COMP 4442 Final Exam, Winter 2022

Krista Miller

knitr::opts\_chunk$set(echo = TRUE)  
  
library(glmnet)

## Warning: package 'glmnet' was built under R version 4.0.5

## Loading required package: Matrix

## Loaded glmnet 4.1-3

library(MASS)  
library(survival)  
library(survminer)

## Warning: package 'survminer' was built under R version 4.0.5

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.5

## Loading required package: ggpubr

## Warning: package 'ggpubr' was built under R version 4.0.5

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':  
##   
## select

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

# Load any additional packages, if any, that you use as part of your answers here

There are four questions on this midterm, all of which have multiple parts. Please be sure to provide answers to all parts of each question. Each question has an associated .csv file, which you will load into memory at the beginning of the question. All of the included data sets are simulated, so any results should not be taken as evidence for or against the existence of anything in the real world. The data were simulated to minimize the ambiguity and messiness that typifies real data. If you feel that something is ambiguous in a way that impedes your ability to answer the questions, please let me know.

Almost there - you can do it!

## Question 1 - Train/validate/test (30 points total)

For this question, a 400 observation data set was split such that 50% of the data was assigned to a training set, 25% was assigned to a validation set, and the remaining 25% was assigned to a test set. The outcome variable is y, which is contained in all three data sets, and is continuous. The predictor variables, also contained in all three data sets, are all continuous.

Run the code chunk below to load the data into memory before beginning your work on this question.

q1.train <- read.csv("q1\_train.csv", header=TRUE, sep=",") # Loads the CSV file into memory. You may need to adapt this line to work on your computer  
  
q1.valid <- read.csv("q1\_valid.csv", header=TRUE, sep=",") # Loads the CSV file into memory. You may need to adapt this line to work on your computer  
  
q1.test <- read.csv("q1\_test.csv", header=TRUE, sep=",") # Loads the CSV file into memory. You may need to adapt this line to work on your computer  
  
# Check variable types   
  
str(q1.train)

## 'data.frame': 200 obs. of 11 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ y : num -4.59 11.39 8.94 -3.16 -6.79 ...  
## $ x1: num 0.734 -1.121 0.804 0.455 0.19 ...  
## $ x2: num -0.436 0.715 -0.55 0.442 -0.539 ...  
## $ x3: num -0.4641 -1.3285 0.0387 2.3444 -1.5627 ...  
## $ x4: num 0.953 0.763 0.832 -1.153 0.272 ...  
## $ x5: num 1.3568 -0.0246 -1.0233 1.815 -1.9222 ...  
## $ x6: num 0.386 -0.514 0.867 -0.974 -0.57 ...  
## $ x7: num -2.524 1.779 0.183 -0.884 -0.422 ...  
## $ x8: num 1.2597 0.8249 0.6353 -0.7715 0.0262 ...  
## $ x9: num 0.853 -0.5235 0.224 0.0744 1.169 ...

str(q1.valid)

## 'data.frame': 100 obs. of 11 variables:  
## $ X : int 201 202 203 204 205 206 207 208 209 210 ...  
## $ y : num -5.27 1.03 5.71 5.96 5.87 ...  
## $ x1: num 0.484 -0.154 1.398 0.134 -0.498 ...  
## $ x2: num -0.7664 1.1196 -0.2013 -0.6587 0.0931 ...  
## $ x3: num 0.659 0.431 -0.112 0.583 0.595 ...  
## $ x4: num 0.211 -1.014 0.176 -0.635 -0.34 ...  
## $ x5: num 1.456 -1.187 -0.808 0.597 0.538 ...  
## $ x6: num 0.774 -2.073 -1.691 0.204 1.343 ...  
## $ x7: num -1.2134 -0.7085 -0.0487 0.8162 0.3041 ...  
## $ x8: num -0.379 -0.68 0.175 -0.512 -0.374 ...  
## $ x9: num 0.498 0.638 0.566 -1.977 -0.487 ...

str(q1.test)

## 'data.frame': 100 obs. of 11 variables:  
## $ X : int 301 302 303 304 305 306 307 308 309 310 ...  
## $ y : num -1.5 10.92 7.95 21.58 14.16 ...  
## $ x1: num 0.2025 -0.3654 0.2806 -0.026 0.0843 ...  
## $ x2: num 0.644 -0.674 -0.443 0.416 0.184 ...  
## $ x3: num -1.035 -1.253 -0.27 0.251 0.248 ...  
## $ x4: num 0.4752 -1.4235 -0.0929 -0.413 -0.329 ...  
## $ x5: num -0.163 1.855 1.497 -0.149 -0.317 ...  
## $ x6: num 1.5316 0.6652 1.1081 -0.5957 0.0307 ...  
## $ x7: num -1.45 1.726 -0.157 1.232 0.31 ...  
## $ x8: num -1.053 0.825 -1.296 -2.1 0.781 ...  
## $ x9: num -2.227 0.928 -0.21 0.444 -0.328 ...

