results
print(results)
```

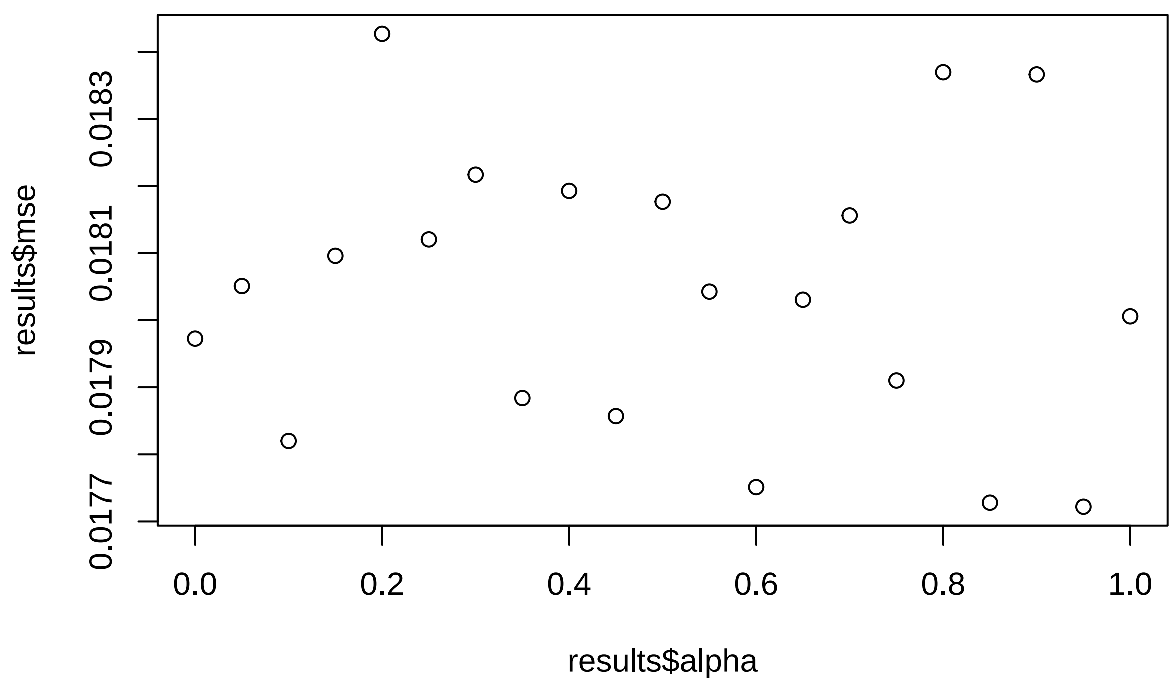
```
##    alpha        mse      name
## 1   0.00 0.01797252    alpha0
## 2   0.05 0.01805089 alpha0.05
## 3   0.10 0.01782000  alpha0.1
## 4   0.15 0.01809596 alpha0.15
## 5   0.20 0.01842683  alpha0.2
## 6   0.25 0.01812039 alpha0.25
## 7   0.30 0.01821691  alpha0.3
## 8   0.35 0.01788393 alpha0.35
## 9   0.40 0.01819278  alpha0.4
## 10  0.45 0.01785701 alpha0.45
## 11  0.50 0.01817655  alpha0.5
## 12  0.55 0.01804256 alpha0.55
## 13  0.60 0.01775117  alpha0.6
## 14  0.65 0.01803053 alpha0.65
## 15  0.70 0.01815610  alpha0.7
## 16  0.75 0.01791009 alpha0.75
## 17  0.80 0.01836956  alpha0.8
## 18  0.85 0.01772802 alpha0.85
## 19  0.90 0.01836633  alpha0.9
## 20  0.95 0.01772198 alpha0.95
```

```
## 21  1.00 0.01800572    alpha1
```

```
#min results
results %>% slice_min(mse)
```

```
##   alpha        mse      name
## 1  0.95 0.01772198 alpha0.95
```

```
#plot results
plot(results$alpha, results$mse)
```



Capture optimal lambda

```
alpha = 0.7         # glmnet tuning alpha (1 = lasso, 0 = ridge)

