Problem_Set_9

1.

The way Starbucks formulated its success was through the active advancement of its data analytics capabilities. For a start up donut and cafe, I would advise them to take a similar approach to analyzing data from local businesses as well as their own to formulate the optimal menu. It is incredibly important to approach a personalized experience, similar to starbucks, and create some kind of app that goes directly to the customer and gives them an incentive to spend more money at one of the stores locations. According to the article the app and personlaization resulted in, "36% of U.S. company-operated sales last year and mobile payment was 29 percent of transactions." In regards to other options that I would advise to the company is to keep in mind of what type of donuts seem to sell with what types of coffee. Study what donuts individuals are buying and what coffees people are buying with them. In the morning for instance, you could sell your coffee at the standard price but sell the donuts as a compliment at a discounted price to help boost your overall sales. We could collect this data based on the information that is stored on the app.

2.

a.

```r
library(MASS)
set.seed(1861)
```

b.

```r
meanValue <- mean(Boston$medv)
Boston$priceyHomes <- ifelse(Boston$medv >= meanValue, 1,0)
```

c.

```r
help(Boston)
library(useful)
```

```
## Loading required package: ggplot2
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```r
myFormula <- as.formula(priceyHomes ~ crim  + zn + indus+ chas + nox + rm + age + dis + rad + tax + ptra
xVars <- build.x(myFormula, data = Boston)
yVars <- build.y(myFormula, data = Boston)
LassoMod <- cv.glmnet(x = xVars, y = yVars , alpha =  1 , nfold = 10)
```

```
LassoModMin <- coef(LassoMod, s = "lambda.min")
LassoMod1se <- coef(LassoMod, s = "lambda.1se")
```

d.

```
LassoMod1se
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  1.4349694614
## (Intercept)  .
## crim         .
## zn           0.0002443942
## indus       -0.0050990332
## chas         0.0304131803
## nox         -0.3807754381
## rm           0.1194157037
## age         -0.0028769532
## dis         -0.0453220514
## rad          0.0035906289
## tax          .
## ptratio     -0.0492612684
## black        0.0001222349
## lstat       -0.0244009957
```

e.

In regards to the coneptual differences between the two lambdas, lambda min represents the minimum level or error that lambda can have in a cross validation model. Wheras lambda 1se is a cross validation model that is 1 standard error away from from a model that may over fit the data which is lambda min. So in other words, lambda 1se creates a simpler model.

reference: https://stats.stackexchange.com/questions/70249/feature-selection-model-with-glmnet-on-methylation-data-pn

f.

```
RidgeMod <- cv.glmnet(x = xVars, y = yVars , alpha =  0, nfold = 10)
RidgeModMin <- coef(LassoMod, s = "lambda.min")
RidgeMod1se <- coef(LassoMod, s = "lambda.1se")

RidgeMod1se
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  1.4349694614
## (Intercept)  .
## crim         .
## zn           0.0002443942
## indus       -0.0050990332
## chas         0.0304131803
## nox         -0.3807754381
## rm           0.1194157037
## age         -0.0028769532
## dis         -0.0453220514
## rad          0.0035906289
## tax          .
## ptratio     -0.0492612684
## black        0.0001222349
## lstat       -0.0244009957
```

g.

```
ElasticMod1 <- cv.glmnet(x = xVars, y = yVars , alpha =  0.25)
ElasticMod2 <- cv.glmnet(x = xVars, y = yVars , alpha =  0.5)
ElasticMod3 <- cv.glmnet(x = xVars, y = yVars , alpha =  0.75)
ElasticMod1se <- coef(ElasticMod1, s = "lambda.1se")
ElasticMod2se <- coef(ElasticMod2, s = "lambda.1se")
ElasticMod3se <- coef(ElasticMod3, s = "lambda.1se")
ElasticMod1se
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  8.989552e-01
## (Intercept)  .
## crim         .
## zn           .
## indus       -3.531542e-03
## chas         1.517601e-02
## nox         -1.135455e-01
## rm           1.295564e-01
## age         -2.081379e-03
## dis         -1.593099e-02
## rad          .
## tax          .
## ptratio     -4.045446e-02
## black        5.795378e-05
## lstat       -2.131782e-02
```

```
ElasticMod2se
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  9.022527e-01
## (Intercept)  .
## crim         .
## zn           .
## indus       -2.694434e-03
## chas         .
## nox         -4.588969e-02
## rm           1.263024e-01
## age         -1.948354e-03
## dis         -1.223756e-02
## rad          .
## tax          .
## ptratio     -4.106074e-02
## black        1.432762e-05
## lstat       -2.329662e-02
```

```
ElasticMod3se
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  9.413904e-01
## (Intercept)  .
## crim         .
## zn           .
## indus       -2.512827e-03
## chas         .
## nox         -3.546481e-02
## rm           1.247348e-01
## age         -1.992794e-03
## dis         -1.363053e-02
## rad          .
## tax          .
## ptratio     -4.169646e-02
## black        1.461372e-06
## lstat       -2.427945e-02
```

h.