# Q1, Part 1 - Generating candidate models on the training set (15 points)

You will generate four candidate models: a model obtained through forward selection, a model obtained through backward selection, a cross-validated ridge regression model, and a cross-validated lasso regression model. For both cross-validated models, use the model associated with lambda.1se. You will fit all of these models to the training data (q1.train), using y as the outcome and x1-x9 as the predictors. Please do not include interactions or squares in your pool of potential predictors. Once you have done this, answer the four questions below.

Forward selection:

# Code to set up your training data for forward selection  
glm.null<- glm(y~1, data= q1.train)  
all<-glm(y~., data= q1.train)  
  
# Code to conduct forward selection - be sure to include trace=1  
  
model.forward <- step(glm.null, direction= 'forward', scope=formula(all), trace=1)

## Start: AIC=1316.16  
## y ~ 1  
##   
## Df Deviance AIC  
## + x7 1 6082.2 1256.5  
## + x2 1 7509.4 1298.7  
## + x9 1 7771.4 1305.5  
## + x8 1 7823.1 1306.9  
## + x6 1 7922.1 1309.4  
## + x1 1 8046.2 1312.5  
## + x3 1 8167.6 1315.5  
## <none> 8276.9 1316.2  
## + x5 1 8215.6 1316.7  
## + x4 1 8264.1 1317.8  
## + X 1 8267.3 1317.9  
##   
## Step: AIC=1256.54  
## y ~ x7  
##   
## Df Deviance AIC  
## + x2 1 5361.6 1233.3  
## + x6 1 5726.2 1246.5  
## + x1 1 5903.3 1252.6  
## + x3 1 5958.2 1254.4  
## + x5 1 5962.3 1254.5  
## <none> 6082.2 1256.5  
## + x8 1 6025.2 1256.7  
## + x9 1 6039.1 1257.1  
## + X 1 6054.2 1257.6  
## + x4 1 6082.1 1258.5  
##   
## Step: AIC=1233.32  
## y ~ x7 + x2  
##   
## Df Deviance AIC  
## + x6 1 5122.0 1226.2  
## + x8 1 5251.1 1231.2  
## + x9 1 5271.3 1231.9  
## + x5 1 5282.2 1232.3  
## <none> 5361.6 1233.3  
## + X 1 5334.9 1234.3  
## + x1 1 5356.5 1235.1  
## + x4 1 5360.0 1235.2  
## + x3 1 5361.6 1235.3  
##   
## Step: AIC=1226.17  
## y ~ x7 + x2 + x6  
##   
## Df Deviance AIC  
## + x9 1 5016.5 1224.0  
## + x8 1 5038.5 1224.9  
## <none> 5122.0 1226.2  
## + x5 1 5088.5 1226.9  
## + X 1 5096.2 1227.2  
## + x1 1 5116.6 1228.0  
## + x4 1 5116.9 1228.0  
## + x3 1 5120.8 1228.1  
##   
## Step: AIC=1224.01  
## y ~ x7 + x2 + x6 + x9  
##   
## Df Deviance AIC  
## <none> 5016.5 1224.0  
## + x8 1 4971.9 1224.2  
## + x5 1 4982.2 1224.6  
## + X 1 4985.9 1224.8  
## + x4 1 5005.8 1225.6  
## + x1 1 5010.1 1225.8  
## + x3 1 5014.9 1226.0

# Display the final model selected by forward selection here  
  
summary(model.forward)

##   
## Call:  
## glm(formula = y ~ x7 + x2 + x6 + x9, data = q1.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -15.7632 -3.1881 0.5546 3.2557 16.1706   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.3638 0.3637 11.998 < 2e-16 \*\*\*  
## x7 2.9305 0.3735 7.846 2.76e-13 \*\*\*  
## x2 2.0546 0.4089 5.025 1.13e-06 \*\*\*  
## x6 1.1549 0.3670 3.147 0.00191 \*\*   
## x9 0.8273 0.4086 2.025 0.04426 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 25.72585)  
##   
## Null deviance: 8276.9 on 199 degrees of freedom  
## Residual deviance: 5016.5 on 195 degrees of freedom  
## AIC: 1224  
##   
## Number of Fisher Scoring iterations: 2

Backward selection:

# Code to set up your training data for backward selection  
glm.null<- glm(y~1, data= q1.train)  
all<-glm(y~., data= q1.train)  
  
# Code to conduct backward selection - be sure to include trace=1  
  
model.backward <- step(all, direction= 'backward', trace=1)

