CharlesCoffey17835PSET4

Charles Coffey

10/21/2020

QUESTION 1

1.1

```
library(matrixStats)
library(ggplot2)
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.0-2

library(e1071)

house_votes = read.csv("house_votes.csv")

taxbill_vote = data.frame(tapply(house_votes$outcome, house_votes$democrat, F
UN=sum))

colnames(taxbill_vote) = "Votes"
rownames(taxbill_vote) = c("Republican", "Democrat")

num_repub = sum(house_votes$republican)
num_democ = sum(house_votes$democrat)
```

I immediately observe that 0 out of 173 democrats voted for this bill while 191 out of 203 republicans voted for this bill.

1.2

```
repub_house_votes = house_votes[house_votes$republican == 1, ]

set.seed(02139)

training_data_idxs = sort(sample(c(1:203), 160, replace = FALSE)) #finding ra
ndom sample indices for data set

training_data = repub_house_votes[training_data_idxs,]
test_data = repub_house_votes[-training_data_idxs,]

predictors = training_data[,c(499:657)]
```

```
test_data_p = test_data[,c(499:657)] #same subsetted columns as training data
predictors
```

1.3

```
combined_matrix = cbind(training_data$outcome, predictors)

ols_model = lm(combined_matrix$`training_data$outcome` ~. , data = combined_m
atrix)

print(paste("The unadjusted R^2 for the data is 0.894."))

## [1] "The unadjusted R^2 for the data is 0.894."
```

R^2 values tell us how well the data fits the model. This R^2 tells us that data fits to the OLS model relatively well but could be much better.

1.4

```
yhat_ols = predict(ols_model, newdata = test_data_p)

## Warning in predict.lm(ols_model, newdata = test_data_p): prediction from a
rank-
## deficient fit may be misleading

Y_vote_ols = c()
Y_vote_ols[which(yhat_ols >= 0.5)] = 1
Y_vote_ols[which(yhat_ols < 0.5)] = 0

size = length(yhat_ols)

matches_ols = data.frame(Match=integer(size), TruePositive=integer(size), Tru
eNegative=integer(size), FalsePositive=integer(size), FalseNegative=integer(s
ize)) #creating a data frame to keep track of all of the information


for (idx in 1:size){

  if (Y_vote_ols[idx] == test_data$outcome[idx]){ #checking for matches
    matches_ols[idx, "Match"] = 1

    if ((test_data$outcome[idx] == 1) && (Y_vote_ols[idx] == 1)){ #checking f
or true positive
      matches_ols[idx, "TruePositive"] = 1
    }
    if ((test_data$outcome[idx] == 0) && (Y_vote_ols[idx] == 0)){ #checking f
or true negative
      matches_ols[idx, "TrueNegative"] = 1
    }
  }
  else{
```

```
    matches_ols[idx, "Match"] = 0 # non-matches

    if ((test_data$outcome[idx] == 0) && (Y_vote_ols[idx] == 1)){ #checking f
or false positive
      matches_ols[idx, "FalsePositive"] = 1

    }
    if ((test_data$outcome[idx] == 1) && (Y_vote_ols[idx] == 0)){ #checking f
or false negative
      matches_ols[idx, "FalseNegative"] = 1

    }
  }
}

num_correct_pred_ols = sum(matches_ols$Match)
precision = sum(matches_ols$TruePositive)/(sum(matches_ols$TruePositive) + su
m(matches_ols$FalsePositive))
recall = sum(matches_ols$TruePositive)/(sum(matches_ols$TruePositive) + sum(m
atches_ols$FalseNegative))
F1 = 2/((1/precision)+(1/recall))

print(paste("Precision: ", toString(precision)))

## [1] "Precision:  0.973684210526316"

print(paste("Recall: ", toString(recall)))

## [1] "Recall:  0.902439024390244"

print(paste("F1: ", toString(F1)))

## [1] "F1:  0.936708860759494"
```

With an F1 score of 0.94 I can conclude that the OLS model performs well with relatively little errors. This being said, it definitely could be more accurate. Republican defectors would people that were predicted to vote in favor but ended up not. In this case, they would be false positives. The model was able to detect one false positive. I'm not sure if it detected all defectors but it was able to detect at least one.