#capture lambda valyes
lambda_values = tibble()
MSE_values = tibble()


for(m in 1:M) {

#Build Training Models using cross-validation
  lasso_cv = cv.glmnet(X.train, Y.train, alpha = alpha, nfolds = K)
```

```r
  #get lambda that minimizes cv error and 1 SE rule
  min_lambda = lasso_cv$lambda.min
  se_lambda = lasso_cv$lambda.1se


  #Predict y values for test data (for each model: min, 1SE)
  yhat_min = predict(lasso_cv, X.test, s = "lambda.min")
  yhat_1se = predict(lasso_cv, X.test, s = "lambda.1se")

  #evaluate predictions
  MSE_min = mean((Y.test - yhat_min)^2)
  MSE_1se = mean((Y.test - yhat_1se)^2)
  MSE_values = rbind(MSE_values, c(MSE_min, MSE_1se))

}

#update table names
names(lambda_values)[1] <- "min"
```

```
## Warning: The `value` argument of `names<-` must have the same length as `x` as of tibble 3.0.0.
## `names` must have length 0, not 1.
```

```r
names(MSE_values)[1] <- "min"
names(MSE_values)[2] <- "SE1"

#return values
colMeans(MSE_values)
```

```
##        min        SE1
## 0.01711176 0.01795274
```

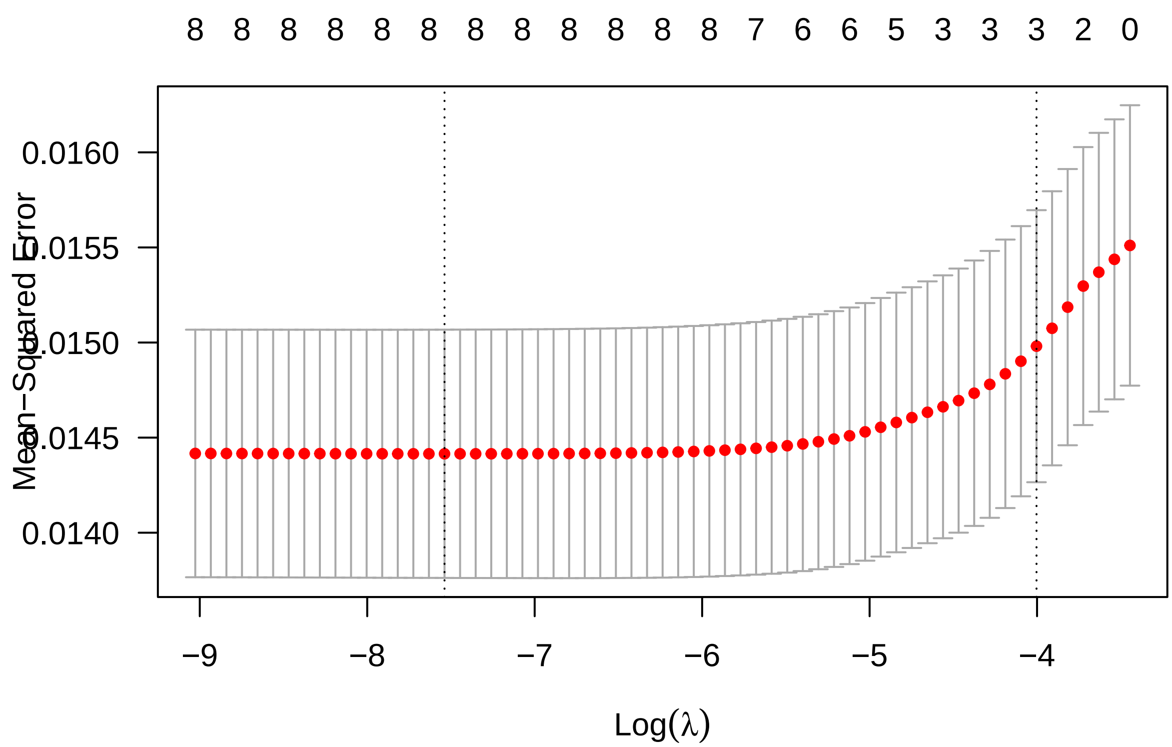
```r
t.test(MSE_values$min, MSE_values$SE1, paired = TRUE, alternative = "two.sided")
```