## Start: AIC=1231.25  
## y ~ X + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9  
##   
## Df Deviance AIC  
## - x1 1 4899.6 1229.3  
## - x3 1 4904.0 1229.5  
## - x4 1 4914.5 1229.9  
## - X 1 4917.0 1230.0  
## - x5 1 4930.2 1230.5  
## - x8 1 4934.5 1230.7  
## <none> 4898.6 1231.2  
## - x9 1 4977.8 1232.5  
## - x6 1 5097.7 1237.2  
## - x2 1 5369.8 1247.6  
## - x7 1 6270.0 1278.6  
##   
## Step: AIC=1229.29  
## y ~ X + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9  
##   
## Df Deviance AIC  
## - x3 1 4905.2 1227.5  
## - x4 1 4916.4 1228.0  
## - X 1 4919.2 1228.1  
## - x5 1 4930.5 1228.5  
## - x8 1 4938.0 1228.9  
## <none> 4899.6 1229.3  
## - x9 1 4978.1 1230.5  
## - x6 1 5098.7 1235.3  
## - x2 1 5510.8 1250.8  
## - x7 1 6272.6 1276.7  
##   
## Step: AIC=1227.52  
## y ~ X + x2 + x4 + x5 + x6 + x7 + x8 + x9  
##   
## Df Deviance AIC  
## - x4 1 4922.7 1226.2  
## - X 1 4924.8 1226.3  
## - x5 1 4932.1 1226.6  
## - x8 1 4943.9 1227.1  
## <none> 4905.2 1227.5  
## - x9 1 4983.1 1228.7  
## - x6 1 5102.0 1233.4  
## - x2 1 5573.9 1251.1  
## - x7 1 6280.0 1274.9  
##   
## Step: AIC=1226.23  
## y ~ X + x2 + x5 + x6 + x7 + x8 + x9  
##   
## Df Deviance AIC  
## - X 1 4941.8 1225.0  
## - x5 1 4949.6 1225.3  
## - x8 1 4955.9 1225.6  
## <none> 4922.7 1226.2  
## - x9 1 4996.3 1227.2  
## - x6 1 5113.8 1231.8  
## - x2 1 5580.1 1249.3  
## - x7 1 6364.0 1275.6  
##   
## Step: AIC=1225.01  
## y ~ x2 + x5 + x6 + x7 + x8 + x9  
##   
## Df Deviance AIC  
## - x5 1 4971.9 1224.2  
## - x8 1 4982.2 1224.6  
## <none> 4941.8 1225.0  
## - x9 1 5010.3 1225.8  
## - x6 1 5130.2 1230.5  
## - x2 1 5601.0 1248.0  
## - x7 1 6370.2 1273.8  
##   
## Step: AIC=1224.22  
## y ~ x2 + x6 + x7 + x8 + x9  
##   
## Df Deviance AIC  
## - x8 1 5016.5 1224.0  
## <none> 4971.9 1224.2  
## - x9 1 5038.5 1224.9  
## - x6 1 5201.5 1231.2  
## - x2 1 5648.6 1247.7  
## - x7 1 6377.8 1272.0  
##   
## Step: AIC=1224.01  
## y ~ x2 + x6 + x7 + x9  
##   
## Df Deviance AIC  
## <none> 5016.5 1224.0  
## - x9 1 5122.0 1226.2  
## - x6 1 5271.3 1231.9  
## - x2 1 5666.2 1246.4  
## - x7 1 6600.4 1276.9

# Display the final model selected by backward selection here  
  
summary(model.backward)

##   
## Call:  
## glm(formula = y ~ x2 + x6 + x7 + x9, data = q1.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -15.7632 -3.1881 0.5546 3.2557 16.1706   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.3638 0.3637 11.998 < 2e-16 \*\*\*  
## x2 2.0546 0.4089 5.025 1.13e-06 \*\*\*  
## x6 1.1549 0.3670 3.147 0.00191 \*\*   
## x7 2.9305 0.3735 7.846 2.76e-13 \*\*\*  
## x9 0.8273 0.4086 2.025 0.04426 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 25.72585)  
##   
## Null deviance: 8276.9 on 199 degrees of freedom  
## Residual deviance: 5016.5 on 195 degrees of freedom  
## AIC: 1224  
##   
## Number of Fisher Scoring iterations: 2

Cross-validated ridge regression (lambda.1se); please use the provided random seed

# Code to set up your training data for cross-validated ridge regression  
  
x<- model.matrix(y~x1+x2+x3+x4+x5+x6+x7+x8+x9, q1.train)  
x<- x[,-1]  
y<- q1.train$y  
  
set.seed(123456)  
  
# Code to conduct cross-validated ridge regression here  
  
cv.ridge <- cv.glmnet(x, y, alpha=0) #alpha=0 for ridge regression  
  
# Display lambda.1se  
cv.ridge$lambda.1se

## [1] 8.596511

# Display the coefficients associated with lambda.1se   
coef(cv.ridge, s= "lambda.1se")

## 10 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 4.3578745  
## x1 0.2646891  
## x2 0.8429917  
## x3 0.1350471  
## x4 -0.1438606  
## x5 0.1943090  
## x6 0.5042810  
## x7 1.2469858  
## x8 0.4388385  
## x9 0.5079313

