{Data Analytics I: Summer 2020}

Assignment 2: An Assessment of Logistic Regression and Linear Discriminant Models to Predict Spam

Marshall Tuck

7/13/2020

Introduction

In this assignment we will explore a dataset called spambase.csv, and develop a model using logistic regression and linear discriminant analysis techniques to classify whether an email with given characteristics is spam or not. We will explore the use of an appropriate probability tolerance level to classify whether spam or not for the purposes of our model, and then further explore how the use of that tolerance level influences false positives and overall error rate in our model against train data data, validation data, and test data.

In our analysis, there are a few important parameters:
1. A false positive is highly undesirable in the context of our problem statement (classifying a non-spam email as spam, and therefore screening it from our recipients)
2. Testing will be conducted using a 20% test set (in other words, 80% of our data will be used to train our model)
3. Both Logistic Regression and Linear Discriminant Analysis will be used to create models
4. A 5-fold iterated cross validation technique will be developed to analyze rates of false positives, false negatives, and to create confusion matrices to explore accuracy.

Background on Dataset and Parameters of Investigation

Our dataset, as explained in the spambase documentation, is a collection of 58 continuous and categorical variables, including the response variable of 1 or 0 (1 if an email is spam, 0 if it is not). 4601 observations are given in the raw data set.

The given explanatory variables in the original set, used as the basis of our models, can be summarized as: * 48 word match percentages: the percentage of words in the observation email that match the column header 'WORD' + i.e. the first column is 'order' - a percentage value of .2 indicates the word 'order' makes up 20% of the words in the observation email

- 6 character match percentages: the percentage of characters in the observation email that match the column header 'CHAR'
  - i.e. a 'cf.semicol' value of .01 indicates that 1% of the characters in a given observation email are ';'
- 3 capital letter variables: the longest unbroken string of capital letters, the average length of capital letter sequences, and the total number of capital letters in the observation email
- The response variable, 1 or 0, indicating whether spam or not.

```
#Read in csv
setwd("~/Documents/Statistics/Data Analytics I - Summer/Homeworks/HW2")
spam<-read.csv("spambase.csv")
#Coerce as spam
spam$spam<- as.factor(spam$spam)
```

Set Aside Training Data and Testing Data

First we will set aside 80% of our data as training data, and the remaining 20% as testing data. We will use our training data to build models and validate those models, and then use the "unseen" test data to explore how our models react to new data.

```
set.seed(42)

trainingindex<-sample(1:nrow(spam), .8*nrow(spam))

spam_train<- spam[trainingindex,]
spam_test<- spam[-trainingindex,]
```

Logistic Regression

Our first method of model-building uses the logistic regression technique. Logistic Regression is a model creation technique in which coefficients are assigned to each of the included variables, similar to linear regression, such that $p(X) = \frac{e^{intercept+coefficient_1*observation_1+...+coefficient_n*observation_n}}{1+e^{intercept+coefficient_1*observation_1+...+coefficient_n*observation_n}}$.

Observation n is then classified into one of two outcome categories, to maximize p(X) found above.

Model Fit

We start by reducing our variables to include only variables in our model with a p-value less than .05. We find 29 variables meet the criteria of significant, and are to be included in our initial regression model.