1.5

```
#implementing support vector machine
svm_model = svm(predictors, training_data$outcome, kernel ="radial")

## Warning in svm.default(predictors, training_data$outcome, kernel = "radial
"):
## Variable(s) 'vote_515' and 'vote_519' and 'vote_521' and 'vote_526' and
## 'vote_534' and 'vote_535' and 'vote_540' and 'vote_547' and 'vote_552' and
## 'vote_553' and 'vote_555' and 'vote_563' and 'vote_574' and 'vote_575' and
```

```
## 'vote_576' and 'vote_577' and 'vote_584' and 'vote_586' and 'vote_604' and
## 'vote_612' and 'vote_628' constant. Cannot scale data.

yhat_svm = predict(svm_model, test_data_p)
Y_vote_svm = c()
Y_vote_svm[which(yhat_svm >= 0.5)] = 1
Y_vote_svm[which(yhat_svm < 0.5)] = 0

size = length(yhat_svm)

matches_svm = data.frame(Match=integer(size), TruePositive=integer(size), Tru
eNegative=integer(size), FalsePositive=integer(size), FalseNegative=integer(s
ize))


for (idx in 1:size){

  if (Y_vote_svm[idx] == test_data$outcome[idx]){
    matches_svm[idx, "Match"] = 1

    if ((test_data$outcome[idx] == 1) && (Y_vote_svm[idx] == 1)){
      matches_svm[idx, "TruePositive"] = 1
    }
    if ((test_data$outcome[idx] == 0) && (Y_vote_svm[idx] == 0)){
      matches_svm[idx, "TrueNegative"] = 1
    }
  }
  else{
    matches_svm[idx, "Match"] = 0

    if ((test_data$outcome[idx] == 0) && (Y_vote_svm[idx] == 1)){
      matches_svm[idx, "FalsePositive"] = 1

    }
    if ((test_data$outcome[idx] == 1) && (Y_vote_svm[idx] == 0)){
      matches_svm[idx, "FalseNegative"] = 1

    }
  }
}
```

1.6

```
num_correct_pred_svm = sum(matches_svm$Match)
precision = sum(matches_svm$TruePositive)/(sum(matches_svm$TruePositive) + su
m(matches_svm$FalsePositive))
recall = sum(matches_svm$TruePositive)/(sum(matches_svm$TruePositive) + sum(m
atches_svm$FalseNegative))
F1 = 2/((1/precision)+(1/recall))
```

```
print(paste("Precision: ", toString(precision)))
```

```
## [1] "Precision:  0.976190476190476"
```

```
print(paste("Recall: ", toString(recall)))
```

```
## [1] "Recall:  1"
```

```
print(paste("F1: ", toString(F1)))
```

```
## [1] "F1:  0.987951807228916"
```

The SVM model was able to detect the one False Positive value that the OLS model detected. Based on the statistics, SVM outperformed the OLS model with a higher F1 value.

1.7

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rF_model = randomForest(predictors, training_data$outcome, kernel ="radial")
```

```
## Warning in randomForest.default(predictors, training_data$outcome, kernel
=
## "radial"): The response has five or fewer unique values. Are you sure you
want
## to do regression?
```