Cross-validated lasso regression (lambda.1se); please use the provided random seed

# Code to set up your training data for cross-validated lasso regression  
x2<- model.matrix(y~x1+x2+x3+x4+x5+x6+x7+x8+x9, q1.train)  
x2<- x2[,-1]  
y2<- q1.train$y  
  
set.seed(123456)  
  
# Code to conduct cross-validated lasso regression here  
  
cv.lasso <- cv.glmnet(x2, y2, alpha=1) #alpha=1 for lasso regression  
  
# Display lambda.1se  
cv.lasso$lambda.1se

## [1] 1.084757

# Display the coefficients associated with lambda.1se   
coef(cv.lasso, s= "lambda.1se")

## 10 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 4.4082736  
## x1 .   
## x2 0.9160066  
## x3 .   
## x4 .   
## x5 .   
## x6 0.1436171  
## x7 2.1563372  
## x8 .   
## x9 .

1. What predictors are included in the model selected by forward selection?

Your answer here: x7, x2, x6, x9

1. What predictors are included in the model selected by backward selection?

Your answer here: x2, x6, x7, x9

1. What predictors are included in the cross-validated ridge model (lambda.1se)?

Your answer here: x1, x2, x3, x4, x5, x6, x7, x8, x9

1. What predictors are included in the cross-validated lasso model (lambda.1se)?

Your answer here: x2, x6, x7

# Q1, Part 2 - Fitting candidate models on the validation set and choosing a model (10 points)

Now, use the validation data (q1.valid) to generate new predictions for the four candidate models obtained in the previous part. Use these to compute the mean squared prediction error to evaluate how well each candidate model (which was estimated using training data) predicts the y’s in the validation data set. Once you have done this, answer the three questions below.

Forward model:

# Obtain new predictions using the validation set  
  
cvpred.forward<- predict(model.forward, q1.valid, type="response")  
  
# Compute mean squared prediction error for the forward model  
  
mspe.forward<- mean((q1.valid$y-cvpred.forward)^2)  
mspe.forward

## [1] 24.01926

Backward model:

# Obtain new predictions using the validation set  
  
cvpred.backward<- predict(model.backward, q1.valid, type="response")  
  
# Compute mean squared prediction error for the backward model  
  
mspe.backward.valid <- mean((q1.valid$y-cvpred.backward)^2)  
mspe.backward.valid

## [1] 24.01926

Cross-validated ridge model (lambda.1se):

# Obtain new predictions using the validation set  
  
ridge.valid<- model.matrix(y~x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9, q1.valid)  
ridge.valid<- ridge.valid[,-1]  
  
ridge.valid.pred<- predict(cv.ridge, ridge.valid)  
  
# Compute mean squared prediction error for the cross-validated ridge model  
  
mspe.ridge.valid <- mean((q1.valid$y-ridge.valid.pred)^2)  
mspe.ridge.valid

## [1] 26.41709

Cross-validated lasso model (lambda.1se):

# Obtain new predictions using the validation set  
lasso.valid<- model.matrix(y~x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9, q1.valid)  
lasso.valid<- lasso.valid[,-1]  
  
lasso.valid.pred<- predict(cv.lasso, lasso.valid)  
  
# Compute mean squared prediction error for the cross-validated lasso model  
  
mspe.lasso.valid <- mean((q1.valid$y-lasso.valid.pred)^2)  
mspe.lasso.valid

## [1] 27.47134

1. List the mean squared prediction error for each of your candidate models here:

Forward model mean squared prediction error: 24.01 Backward model mean squared prediction error: 24.01 Ridge model mean squared prediction error: 26.42 Lasso model mean squared prediction error: 27.47

1. Based on the mean squared prediction error, which model do you choose?

Your answer here: forward

1. Display the coefficient values for the model you chose here:

# Display candidate model chosen based on performance on the validation set here  
summary(model.forward)

##   
## Call:  
## glm(formula = y ~ x7 + x2 + x6 + x9, data = q1.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -15.7632 -3.1881 0.5546 3.2557 16.1706   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.3638 0.3637 11.998 < 2e-16 \*\*\*  
## x7 2.9305 0.3735 7.846 2.76e-13 \*\*\*  
## x2 2.0546 0.4089 5.025 1.13e-06 \*\*\*  
## x6 1.1549 0.3670 3.147 0.00191 \*\*   
## x9 0.8273 0.4086 2.025 0.04426 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 25.72585)  
##   
## Null deviance: 8276.9 on 199 degrees of freedom  
## Residual deviance: 5016.5 on 195 degrees of freedom  
## AIC: 1224  
##   
## Number of Fisher Scoring iterations: 2