We see that the reduced model now has 29 regressors, 28 of which are significant as indicated by a pvalue less than .05. Our now single insignificant variable is 'crl.avg' with a pvalue of .155. We will remove this variable and refit our model. All terms included are now significant and we proceed to our model evaluation steps.

```
#Fit reduced model
spamfit_reduced<- glm(spam~our+over+remove+internet+order+free+  business+you+your+n000+money+hp+hpl+ge
family="binomial", data=spam_train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#Return Summary
summary(spamfit_reduced)
```

```
##
## Call:
## glm(formula = spam ~ our + over + remove + internet + order +
```

```
##     free + business + you + your + n000 + money + hp + hpl +
##     george + data + n85 + technology + meeting + project + re +
##     edu + conference + cf.semicol + cf.exclaim + cf.dollar +
##     cf.pound + crl.longest + crl.total, family = "binomial",
##     data = spam_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0371  -0.2457   0.0000   0.1537   4.4110
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.762e+00  1.303e-01 -13.523  < 2e-16 ***
## our          6.907e-01  1.178e-01   5.862 4.57e-09 ***
## over         8.072e-01  2.611e-01   3.091 0.001995 **
## remove       2.048e+00  3.345e-01   6.123 9.18e-10 ***
## internet     6.147e-01  1.654e-01   3.716 0.000202 ***
## order        6.538e-01  3.132e-01   2.087 0.036885 *
## free         9.023e-01  1.366e-01   6.603 4.03e-11 ***
## business     9.399e-01  2.142e-01   4.388 1.14e-05 ***
## you          1.137e-01  3.652e-02   3.113 0.001852 **
## your         1.798e-01  4.781e-02   3.761 0.000169 ***
## n000         2.205e+00  4.795e-01   4.598 4.26e-06 ***
## money        4.377e-01  1.776e-01   2.465 0.013696 *
## hp          -1.884e+00  2.773e-01  -6.793 1.10e-11 ***
## hpl         -1.006e+00  4.179e-01  -2.408 0.016047 *
## george      -1.705e+01  2.395e+00  -7.118 1.09e-12 ***
## data        -9.473e-01  3.664e-01  -2.586 0.009713 **
## n85         -1.873e+00  8.178e-01  -2.290 0.021994 *
## technology   9.571e-01  3.386e-01   2.827 0.004700 **
## meeting     -3.248e+00  1.027e+00  -3.163 0.001559 **
## project     -2.701e+00  9.800e-01  -2.757 0.005841 **
## re          -7.580e-01  1.562e-01  -4.853 1.22e-06 ***
## edu         -2.120e+00  3.776e-01  -5.614 1.98e-08 ***
## conference  -4.815e+00  1.884e+00  -2.556 0.010588 *
## cf.semicol  -8.881e-01  2.896e-01  -3.067 0.002162 **
## cf.exclaim   3.724e-01  9.598e-02   3.880 0.000104 ***
## cf.dollar    5.225e+00  7.577e-01   6.896 5.34e-12 ***
## cf.pound     3.199e+00  5.719e-01   5.593 2.24e-08 ***
## crl.longest  1.506e-02  2.210e-03   6.815 9.43e-12 ***
## crl.total    6.793e-04  2.067e-04   3.287 0.001013 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4942.6  on 3679  degrees of freedom
## Residual deviance: 1574.6  on 3651  degrees of freedom
## AIC: 1632.6
##
## Number of Fisher Scoring iterations: 11
```

Logistic Regression Visualization

To further explore our data and prediction model, the first thing I will do is look to see how our actual spam values compare to the corresponding fitted values from our logistic regression model. We see that there are false positives and false negatives from our initial model - there are fitted values near 1 for a true spam values of 0 (indicating a false positive) and fitted values of 0 for true spam values of 1 (indicating a false negative).

However, we see that our model appears to be better than a 50/50 chance, as most data seems to be grouped according to its true spam value.

```
#Output ggplot showing jitter of fitted vs. observed
spam_train %>%
  mutate(fitted=spamfit_reduced$fitted.values)%>%
  ggplot(aes(x=spam, y=fitted, fill=spam, color=spam))+
           geom_jitter()+
          labs(x="True Spam", y="Fitted Probability", title="Fitted Probability vs. True Spam Identity
```

Logistic Regression Error Calculation

Our next step will be to set a threshold of .5 and then analyze how this threshold performs in terms of error rates. In other words, our model must predict an observation has a greater than 50% chance of being spam for us to consider it spam. I write a function below that analyzes false positive, true positive, false negative, and true negative values of our fitted values at .5 tolerance vs. our true spam values, and then outputs the overall rates into a table.