```
##
##  Paired t-test
##
## data:  MSE_values$min and MSE_values$SE1
## t = -27.8, df = 19, p-value < 2.2e-16
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -0.0009042897 -0.0007776568
## sample estimates:
## mean difference
##   -0.0008409732
```

```r
#fit linear regression model
fit.lm = cv.glmnet(X.train, Y.train, alpha = alpha, family = "gaussian", nfolds = K)
lm_fit = glmnet(X.train, Y.train, alpha = alpha, lambda = "lambda.min", family = "gaussian")
```

```
## Warning in glmnet(X.train, Y.train, alpha = alpha, lambda = "lambda.min", : NAs
## introduced by coercion
```

```r
#plot fit of elastic net
plot(fit.lm, las = 1)
```




```r
#report coefficients
fit.lm
```

```
##
## Call:  cv.glmnet(x = X.train, y = Y.train, nfolds = K, alpha = alpha,      family = "gaussian")
##
## Measure: Mean-Squared Error
##
##       Lambda Index Measure        SE Nonzero
## min 0.000532    45 0.01442 0.0006527       8
## 1se 0.018257     7 0.01498 0.0007152       3
```

Lambda value used: 0.0005 Alpha value used: 0.7

1b. Fit a penalized logistic regression model to predict linkage. Use a lasso, ridge, or elasticnet penalty (your choice).

Report the value of alpha used (if elasticnet) Report the value of lambda used Report the estimated coefficients

Identify optimal alpha

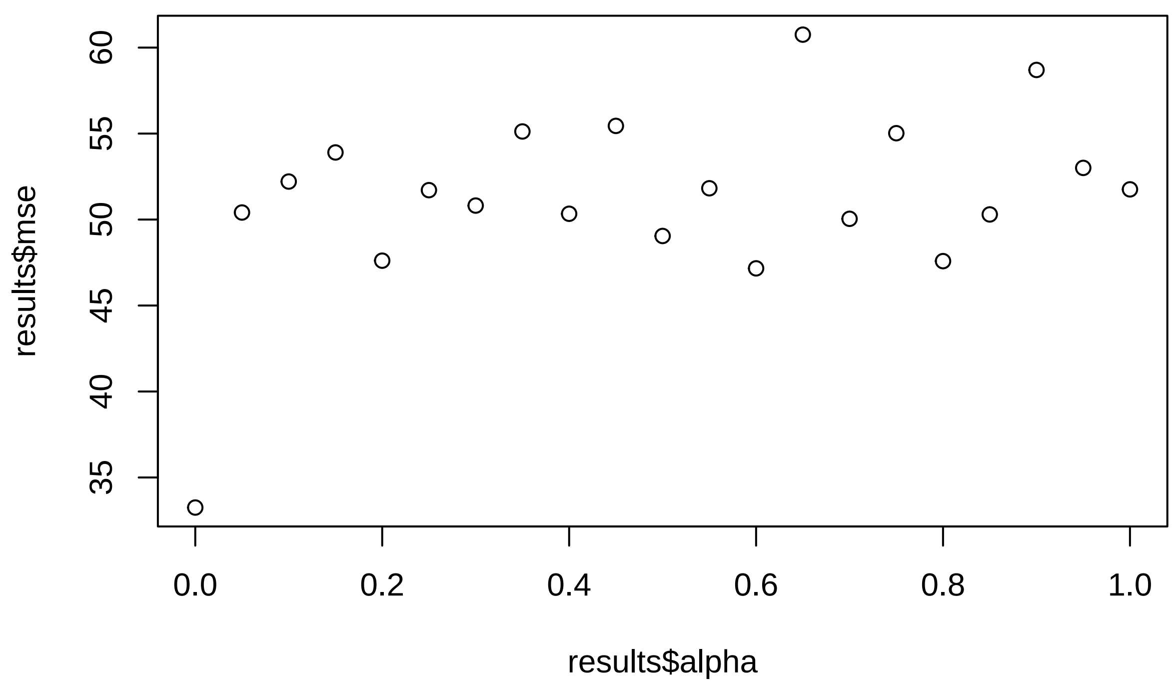
```r
#loop lambda values
models <- list()
for (i in 0:20) {
  name <- paste0("alpha", i/20)