# Q1, Part 3 - Testing the final model on the test set (5 points)

Now that you’ve chosen a model, you will test this model’s generalizability by generating new predictions for the final model chosen in the previous part. Use these new predictions to compute the mean squared prediction error to evaluate how well the final model (which was estimated using training data) predicts the y’s in the test data set. Once you’ve done this, answer the question below:

# Obtain new predictions using the test set  
  
cvpred.final<- predict(model.forward, q1.test, type="response")  
  
# Compute mean squared prediction error for the final model  
  
mspe.final <- mean((q1.test$y-cvpred.forward)^2)  
mspe.final

## [1] 62.3001

1. What is the mean squared prediction error for the final model making predictions on the test set?

Your answer here: 62.30

## Question 2 - Evaluating the generalizability of a “shrunk” logistic regression model (30 points total)

For this question, a different 400-observation data set was split such that 70% of it was assigned to a training set and the remaining 30% was assigned to a test set. The outcome variable is y, which is binary. The predictor variables, V1 - V10, are all continuous.

Run the code chunk below to load the data into memory before beginning your work on this question.

log.train <- read.csv("logistic\_train.csv", header=TRUE, sep=",") # Loads the CSV file into memory. You may need to adapt this line to work on your computer  
  
log.test <- read.csv("logistic\_test.csv", header=TRUE, sep=",") # Loads the CSV file into memory. You may need to adapt this line to work on your computer  
  
str(log.train) # Note: y is an int type; don't change this for this question

## 'data.frame': 300 obs. of 11 variables:  
## $ y : int 1 0 0 0 0 0 0 0 1 0 ...  
## $ V1 : num 0.8063 -0.0893 1.0258 -2.1566 0.0688 ...  
## $ V2 : num 0.19978 0.4983 0.39872 -1.39152 0.00962 ...  
## $ V3 : num 0.787 0.143 1.129 -2.023 -0.213 ...  
## $ V4 : num 0.1045 0.0849 1.2643 -1.4822 -0.4206 ...  
## $ V5 : num 0.637 0.212 0.968 -2.209 0.216 ...  
## $ V6 : num -0.19688 -0.48276 0.00274 -0.91685 -0.01067 ...  
## $ V7 : num 0.02089 0.29542 0.71478 -0.4424 -0.00614 ...  
## $ V8 : num 0.0153 0.1858 0.3519 -2.1991 -0.6784 ...  
## $ V9 : num 0.0874 -0.0845 1.0442 -1.1866 0.5171 ...  
## $ V10: num -0.00807 0.32335 -0.78685 -2.53471 -0.61956 ...

str(log.test) # Note: y is an int type; don't change this for this question

## 'data.frame': 100 obs. of 11 variables:  
## $ y : int 0 0 1 1 0 1 0 0 0 0 ...  
## $ V1 : num 0.778 0.151 2.052 0.587 0.824 ...  
## $ V2 : num 0.172 -1.104 1.255 -0.425 0.804 ...  
## $ V3 : num 0.511 -0.478 2.042 0.318 0.853 ...  
## $ V4 : num 0.224 -0.665 1.927 -0.507 1.073 ...  
## $ V5 : num 0.1185 -0.0299 1.7304 0.5088 0.965 ...  
## $ V6 : num 1.381 0.505 1.046 0.752 -0.566 ...  
## $ V7 : num -1.081 -1.53 0.775 -0.132 0.041 ...  
## $ V8 : num 0.41 -0.411 1.349 0.613 0.255 ...  
## $ V9 : num 0.4726 0.135 1.5779 0.0577 0.6249 ...  
## $ V10: num 0.503 0.321 0.5 1.271 0.384 ...

# Q2, Part 1 - Fitting the “shrunk” logistic model (5 points)

I fitted a penalized logistic regression model to the training data; that is, I applied a method of coefficient shrinkage to a logistic regression model fitted to the training. Run the code chunk below to fit this model and answer the two questions below.

x.train.mat<-model.matrix(y~.,data=log.train)  
x.train.matrix<-x.train.mat[,-1]   
  
y.train.vec<-log.train$y  
  
set.seed(12345)  
  
cvfit.log = cv.glmnet(x=x.train.matrix, y=y.train.vec, family="binomial", alpha=1)   
coef(cvfit.log, s = "lambda.min")

## 11 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 0.11202810  
## V1 0.15054756  
## V2 .   
## V3 .   
## V4 0.52991688  
## V5 .   
## V6 .   
## V7 0.12957868  
## V8 0.03803014  
## V9 .   
## V10 0.09262162