With a tolerance of .5, we see a false positive percentage of .05 (false positive amount = 118), and a false

negative percentage of .11 (false negative amount = 165). Our overall error rate is approximately .077, meaning that our model mis-classifiies 7.7% of our training data.

```
#Write function to output tp, tn, fp, fn, along with some additional alegbra
error_rate<-function(formula, data, tol){

  set<- data %>%
  mutate(fitted=formula$fitted.values)

  tp<- set %>%
  filter(fitted>tol&spam==1)%>%
  nrow()

  tn<- set %>%
    filter(fitted<tol&spam==0)%>%
    nrow()

  fp<- set %>%
    filter(fitted>tol&spam==0)%>%
    nrow()

  fn<- set %>%
    filter(fitted<tol&spam==1)%>%
    nrow()

cat("\n","True Prediction Rate",(tp+tn)/nrow(data),"\n",
    "Error Rate",(fp+fn)/nrow(data),"\n\n",
    "False Positive Rate", fp/(fp+tn),"\n",
    "False Negative Rate", fn/(fn+tp),"\n\n\n",
    "True Positives", tp, "\n",
    "True Negatives", tn, "\n",
    "False Positives", fp, "\n",
    "False Negatives", fn)
}

error_rate(formula=spamfit_reduced, data=spam_train, tol=.5)
```

```
##
##  True Prediction Rate 0.9230978
##  Error Rate 0.07690217
##
##  False Positive Rate 0.05312922
##  False Negative Rate 0.1130912
##
##
##  True Positives 1294
##  True Negatives 2103
##  False Positives 118
##  False Negatives 165
```

A confusion matrix will then be output to understand how our fitted values compare to our true values using the tolerance level of .5.

In the table below, we see a confusion matrix showing the positive and negative values in a different format.

Across the top row, we see the true spam observed value (0= not spam, 1=spam) from our original data set. Downwards, we see the predicted spam values from our model, using a tolerance of .5.

```
#Confusion matrix
spam_observed<-spam_train$spam
spam_pred<-rep("No", nrow(spam_train))
spam_pred[spamfit_reduced$fitted.values > .5]<- "Yes"

table(spam_pred, spam_observed)
```

```
##          spam_observed
## spam_pred    0    1
##       No  2103  165
##       Yes  118 1294
```

Logistic Regression: Adjusting Tolerance to Produce Zero False Negatives

Below, we will investigate to find the optimum tolerance level that will minimize the risk of false positives. In the while loop below, we will iterate through our positive and negative error rate while our false positive count is greater than zero and while our tolerance is .99 or below.

We find a tolerance of .99 will virtually eliminate our false positive rate to .005 (false positive amount = 12). Note: A tolerance of 1 does not make sense, in the way we have required our fitted values to be greater than our tolerance for our observation to be categorized as a 'likely positive'.

However, this corresponds to an increase in our false negative rate to .65 (false negative amount = 946). This is because we have raised our "bar" so high to only categorize fitted values greater than .99 as positives, that we have categorized strong fitted values (likely between .6-.99) as not spam.

Further, we see that increasing our tol to .99 increases our overall error rate to .26.

```
#Write new function that includes a while loop to iterate until
#our false positive rate is minimized and our tolerance is less than or equal to .99
error_rate_iter<-function(formula, data, tol){

  set<- data %>%
  mutate(fitted=formula$fitted.values)

  fp<- set %>%
  filter(fitted>tol&spam==0)%>%
  nrow()

  while (fp>0 & tol<.99){

  tol<-tol+.01

  tp<- set %>%
    filter(fitted>tol&spam==1)%>%
    nrow()

  tn<- set %>%
    filter(fitted<=tol&spam==0)%>%
    nrow()
```

```
  fp<- set %>%
    filter(fitted>tol&spam==0)%>%
    nrow()

  fn<- set %>%
    filter(fitted<=tol&spam==1)%>%
    nrow()

  }


cat("Zero False Positive Tolerance", tol, "\n",
    "\n","True Prediction Rate",(tp+tn)/nrow(data),"\n",
    "Error Rate",(fp+fn)/nrow(data),"\n\n",
    "False Positive Rate", fp/(fp+tn),"\n",
    "False Negative Rate", fn/(fn+tp),"\n\n\n",
    "True Positives", tp, "\n",
    "True Negatives", tn, "\n",
    "False Positives", fp, "\n",
    "False Negatives", fn)
  }



error_rate_iter(formula=spamfit_reduced, data=spam_train, tol=.5)
```

```
## Zero False Positive Tolerance 0.99
##
##  True Prediction Rate 0.7396739
##  Error Rate 0.2603261
##
##  False Positive Rate 0.005402972
##  False Negative Rate 0.6483893
##
##
##  True Positives 513
##  True Negatives 2209
##  False Positives 12
##  False Negatives 946
```