  models[[name]] <-
    cv.glmnet(X.train, Y.train, type.measure="mse", alpha=i/20,
              family="binomial")
}

#predict results
results <- data.frame()
for (i in 0:20) {
  name <- paste0("alpha", i/20)

  ## Use each model to predict 'y' given the Testing dataset
  predicted <- predict(models[[name]],
                       s=models[[name]]$lambda.1se, newx=X.test)

  ## Calculate the Mean Squared Error...
  mse <- mean((Y.test - predicted)^2)

  ## Store the results
  temp <- data.frame(alpha=i/20, mse=mse, name=name)
  results <- rbind(results, temp)
}

#print results
print(results)
```

```
##    alpha      mse      name
## 1   0.00 33.25161    alpha0
## 2   0.05 50.41054 alpha0.05
## 3   0.10 52.20987  alpha0.1
## 4   0.15 53.90198 alpha0.15
## 5   0.20 47.60923  alpha0.2
## 6   0.25 51.71040 alpha0.25
## 7   0.30 50.81271  alpha0.3
## 8   0.35 55.12256 alpha0.35
## 9   0.40 50.33712  alpha0.4
## 10  0.45 55.44644 alpha0.45
## 11  0.50 49.04534  alpha0.5
## 12  0.55 51.82003 alpha0.55
## 13  0.60 47.16170  alpha0.6
## 14  0.65 60.75347 alpha0.65
## 15  0.70 50.04347  alpha0.7
## 16  0.75 55.02219 alpha0.75
## 17  0.80 47.58193  alpha0.8
## 18  0.85 50.29746 alpha0.85
## 19  0.90 58.70052  alpha0.9
## 20  0.95 53.00503 alpha0.95
## 21  1.00 51.75644    alpha1
```

```r
#min results
results %>% slice_min(mse)
```

```
##   alpha      mse   name
## 1     0 33.25161 alpha0
```

```r
#plot results
plot(results$alpha, results$mse)
```



Capture optimal lambda

```r
alpha = 0.0          # glmnet tuning alpha (1 = lasso, 0 = ridge)

#capture lambda valyes
lambda_values = tibble()
MSE_values = tibble()


for(m in 1:M) {

#Build Training Models using cross-validation
  lasso_cv = cv.glmnet(X.train, Y.train, alpha = alpha, nfolds = K)


 #get lambda that minimizes cv error and 1 SE rule
```

```r
  min_lambda = lasso_cv$lambda.min
  se_lambda = lasso_cv$lambda.1se


  #Predict y values for test data (for each model: min, 1SE)
  yhat_min = predict(lasso_cv, X.test, s = "lambda.min")
  yhat_lse = predict(lasso_cv, X.test, s = "lambda.1se")

  #evaluate predictions
  MSE_min = mean((Y.test - yhat_min)^2)
  MSE_lse = mean((Y.test - yhat_lse)^2)
  MSE_values = rbind(MSE_values, c(MSE_min, MSE_lse))

}

#update table names
names(lambda_values)[1] <- "min"
```

```
## Warning: The `value` argument of `names<-` must have the same length as `x` as of tibble 3.0.0.
## `names` must have length 0, not 1.
```

```r
names(MSE_values)[1] <- "min"
names(MSE_values)[2] <- "SE1"