```
yhat_rF = predict(rF_model, test_data_p)
Y_vote_rF = c()
Y_vote_rF[which(yhat_rF >= 0.5)] = 1
Y_vote_rF[which(yhat_rF < 0.5)] = 0

size = length(yhat_rF)

matches_rF = data.frame(Match=integer(size), TruePositive=integer(size), True
Negative=integer(size), FalsePositive=integer(size), FalseNegative=integer(si
ze))


for (idx in 1:size){
```

```
  if (Y_vote_rF[idx] == test_data$outcome[idx]){
    matches_rF[idx, "Match"] = 1

    if ((test_data$outcome[idx] == 1) && (Y_vote_rF[idx] == 1)){
      matches_rF[idx, "TruePositive"] = 1
    }
    if ((test_data$outcome[idx] == 0) && (Y_vote_rF[idx] == 0)){
      matches_rF[idx, "TrueNegative"] = 1
    }
  }
  else{
    matches_rF[idx, "Match"] = 0

    if ((test_data$outcome[idx] == 0) && (Y_vote_rF[idx] == 1)){
      matches_rF[idx, "FalsePositive"] = 1

    }
    if ((test_data$outcome[idx] == 1) && (Y_vote_rF[idx] == 0)){
      matches_rF[idx, "FalseNegative"] = 1

    }
  }
}

num_correct_pred_rF = sum(matches_rF$Match)
precision = sum(matches_rF$TruePositive)/(sum(matches_rF$TruePositive) + sum(
matches_rF$FalsePositive))
recall = sum(matches_rF$TruePositive)/(sum(matches_rF$TruePositive) + sum(mat
ches_rF$FalseNegative))
F1 = 2/((1/precision)+(1/recall))

print(paste("Precision: ", toString(precision)))

## [1] "Precision:  0.953488372093023"

print(paste("Recall: ", toString(recall)))

## [1] "Recall:  1"

print(paste("F1: ", toString(F1)))

## [1] "F1:  0.976190476190476"
```

In comparing F1 values, the random Forest model performs better than the OLS model but not quite as well as the SVM model.

QUESTION 2

2.1

```
prctrain = read.csv("pres_results_by_county_train.csv")

prctrain_county = prctrain[prctrain$Geographic.Subtype == "County",] #subsett
ing for just counties

prctrain_county$Trump.Share = prctrain_county$Donald.J..Trump/prctrain_county
$Total.Vote #this is is trump share with  the NA values still included

prctrain_county = prctrain_county[!is.na(prctrain_county$Trump.Share),] #remo
ving rows where trump share contains NA value

prctrain_subset = prctrain_county[, c(9:59)] #what we are putting into regres
sion

summary(prctrain_county$Trump.Share)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.04087 0.55526 0.66889 0.63903 0.74958 0.94585
```

Yes, this does contradict the fact that Clinton won the popular vote. From what this data says, Trump's average vote share was 63%, that being a majority of votes. I find this interesting and I'm not quite sure what the explanation for this could be. I think this could point to how bias can arise in data science and how people can use their own personal analysis to support narratives that aren't true.

2.2