Logistic Regression Confusion Matrix @ tolerance = .99

We produce a confusion matrix with our tolerance of .99. Our upper left and lower right values are considered our "accurate predictions". Upper right and lower left are considered our "errors".

```
#Output confusion matrix with tolerance of .99
#Set observed values and predict values
#where predict value >.99 replace with Yes, and output table.

spam_observed<-spam_train$spam
spam_pred<-rep("No", nrow(spam_train))
spam_pred[spamfit_reduced$fitted.values >.99]<- "Yes"
```

```r
table(spam_pred, spam_observed)
```

```
##          spam_observed
## spam_pred    0    1
##       No  2209  946
##       Yes   12  513
```

However, a tolerance of .99 means that we simply let all email through, because we want to minimize the amount of non-spam email that gets filtered out (false positives). This is a very simple classification technique, and with it comes an increase in overall error rate from 7.7% to 26%.

Below, we adjust our tolerance to meet the reasonable threshold that no more than 1% of our email observations are to be false positives. From a training data set of 3680 emails, 1% of that is 36.8. In other words, we will toggle below to find a tolerance level that produces no more than 36 false positives in our training data set.

Logistic Fit using Reasonable Tolerance Level @ tolerance = .88

In our confusion matrix below, we find a tolerance of .88 produces 33 false positives, and produces and error rate of (33+522)*100/3680= 15.1%. This will be considered our logistic regression tolerance level as we proceed to validating against our test data.

```r
#Output confusion matrix with tolerance of .88
#Set observed values and predict values
#where predict value >.88 replace with Yes, and output table.

spam_observed<-spam_train$spam
spam_pred<-rep("No", nrow(spam_train))
spam_pred[spamfit_reduced$fitted.values >.88]<- "Yes"

table(spam_pred, spam_observed)
```

```
##          spam_observed
## spam_pred    0    1
##       No  2188  522
##       Yes   33  937
```

Linear Discriminant Analysis

Our next approach for model building the spam data set will be to use the linear discriminant analysis method. First, we fit an LDA model using only the regressors used in the logistic regression model above on our training set. Linear discriminant analysis technique creates discriminant functions for each class of data, and then assigns the each observation to a predicted class based on our discriminant functions.

Model Fit and Interpretation

The first thing we analyze is the calculated prior probabilities of the groups (true spam= 0,1). We see prior probabilities of .604 for spam=0, and .396 for spam=1.

Next we look at the mean values, which show the average amount of each observation value from each regressor in classes of spam= 0,1. For example, we see that 'our' appears nearly three times as much in spam ('our' value = .51) than nonspam ('our' value = .18). Capital letters exist in spam at a rate of almost three times as much than in non-spam (464.4 vs. 163.78)

Finally, we see our individual discriminant functions produced. These coefficients are regarded as scoring vectors, such that for observation 2609 (the first observation in our training set), x=.368 $*'$ *our*$'$ + .466 $*'$ *over*$'$ + ... + .00044 $*$ *crl.total*. This calculated value, along with the prior probability mentioned above, is used to compute the posterior probability of each observation (see formula on slide 12 within the first citation below). Classes are then assigned based on the comparison of the posterior probability to our chosen tolerance level.