#return values
colMeans(MSE_values)
```

```
##        min        SE1
## 0.01711257 0.01805713
```

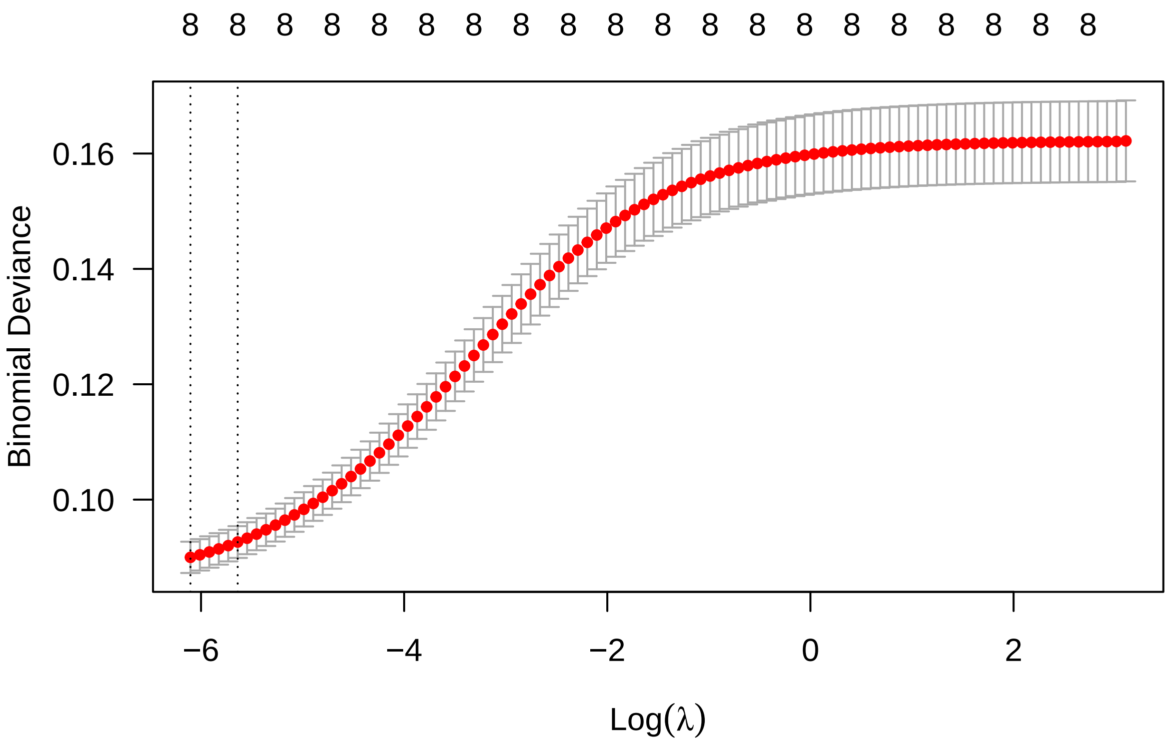
```r
t.test(MSE_values$min, MSE_values$SE1, paired = TRUE, alternative = "two.sided")
```

```
##
##  Paired t-test
##
## data:  MSE_values$min and MSE_values$SE1
## t = -19.443, df = 19, p-value = 5.32e-14
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -0.0010462408 -0.0008428766
## sample estimates:
## mean difference
##   -0.0009445587
```

```r
#fit elastic net
fit.enet = cv.glmnet(X.train, Y.train, alpha = alpha, family = "binomial", nfolds = K)
enet_fit = glmnet(X.train, Y.train, alpha = alpha, lambda = "lambda.min", family = "binomial")
```

```
## Warning in glmnet(X.train, Y.train, alpha = alpha, lambda = "lambda.min", : NAs
## introduced by coercion
```