1. What type of penalized/“shrunk” logistic regression model did I fit to the training data? Be specific.

Your answer here: cross-validated lasso regression

1. Which predictors (not including the intercept) are in the lambda.min penalized/“shrunk” logistic regression model?

Your answer here (list the included predictors): V1, V4, V7, V8, V10

# Q2, Part 2 - Obtaining new predictions from test data (5 points)

In this section, you will generate new predictions of Y based on the values of the predictors in the test set. That is, you will use the predictor values from log.test to predict new y values.

x.test.mat<-model.matrix(y~.,data=log.test)  
x.test.matrix<-x.test.mat[,-1]   
  
probs<-cvfit.log %>% predict(newx = x.test.matrix, type= "response")  
y.test.pred <- ifelse(probs > 0.5, 1, 0)

Next, display the counts of both the predicted y values and the actual y values in log.test. Remember that these are binary variables. Hint: this will help you check your work in the next part.

# Counts for predicted y's  
  
table(y.test.pred)

## y.test.pred  
## 0 1   
## 35 65

# Counts for actual y's  
  
table(log.test$y)

##   
## 0 1   
## 47 53

# Q2, Part 3 - Evaluating the generalizability on the test set (20 points)

For this part, you will create a confusion matrix and compute the four model indices we discussed in class: accuracy, precision, recall, and F1 score. Use the table() function to create the confusion matrix and be sure to display it in the knitted document. Write your code for this below and answer the two questions below.

## Your code for creating and displaying the confusion matrix  
  
confusion.matrix <- table(Actual = log.test$y, Predicted = y.test.pred)  
confusion.matrix

## Predicted  
## Actual 0 1  
## 0 24 23  
## 1 11 42

## Your code for computing and displaying the model indices   
  
# Code to compute accuracy  
  
accuracy <- sum(diag(confusion.matrix))/sum(confusion.matrix)  
accuracy

## [1] 0.66

# Code to compute precision  
  
precision <- confusion.matrix[2,2]/sum(confusion.matrix[,2])  
precision

## [1] 0.6461538

# Code to compute recall  
  
recall <- confusion.matrix[2,2]/sum(confusion.matrix[2,])  
recall

## [1] 0.7924528

# Code to compute F1 score  
  
F1 = 2\*(precision\*recall)/(precision+recall)  
F1

## [1] 0.7118644

1. Fill in the values of your confusion matrix. Note that you should also have the confusion matrix displayed as part of the output from your code.

* True positives (your answer here): 42 False positives (your answer here): 23 True negatives (your answer here): 24 False negatives (your answer here): 11

1. Fill in the values of your model indices. Again, be sure that you also have these displayed as part of the output from your code.

* Accuracy: 0.66 Precision: 0.64 Recall: 0.79 F1 score: 0.71

## Question 3 - Modeling count data (20 points total)

Alyssa analyzed workplace injury incidents obtained from a shipping warehouse. She sampled 45 employee records at random and recorded the number of workplace injury reports in the employee’s file in the past five years (“reports”) and how many years since the employee underwent safety training certification (“years”).

Run the code chunk below to load the data into memory before beginning your work on this question.

injury <- read.csv("injury.csv", header=TRUE, sep=",") # Loads the CSV file into memory. You may need to adapt this line to work on your computer  
  
str(injury)

## 'data.frame': 300 obs. of 3 variables:  
## $ ID : int 3 4 14 16 38 50 52 54 57 71 ...  
## $ reports: int 1 0 6 5 1 3 3 4 1 0 ...  
## $ years : int 0 5 6 5 4 4 9 4 6 4 ...

# Q3, Part 1 - Describing the data (10 points)

Before starting the analysis, Alyssa conducted some descriptive analyses to present alongside her modeling work. Please recreate the descriptive analyses she conducted by answering the following three questions. Any code you use to answer the questions should go into the chunk below:

# Code for your descriptive analyses here  
#percentage of employees with zero or one injury in their file:  
zero\_or\_one\_injuries<- as.table(table(injury$reports<2))  
prop.table(zero\_or\_one\_injuries)

##   
## FALSE TRUE   
## 0.4888889 0.5111111

#among workers who had 6 or more injury reports, what was the median number of years since their most recent safety certification?  
six\_or\_more\_injuries<- filter(injury, reports>5)  
median(six\_or\_more\_injuries$years)

## [1] 8.5

#among workers who had a safety certification that was less than 5 years old, what was the median number of injury reports?  
safety\_less\_than\_5\_yrs<- filter(injury, years<5)  
median(safety\_less\_than\_5\_yrs$reports)

## [1] 1

1. What percentage of the sampled employees had zero or one injury report in their files?

Your answer here: 51.1%

1. Among workers who had 6 or more injury reports, what was the median number of years since their most recent safety certification?

Your answer here: 8.5 years

1. Among workers who had a safety certification that was less than 5 years old, what was the median number of injury reports?