Citation:
* https://web.stanford.edu/class/stats202/content/lec9.pdf
* https://stats.stackexchange.com/questions/87479/what-are-coefficients-of-linear-discriminants-in-lda
* https://stackoverflow.com/questions/40087417/lda-interpretation

```
#Fit LDA model
spamfit_lda<- lda(spam~our+over+remove+internet+order+free+  business+you+your+n000+money+hp+hpl+george+

#Return values from LDA fit
print("Prior Probabilities")
```

```
## [1] "Prior Probabilities"
```

```
head(spamfit_lda$prior)
```

```
##         0         1
## 0.6035326 0.3964674
```

```
print("Means")
```

```
## [1] "Means"
```

```
head(spamfit_lda$means)
```

```
##         our       over     remove   internet      order       free   business
## 0 0.1793967 0.04840162 0.01112112 0.03958577 0.03487618 0.07023863 0.04859523
## 1 0.5057711 0.18095956 0.26568197 0.21503084 0.17198081 0.54302262 0.28175463
##        you     your        n000      money         hp        hpl       george
## 0 1.283260 0.438118 0.006825754 0.01866276 0.91652859 0.44134174 1.2703241783
## 1 2.262392 1.365113 0.248540096 0.21701165 0.02029472 0.01030843 0.0009252913
##         data         n85 technology     meeting     project        re       edu
## 0 0.15809995 0.168153985 0.14203512 0.197393066 0.122309770 0.4033724 0.2607429
## 1 0.01376285 0.007806717 0.03057574 0.002234407 0.004235778 0.1289856 0.0127416
##    conference cf.semicol cf.exclaim  cf.dollar   cf.pound crl.longest crl.total
## 0 0.045997299 0.05184827  0.1151751 0.01243764 0.02017875    18.43044  163.7753
## 1 0.002549692 0.02060658  0.5090610 0.17678821 0.07699315   107.14325  464.4071
```

```
print("Scaling Values")
```

```
## [1] "Scaling Values"
```

```r
head(spamfit_lda$scaling)
```

```
##                LD1
## our      0.3682361
## over     0.4655273
## remove   0.9631513
## internet 0.4324144
## order    0.3906307
## free     0.3430133
```

```r
#Return predicted values
lda_pred<-predict(spamfit_lda)

#Get posteriors, return top 5
lda_pred_post<- lda_pred$posterior
print("First 5 Posterior Probabilities")
```

```
## [1] "First 5 Posterior Probabilities"
```

```r
head(lda_pred_post,5)
```

```
##              0           1
## 2609 0.9256274 0.074372601
## 4069 0.9903322 0.009667796
## 2369 0.8540707 0.145929278
## 1098 0.2922694 0.707730554
## 1252 0.1648563 0.835143676
```

Again, we find a tolerance of .99 is required to ensure our false positive count is as close to zero as possible (as was also seen in our logistic regression model).

At a tolerance of .99 (meaning that our posterior probability of spam = 1 is required to be greater than .99 in order to be classified as spam in our model), we find a false positive count of 7, which corresponds to a false positive rate of .0032.

Our error rate at this .99 tolerance level is .349, or a mis-classification rate of 35%.

```r
#Bind columns by row number between original spam_train data and posterior probabilities
spam_train_post<-spam_train%>%
  bind_cols(post0=lda_pred_post[,1], post1=lda_pred_post[,2])

#Calculate Metrics on Bound Data

#False Positives
fp<- spam_train_post%>%
  filter(post1>.99 & spam==0)%>%
  nrow()

#False Negatives
fn<- spam_train_post%>%
  filter(post1<=.99 & spam==1)%>%
  nrow()
```

```
#True Positives
tp<- spam_train_post%>%
  filter(post1>.99 & spam==1)%>%
  nrow()

#True Negatives
tn<- spam_train_post%>%
  filter(post1<=.99 & spam==0)%>%
  nrow()

cat("\n","True Prediction Rate",(tp+tn)/nrow(spam_train),"\n",
    "Error Rate",(fp+fn)/nrow(spam_train),"\n\n",
    "False Positive Rate", fp/(fp+tn),"\n",
    "False Negative Rate", fn/(fn+tp),"\n\n\n",
    "True Positives", tp, "\n",
    "True Negatives", tn, "\n",
    "False Positives", fp, "\n",
    "False Negatives", fn)
```

```
##
##  True Prediction Rate 0.6529891
##  Error Rate 0.3470109
##
##  False Positive Rate 0.003151733
##  False Negative Rate 0.8704592
##
##
##  True Positives 189
##  True Negatives 2214
##  False Positives 7
##  False Negatives 1270
```

LDA Confusion Matrix @ tolerance = .99

Next we output a confusion matrix showing the false positives and false negatives for a tolerance level of .99 in our LDA model. We see the same false positive count.