```r
#plot fit of elastic net
plot(fit.enet, las = 1)
```




```r
#print coefficients
fit.enet
```

```
##
## Call:  cv.glmnet(x = X.train, y = Y.train, nfolds = K, alpha = alpha,      family = "binomial")
##
## Measure: Binomial Deviance
##
##       Lambda Index Measure       SE Nonzero
## min 0.002233   100 0.09000 0.002712       8
## 1se 0.003556    95 0.09265 0.002751       8
```

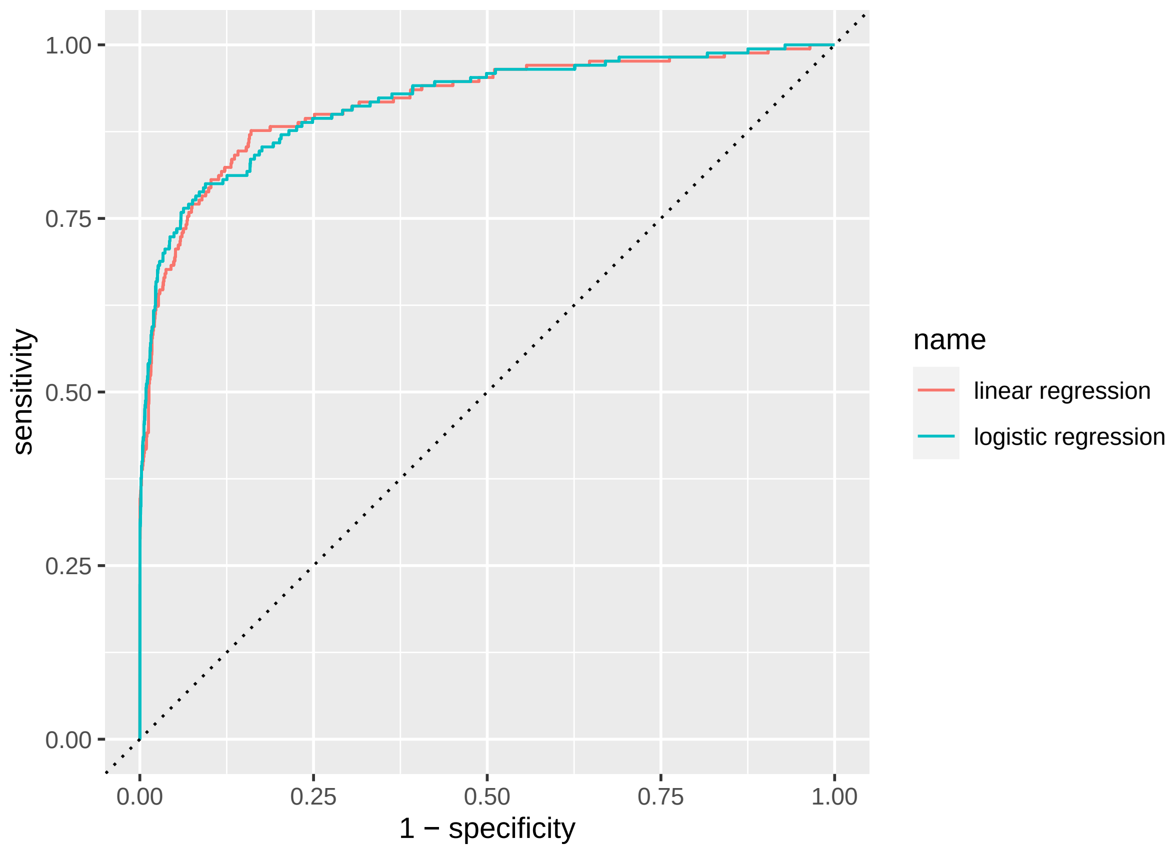
Lambda value used: 0.0000538 Alpha value used: 0.0

#1c. Produce one plot that has the ROC curves, using the training data, for both models (from part a and b). Use color and/or linetype to distinguish between models and include a legend.

ROC Outputs

```r
#Get predictions (of p(x)) on test data
p.hat_lm = predict(fit.lm, X.test, type='response')
```

```r
p.hat_lr = predict(fit.enet, X.test, type='response')

#Make Hard classification (use .10 as cut-off)
G.hat_lm = ifelse(p.hat_lm >= .10, 1, 0)