Your answer here: 1

# Q3, Part 2 - Fitting a quasipoisson model (10 points)

After some exploratory modeling, Alyssa determined that a *quasipoisson* model was the most appropriate model for these data.

First, please re-create her analysis by fitting a quasipoisson model to the data, using reports as the outcome and years as the predictor.

# Code for your model here  
  
model.quasipoisson <- glm(reports~ years, data= injury, family= "quasipoisson")  
summary(model.quasipoisson)

##   
## Call:  
## glm(formula = reports ~ years, family = "quasipoisson", data = injury)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.42630 -1.36940 -0.01387 0.95518 2.76649   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.30079 0.30661 -0.981 0.332   
## years 0.23313 0.04511 5.168 5.83e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for quasipoisson family taken to be 1.828019)  
##   
## Null deviance: 142.323 on 44 degrees of freedom  
## Residual deviance: 91.294 on 43 degrees of freedom  
## (255 observations deleted due to missingness)  
## AIC: NA  
##   
## Number of Fisher Scoring iterations: 5

Next, answer the following three questions about the analysis:

1. Was the number of years since safety training a significant predictor of the number of injury reports?

Your answer here (yes/no): yes

1. Based on the results, does the predicted count of workplace injury reports *increase* or *decrease* as the number of years since an employees most recent safety certification increases? How do you know?

Your answer here: increase

1. Based on your answer to the previous question, do you think Alyssa recommended greater frequency of safety certification or a lower frequency of safety certification?

Your answer here (higher or lower): higher

## Question 4 - Modeling time-to-event data (20 points total)

As part of receiving services at a community mental health clinic, a prospective client must first submit a completed intake form; upon approval, the client may then schedule a first visit with a provider. Ahmed designed an intervention study to determine if different follow-up schedules (“group”) decrease the amount of time in between the client’s approval and the client’s first appointment (“first\_visit”, in decimal days). Clients in Group A, which was the current standard for client follow-up at the time, received a reminder via text message to schedule their first appointment 14 days after approval. Clients in Group B received reminders 14 and 21 days after approval, and clients in Group C received reminders 7 and 14 days after approval. If the client saw a provider within 30 days of approval, “status” is equal to 1; otherwise, it’s equal to zero.

Run the code chunk below to load the data into memory before beginning your work on this question.

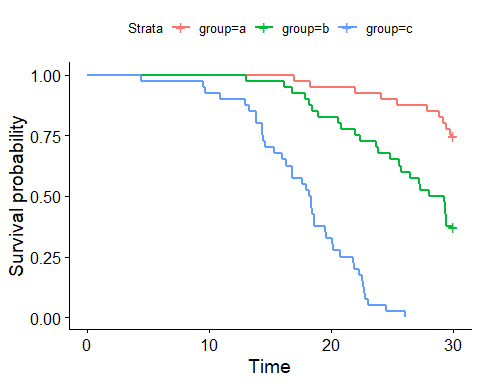
intake <- read.csv("enrollment.csv", header=TRUE, sep=",") # Loads the CSV file into memory. You may need to adapt this line to work on your computer  
  
str(intake)

## 'data.frame': 120 obs. of 3 variables:  
## $ status : int 0 0 1 1 0 0 1 0 0 0 ...  
## $ group : chr "a" "a" "a" "a" ...  
## $ first\_visit: num 30 30 28.8 29.4 30 ...

# Q4, Part 1 - Kaplan-Meier curves and censoring (10 points)

Ahmed started his analysis by fitting Kaplan-Meier curves for each group, which can be seen by running the code chunk below. Please examine this visualization and answer the three questions below.

fit <- survfit(Surv(first\_visit,status) ~ group,  
 data = intake)  
ggsurvplot(fit)



1. Based on what you see in the Kaplan-Meier curves, which group had the most censored observations?

Your answer here: Group A

1. The line for group C ends somewhere between 20 and 30 days. What does this mean for members of group C in terms of their time-to-first visit?

Your answer here: In group C, there is a 0% probability of event (time-to-visit) happening beyond approximately 26 days. In other words, everyone in group C experienced the scheduled first visit event within approximately 26 days.

1. During the design phase, Ahmed took steps to prevent left censoring from occurring. If this had happened and a client’s status was left-censored, what would that have meant with regard to their time-to-first visit?

Your answer here: The patient scheduled a first visit (event) with a provider prior to the intervention (follow up reminders).