In summary, even a tolerance level of .99 in our model contains greater than 0 false positives.

```
#LDA Confusion Matrix
#Set observed values and predict values
#where predict value >.99 replace with Yes, and output table.
spam_observed_lda<-spam_train$spam
spam_pred_lda<-rep("No", nrow(spam_train))
spam_pred_lda[spam_train_post$post1 > .99]<- "Yes"

table(spam_pred_lda, spam_observed_lda)
```

```
##              spam_observed_lda
## spam_pred_lda    0    1
##           No  2214 1270
##           Yes    7  189
```

Our next step will be to find a "reasonable" tolerance level, such that the number of false positives again is less than 1% of our total observation count in our training data set. This corresponds to a number of false positives no greater than 36.

LDA Confusion Matrix @ tolerance = .84

We find that a tolerance of .84 returns a false positive count from our training data set of 34, which is less than our threshold of 36. We will proceed with the use of .84 as our tolerance level for our linear discriminant model, as we look to assessing our model fit against unseen test data.

Our overall error rate at this tolerance level is (735+34)*100/3680 = 20.8%

```
#LDA Confusion Matrix
#Set observed values and predict values
#where predict value >.84 replace with Yes, and output table.
spam_observed_lda<-spam_train$spam
spam_pred_lda<-rep("No", nrow(spam_train))
spam_pred_lda[spam_train_post$post1 > .84]<- "Yes"

table(spam_pred_lda, spam_observed_lda)
```

```
##              spam_observed_lda
## spam_pred_lda    0    1
##           No  2187  735
##           Yes   34  724
```

Initial Interpretation and Recommendation

We now have two models - a model from logistic regression functional form using a tolerance of .88 and a model using the linear discriminant analysis functional form with a tolerance of .84. At this point, the preferred model is the logistic fit model, which produces 522 false negatives compared to 735 false negatives produced by the LDA model.

Due to keeping our false positive count below 36, the lower false negative count of our logistic fit model vs. our LDA model corresponds to a lower overall error count (15.1% error rate compared to 20.8% error rate with our LDA model).

Cross Validation

Next we will proceed to measure the variability of our model accuracy against our training data set, again utilizing an 80/20 split of training data vs. holdout data. Our method for performing this analysis will be iterated 5 fold cross validation - or repeatedly sampling a 80/20 split from our training data, using 80% of our data to train a model using our predefined tolerance levels (.88 for logistic fit and .84 for LDA), and validating our predicted fits against our training data and our testing data.

Logistic Regression Iterated 5 fold Cross Validation

First we will perform a 5-fold iterated method 100 times to find 500 total error rates from our logistic regression model on both our training data and our validation error. Below I show the first five results from our 5 fold cross validation for our logistic regression model at a tolerance of .88.

```r
logistic.k.fold.validator <- function(df, K, iter) {

  # this function calculates the errors of a single fold using the fold as the holdout data
  fold.errors <- function(df, holdout.indices) {
    train.data <- df[-holdout.indices, ]
    holdout.data <- df[holdout.indices, ]
    fit<- glm(spam~our+over+remove+internet+order+free+
                          business+you+your+n000+money+hp+hpl+george+data+n85+technology+
                          meeting+project+re+edu+conference+cf.semicol+cf.exclaim+cf.dollar+cf.pound+cr
                        family="binomial", data=train.data)

    train.predict <- predict(fit)
    #Revert fitted log values back to probabilities
    train.predict.p<- exp(train.predict)/(1+exp(train.predict))
    logistic.train.error <- train.data%>%
      mutate(predict=train.predict.p)%>%
      summarize(logistic.train.error=mean((spam==0 & predict >.88) | (spam==1 &predict<=.88)))


    holdout.predict <- predict(fit, newdata = holdout.data)
    #Revert fitted log values back to probabilities
    holdout.predict.p <- exp(holdout.predict)/(1+exp(holdout.predict))
    logistic.valid.error <- holdout.data%>%
      mutate(predict=holdout.predict.p)%>%
      summarize(logistic.valid.error=mean((spam==0 & predict >.88) | (spam==1 & predict<=.88)))

    tibble(logistic.train.error, logistic.valid.error)
  }
  errors <- tibble()
#Add an outer for loop which iterates iter times
  for (j in 1:iter) {
  # shuffle the data and create the folds
  indices <- sample(1:nrow(df))

  folds <- cut(indices, breaks = K, labels = F)
  # set error to 0 to begin accumulation of fold error rates

  # iterate on the number of folds
  for (i in 1:K) {
    holdout.indices <- which(folds == i, arr.ind = T)
    folded.errors <- fold.errors(df, holdout.indices)
    errors <- errors %>%
      bind_rows(folded.errors)
  }
  }
errors
}

suppressWarnings(logistic.k.fold.validator(spam_train,5,1))
```

```
## # A tibble: 5 x 2
##   logistic.train.error logistic.valid.error
##                  <dbl>                <dbl>
```

```
## 1               0.151               0.133
## 2               0.146               0.170
## 3               0.139               0.163
## 4               0.155               0.128
## 5               0.149               0.166
```