G.hat_lr = ifelse(p.hat_lr >= .10, 1, 0)

#Get predictions (of gamma(x)) on test data
gamma_lm = predict(fit.lm, X.test, type='link')[,1]

gamma_lr = predict(fit.enet, X.test, type='link')[,1]

#ROCs
ROC_lm = tibble(truth = factor(Y.test, levels=c(1,0)), gamma_lm) %>%
  yardstick::roc_curve(truth, gamma_lm)

#ROC plots
ROC_lr = tibble(truth = factor(Y.test, levels=c(1,0)), gamma_lr) %>%
  yardstick::roc_curve(truth, gamma_lr)

#add column to name and bind
ROC_lr$name <- 'logistic regression'
ROC_lm$name <- 'linear regression'
ROC <- rbind(ROC_lm, ROC_lr)

#plot ROC
ROC %>%
  ggplot(aes(1-specificity, sensitivity)) + geom_line(aes(color = name)) +
  geom_abline(lty=3) +
  coord_equal()
```



1d. Recreate the ROC curve from the penalized logistic regression model using repeated hold-out data. The following steps will guide you:

Fix alpha=.75 Run the following steps 25 times: - Hold out 500 observations - Use the remaining observations to estimate lambda using 10-fold CV - Predict the probability of linkage for the 500 hold-out observations - Store the predictions and hold-out labels Combine the results and produce the hold-out based ROC curve Note: by estimating lambda each iteration, we are incorporating the uncertainty present in estimating that tuning parameter.

```
#create new conditions
alpha = 0.75 #new alpha
M = 25 # number of simulations
K = 10

#create list to capture new results
p_hat_list = vector("list", length = M)

#for loop to generate phat values
for (m in 1:M){
  #hold out 500 values
  Split.D = sample(c(rep(0, nrow(linkage_train)-500),
                     rep(1,500)))

  #perform test train split
  Split.trainD = linkage_train[Split.D == 0,]
```

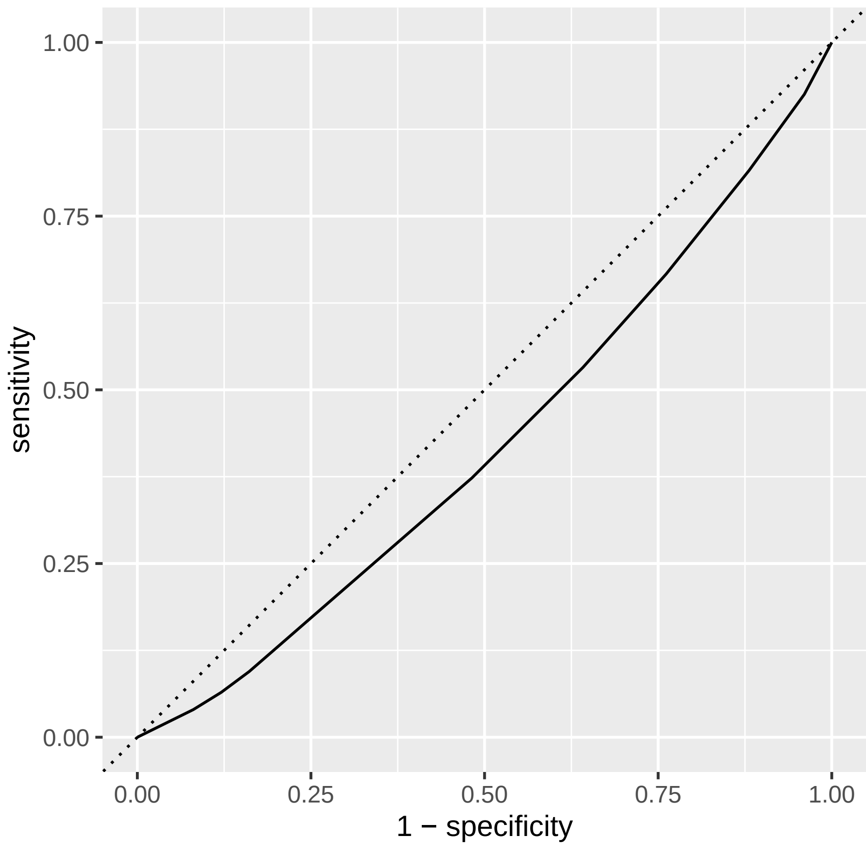
```
  Split.testD = linkage_train[Split.D == 1, ]