An elastic model is a model that is between Lasso and Ridge in that each additional variable/attribute carrys a certain weight. This weight is similar to the penalty that is put in place by a Lasso or Ridge model, but instead of a penalty there is a designated weight. The convenience of an elastic model is that the user has the ability to determine how much "weight" they are willing to put on their model in order not to burden it.

i.

```
LassoModMinRound<- round(LassoModMin , digits = 4)
LassoMod1seRound<- round(LassoMod1se , digits = 4)
RidgeModMinRound<- round(RidgeModMin , digits = 4)
RidgeMod1seRound<- round(RidgeMod1se , digits = 4)
ElasticMod1seRound<- round(ElasticMod1se , digits = 4)
ElasticMod2seRound<- round(ElasticMod2se , digits = 4)
ElasticMod3seRound<- round(ElasticMod3se , digits = 4)

cbind(LassoMod1seRound ,RidgeMod1seRound,  ElasticMod1seRound, ElasticMod2seRound,ElasticMod3seRound )
```

```
## 15 x 5 sparse Matrix of class "dgCMatrix"
##                    1       1       1       1       1
## (Intercept)  1.4350  1.4350  0.8990  0.9023  0.9414
## (Intercept)  .       .       .       .       .
## crim         .       .       .       .       .
## zn           0.0002  0.0002  .       .       .
## indus       -0.0051 -0.0051 -0.0035 -0.0027 -0.0025
## chas         0.0304  0.0304  0.0152  .       .
## nox         -0.3808 -0.3808 -0.1135 -0.0459 -0.0355
## rm           0.1194  0.1194  0.1296  0.1263  0.1247
## age         -0.0029 -0.0029 -0.0021 -0.0019 -0.0020
## dis         -0.0453 -0.0453 -0.0159 -0.0122 -0.0136
## rad          0.0036  0.0036  .       .       .
## tax          .       .       .       .       .
## ptratio     -0.0493 -0.0493 -0.0405 -0.0411 -0.0417
## black        0.0001  0.0001  0.0001  0.0000  0.0000
## lstat       -0.0244 -0.0244 -0.0213 -0.0233 -0.0243
```

j.

As alpha increases we notice that variables like "chas", "rad", and "zn" are dropped. This leads us to conclude that further increases in the error margin we no longer see these variables as important factors in our predictions.

k.

```r
#Hall helped out with this equation.
RMSE <- function(truth, predict) {sqrt(mean((truth - predict)^2))}
RMSE(predict(LassoMod, newx = xVars), Boston$priceyHomes)
```

```
## [1] 0.3410317
```

```r
RMSE(predict(RidgeMod, newx = xVars), Boston$priceyHomes)
```

```
## [1] 0.3474512
```

```r
RMSE(predict(ElasticMod1, newx = xVars), Boston$priceyHomes)
```

```
## [1] 0.3496015
```

```r
RMSE(predict(ElasticMod2, newx = xVars), Boston$priceyHomes)
```

```
## [1] 0.3503205
```

```r
RMSE(predict(ElasticMod3, newx = xVars), Boston$priceyHomes)
```

```
## [1] 0.3497895
```

My LassoMod has the lowest mean squared error. But there are super close so maybe they wouldn't be the best to compare in regards to to ridge model.

l.

```r
#install.packages("glmnetUtils")
library("glmnetUtils")
```

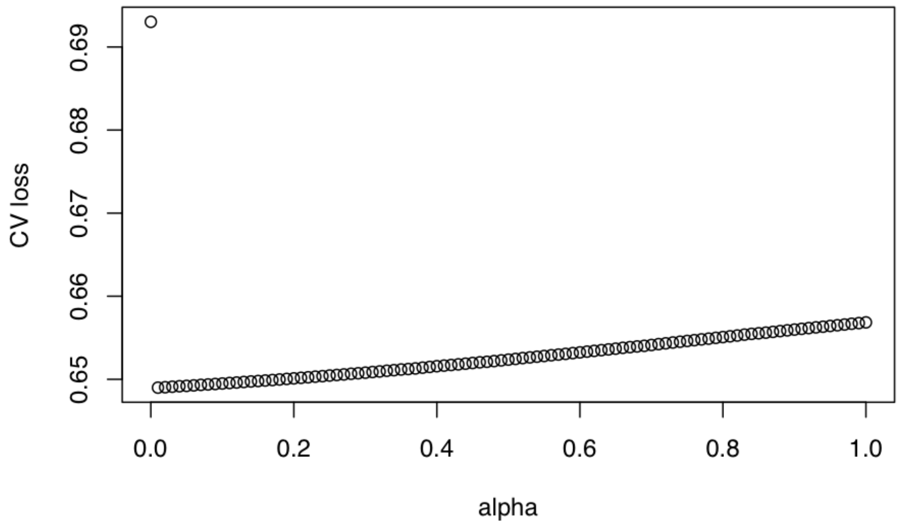
```
##
## Attaching package: 'glmnetUtils'

## The following objects are masked from 'package:glmnet':
##
##     cv.glmnet, glmnet
```

```r
CVAFit <- cva.glmnet(xVars, yVars, nfolds = 10, parallel = TRUE, alpha = seq(0,
1, by = 0.01), family = "binomial")
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```r
minlossplot(CVAFit, cv.type = "min")
```



m.

This plot is a visible indicator of the increasing minimum error of our k fold 10 of our elastic model. At ridge we see that our CV loss is 0.69. To be honest I am not sure what a CV loss is an indicator of. However it is clear that an increaes in alpha leads to a steady increase of cv loss starting from .64 all the way to 0.65.

n

So in comparison with my k values, my LassoMod in both k and with l have a lower CV loss than ridge which has a very high cv loss but not a significantly higher RMSE. Im not sure entirely what is going on but there seems to be a higher CV loss with ridge than there is with Lasso.