```
mod_lasso = glmnet(y = prctrain_county$Trump.Share, x = data.matrix(prctrain_
subset), alpha = 1)
```

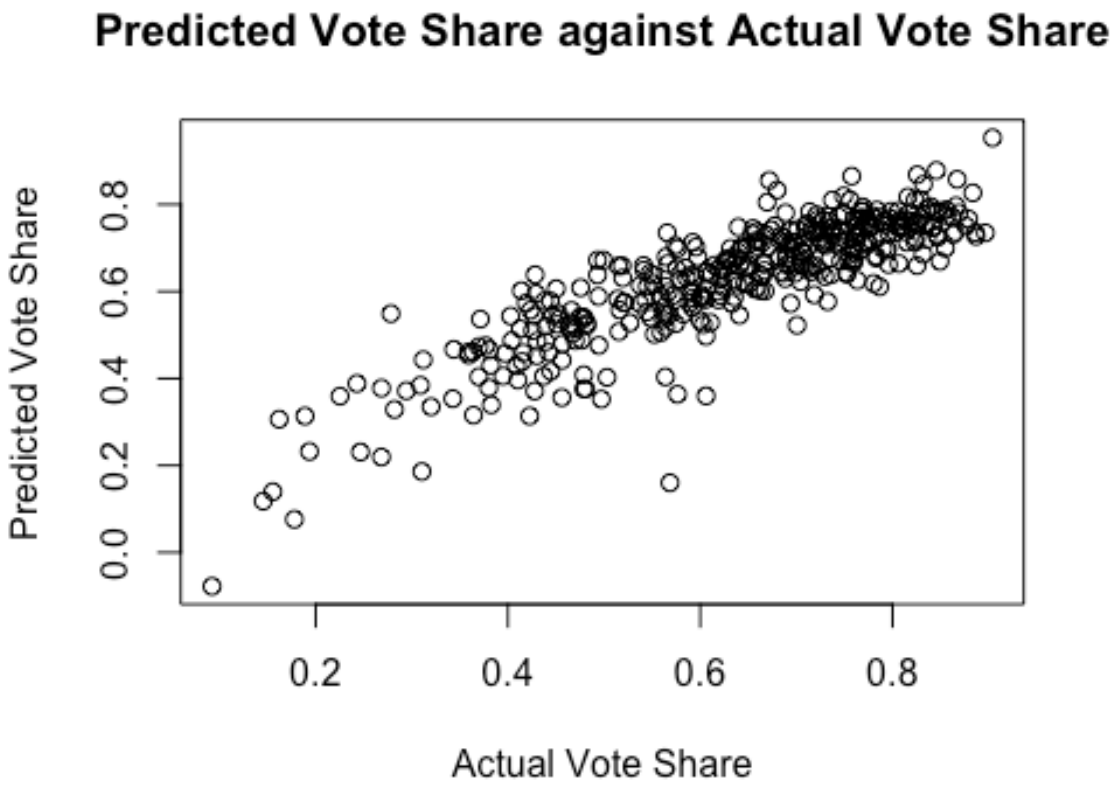
2.3

```
mse_min = Inf
choice_lamba = NaN

#iterate through lambda values and find minimum mse value
for (lambda in mod_lasso$lambda){

    yhat.lasso = predict(mod_lasso, newx=as.matrix(prctrain_subset), s=lambda
) #calculate predicted values

    mse = mean((prctrain_county$Trump.Share-yhat.lasso)^2)

    if (mse < mse_min){ #check to see if current mse is smaller than current
min
```

```
      mse_min = mse
      choice_lamba = lambda

    }
}

print(paste(toString(choice_lamba), " is the best lambda value with an in-sam
ple MSE value of ", toString(mse_min), "."))

## [1] "8.7849245941921e-06  is the best lambda value with an in-sample MSE v
alue of  0.00610317920178813 ."
```

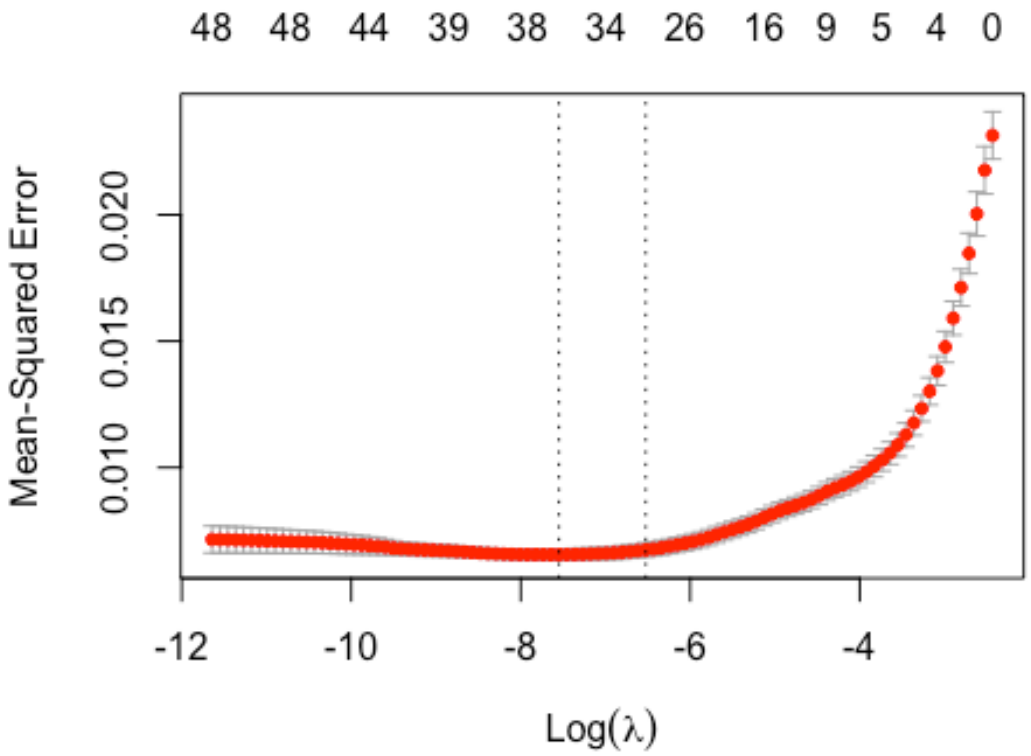
2.4