  #generate new x and y values for train
  X.trainD = glmnet::makeX(Split.trainD %>% select(-y))
  Y.trainD = Split.trainD$y

  #generate new x and y values for test
  X.testD = glmnet::makeX(Split.testD %>% select(-y))
  Y.testD = Split.testD$y

  fold = sample(rep(1:n.folds, length=nrow(X.trainD)))

  #retrain models with new x and y values
  enet_fitD = glmnet(X.trainD, Y.trainD, alpha = alpha,
                  lambda = "lambda.min", family = "binomial")

  #generate probability for 500 observations
  p_hatD = predict(enet_fitD, X.testD, type = 'response')[,1]

  #create df for storing results
  p_hat_lists = tibble(p_hatD, true_label = Y.testD) %>%
    mutate(sim = m)

  #store results in df
  p_hat_list[[m]] = p_hat_lists

}
```

```
## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
```

```
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion

## Warning in glmnet(X.trainD, Y.trainD, alpha = alpha, lambda = "lambda.min", :
## NAs introduced by coercion
```

```
#bind two lists for phat
p_hat_finalD = bind_rows(p_hat_list)

#store phat results for ROC curve
```

```
p_hat_ROCD = p_hat_finalD$p_hatD

#df for ROC curve results
ROC_d = tibble(truth = factor(p_hat_finalD$true_label, levels = c(1,0)), p_hat_ROCD) %>%
  yardstick::roc_curve(truth, p_hat_ROCD)

#visualize ROC curve
ROC_d %>%
  ggplot(aes(1-specificity, sensitivity)) + geom_line() +
  geom_abline(lty=3) +
  coord_equal()
```



1e. Contest Part 1: Predict the estimated probability of linkage for the test data (using any model).

- Submit a .csv file (ensure comma separated format) named lastname_firstname_1.csv that includes the column named p that is your estimated posterior probability. We will use automated evaluation, so the format must be exact.
- You are free to use any tuning parameters
- You are free to use any data transformation or feature engineering
- You will receive credit for a proper submission; the top five scores will receive 2 bonus points.
- Your probabilities will be evaluated with respect to the mean negative Bernoulli log-likelihood (known as the average log-loss metric)

```
fit_lasso = cv.glmnet(X.train, Y.train, alpha = 1, family = binomial)

p_hat = predict(fit_lasso, X.test, type="response", s = "lambda.min")

#export csv file
write.csv(p_hat,"C:\\Users\\brwil\\Desktop\\SY MSDS\\DS 6030 Stat Learning\\Week 5\\wilson_benjamin_1.cs
```

1f. Contest Part 2: Predict the linkages for the test data (using any model).

- Submit a .csv file (ensure comma separated format) named lastname_firstname_2.csv that includes the column named linkage that takes the value of 1 for linkages and 0 for unlinked pairs. We will use automated evaluation, so the format must be exact.
- You are free to use any tuning parameters.
- You are free to use any data transformation or feature engineering.
- Your labels will be evaluated based on total cost, where cost is equal to $1FP + 8$FN. This implies that False Negatives (FN) are 8 times as costly as False Positives (FP)
- You will receive credit for a proper submission; the top five scores will receive 2 bonus points. Note: you only will get bonus credit for one of the two contests.

```
linkage = predict(fit_lasso, X.test, type="response", s = "lambda.min")
linkage_tibble = tibble(linkage)
names(linkage_tibble)[1] <- "linkage"

#export csv file
write.csv(p_hat,"C:\\Users\\brwil\\Desktop\\SY MSDS\\DS 6030 Stat Learning\\Week 5\\wilson_benjamin_2.cs
```