# Q4, Part 2: Fitting parametric survival models (10 points)

Next, Ahmed fitted two accelerated failure time (AFT) models: an exponential AFT model and a Weibull AFT model. Run the code chunk below and review the results of the two analyses.

model.weib<-survreg(Surv(first\_visit,status)~group, dist="weibull",data=intake)  
summary(model.weib)

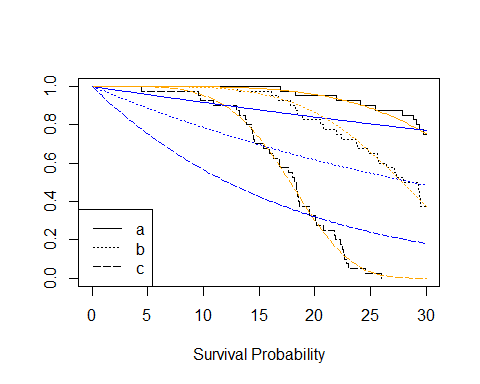
##   
## Call:  
## survreg(formula = Surv(first\_visit, status) ~ group, data = intake,   
## dist = "weibull")  
## Value Std. Error z p  
## (Intercept) 3.6725 0.0732 50.14 <2e-16  
## groupb -0.2670 0.0832 -3.21 0.0013  
## groupc -0.7144 0.0839 -8.52 <2e-16  
## Log(scale) -1.5413 0.0983 -15.68 <2e-16  
##   
## Scale= 0.214   
##   
## Weibull distribution  
## Loglik(model)= -261.7 Loglik(intercept only)= -310.6  
## Chisq= 97.9 on 2 degrees of freedom, p= 5.5e-22   
## Number of Newton-Raphson Iterations: 6   
## n= 120

model.exp<-survreg(Surv(first\_visit,status)~group, dist="exponential",data=intake)  
summary(model.exp)

##   
## Call:  
## survreg(formula = Surv(first\_visit, status) ~ group, data = intake,   
## dist = "exponential")  
## Value Std. Error z p  
## (Intercept) 4.746 0.316 15.01 < 2e-16  
## groupb -1.024 0.374 -2.74 0.0062  
## groupc -1.879 0.354 -5.31 1.1e-07  
##   
## Scale fixed at 1   
##   
## Exponential distribution  
## Loglik(model)= -330.2 Loglik(intercept only)= -348.9  
## Chisq= 37.26 on 2 degrees of freedom, p= 8.1e-09   
## Number of Newton-Raphson Iterations: 4   
## n= 120

He then created a Kaplan-Meier plot with overlaid cumulative distribution curves for both models. The orange cumulative distribution curves are derived from the fitted Weibull AFT model, and the blue curves are from the fitted exponential AFT model. Run the code chunk below to view these and answer the four questions below.

plot(survfit(Surv(first\_visit,status) ~ group,  
 data = intake),lty=c(1,3,5),xlab="Survival Probability")  
legend("bottomleft", c("a", "b","c"), lty = c(1,3,5))  
points(seq(0,30,by=.2),  
 1-psurvreg(seq(0,30,by=.2),mean=model.weib$coefficients[1],  
 scale=model.weib$scale),type="l",lty=1,col="orange")  
points(seq(0,30,by=.2),  
 1-pexp(seq(0,30,by=.2),rate=1/exp(model.exp$coefficients[1])),  
 type="l", lty=1,col="blue")  
points(seq(0,30,by=.2),  
 1-psurvreg(seq(0,30,by=.2),mean=sum(model.weib$coefficients[c(1,2)]),  
 scale=model.weib$scale),type="l",lty=3,col="orange")  
points(seq(0,30,by=.2),  
 1-psurvreg(seq(0,30,by=.2),mean=sum(model.exp$coefficients[c(1,2)]),  
 distribution = "exponential"),type="l",lty=3,col="blue")  
points(seq(0,30,by=.2),  
 1-psurvreg(seq(0,30,by=.2),mean=sum(model.weib$coefficients[c(1,3)]),scale=model.weib$scale),  
 type="l",lty=5,col="orange")  
points(seq(0,30,by=.2),  
 1-psurvreg(seq(0,30,by=.2),mean=sum(model.exp$coefficients[c(1,3)]),  
 distribution = "exponential"),type="l",lty=5,col="blue")



1. Based on the visualization, which of the two models - exponential AFT (blue) or Weibull AFT (orange) - appear to best model the time-to-first visit for the different intervention groups?

Your answer here (exponential or Weibull): Weibull AFT

1. Look at the model output for the model you chose in the previous question. Is there evidence of a difference between Group A and Group B?

Your answer here (yes/no): yes

1. From this same model output, is there evidence of a difference between Group A and Group C?

Your answer here (yes/no): yes

1. Considering the results shown in both parts 1 and 2 of this question, which follow-up schedule - Day 14 only, Days 7 and 14, or Days 14 and 21 - do you think Ahmed recommended to the clinic? Please note that stating the group letter alone here won’t earn full credit; you must explicitly reference the specific intervention the group experienced.

Your answer here: Ahmed likely recommended the follow up schedule that Group C received (text messages on days 7 and 14); belonging to group C accelerates the time to event by a factor of exp(-0.7144)

## All done - you did it! Treat yourself to something nice :)