```
prctest = read.csv("pres_results_by_county_test.csv")

prctest_county = prctest[prctest$Geographic.Subtype == "County",]

prctest_county$Trump.Share = prctest_county$Donald.J..Trump/prctest_county$To
tal.Vote #this is is trump share with  the NA values still included

prctest_county = prctest_county[!is.na(prctest_county$Trump.Share),]

prctest_subset = prctest_county[, c(9:59)] #what we are putting into regressi
on

#performing prediction based on parameters chosen before
yhat.lasso = predict(mod_lasso, newx=as.matrix(prctest_subset), s=choice_lamb
a)

mse = mean((prctest_county$Trump.Share-yhat.lasso)^2)

print(paste("The MSE for the data is", toString(mse), "."))

## [1] "The MSE for the data is 0.00678634250799078 ."

plot(prctest_county$Trump.Share, yhat.lasso,
     xlab = "Actual Vote Share",
     ylab = "Predicted Vote Share")
title("Predicted Vote Share against Actual Vote Share") #apparently this shou
ld have a length of 413 according to Piazza?
```



2.5

```
set.seed(02139)

#implementing cross validated lasso regression
mod_lasso_cross = cv.glmnet(y = prctrain_county$Trump.Share, x = data.matrix(
prctrain_subset), alpha = 1)

plot(mod_lasso_cross)
```




```
#extracting chosen lambda value and its MSE
selected_lambda = mod_lasso_cross$lambda.min
selected_MSE = mod_lasso_cross$cvm[which(mod_lasso_cross$lambda==selected_lam
bda)]

print(paste("The selected lambda value is", toString(selected_lambda), "with
an MSE of", toString(selected_MSE), "."))
```

```
## [1] "The selected lambda value is 0.00052664239143802 with an MSE of 0.006
55438268655898 ."
```

2.6

```
#cross validated model used on test data

yhat.lasso.cross = predict(mod_lasso_cross, newx=as.matrix(prctest_subset), s
=selected_lambda)
mse.cross = mean((prctest_county$Trump.Share-yhat.lasso.cross)^2)

print(paste("The MSE for the data is", toString(mse.cross), "."))
```

```
## [1] "The MSE for the data is 0.00678106918208572 ."
```

```
print(paste("The MSE from the previous model (", toString(mse),") is greater
than the MSE from the cross-validated model(", toString(mse.cross), "), there
fore the cross-validated model performed better."), sep="")
```

```
## [1] "The MSE from the previous model ( 0.00678634250799078 ) is greater th
an the MSE from the cross-validated model( 0.00678106918208572 ), therefore t
he cross-validated model performed better."
```

2.7

```
coeff = mod_lasso_cross$nzero[which(mod_lasso_cross$lambda==selected_lambda)]
print(coeff)
```

```
## s55
##  37
```

The lambda that was chosen has 37 non-zero coefficients.