LDA Iterated 5 fold Cross Validation

We will perform a similar cross validation method on our LDA function, returning the individual error rates from our of our k*p=500 iterations. Below I show a sample of our first 5 values from our first iteration of 5 fold cross validation at a tolerance of .88.

```
lda.k.fold.validator <- function(df, K, iter) {

  # this function calculates the errors of a single fold using the fold as the holdout data
  fold.errors <- function(df, holdout.indices) {
    train.data <- df[-holdout.indices, ]
    holdout.data <- df[holdout.indices, ]
    fit <- lda(spam~our+over+remove+internet+order+free+
                    business+you+your+n000+money+hp+hpl+george+data+n85+technology+
                    meeting+project+re+edu+conference+cf.semicol+cf.exclaim+cf.dollar+cf.pound+crl.lon
                    data=train.data)

    #Training Data Error Rate
    train.predict <- predict(fit, train.data)

    train.predict.post<-train.predict$posterior

    train.error.count <- train.data%>%
      bind_cols(post0=train.predict.post[,1], post1=train.predict.post[,2])%>%
      filter(((spam==0 & post1>.88)|(spam==1 & post1<=.88)))%>%
      nrow()

    train.error<- train.error.count/nrow(train.data)

    #Holdout Data Error Rate
    holdout.predict <- predict(fit, newdata = holdout.data)
    holdout.predict.post<-holdout.predict$posterior

    holdout.error.count<-holdout.data%>%
      bind_cols(post0=holdout.predict.post[,1], post1=holdout.predict.post[,2])%>%
      filter((spam==0 & post1>.88)|(spam==1 & post1<=.88))%>%
      nrow()

    holdout.error<-holdout.error.count/nrow(holdout.data)

    tibble(lda.train.error = train.error, lda.valid.error = holdout.error)
  }
errors<-tibble()
#Add an outer for loop which iterates iter times
  for (j in 1:iter) {
  # shuffle the data and create the folds
  indices <- sample(1:nrow(df))
```

```
  folds <- cut(indices, breaks = K, labels = F)
  # set error to 0 to begin accumulation of fold error rates

  # iterate on the number of folds

  for (i in 1:K) {
    holdout.indices <- which(folds == i, arr.ind = T)
    folded.errors <- fold.errors(df, holdout.indices)
    errors <- errors %>%
      bind_rows(folded.errors)
  }
  }
errors
}

lda.k.fold.validator(spam_train, 5, 1)
```

```
## # A tibble: 5 x 2
##   lda.train.error lda.valid.error
##             <dbl>           <dbl>
## 1           0.226           0.231
## 2           0.218           0.209
## 3           0.223           0.230
## 4           0.230           0.227
## 5           0.228           0.221
```

Bootstrap Similarities

In essence, this iterated cross validation technique is a bootstrap method, with the added benefit that repeated subsampling of our training data into training data and validation data allows us to measure our error rate as a random variable over our repeated subsamples. We are then able to see the variance of our model over our repeated measures.

Boxplot of Error Rates

Below is a boxplot representation of the spread of errors, at a tolerance of .88 for logistic regression and a tolerance of .84 for Linear Discriminant Analysis. We see lower error rates for both our training data and our validation data for our logistic fit model, which indicates that our preferred model (as defined by minimizing overall error rate) is our logistic fit model.

Further, we notice larger variability for our validation holdout set as compared to our training set, which is expected as our model would be expected to perform worse against data that was not used to build the model.

It appears that the variability for each measure (training and validation error rate) is approximately equal between our two models.

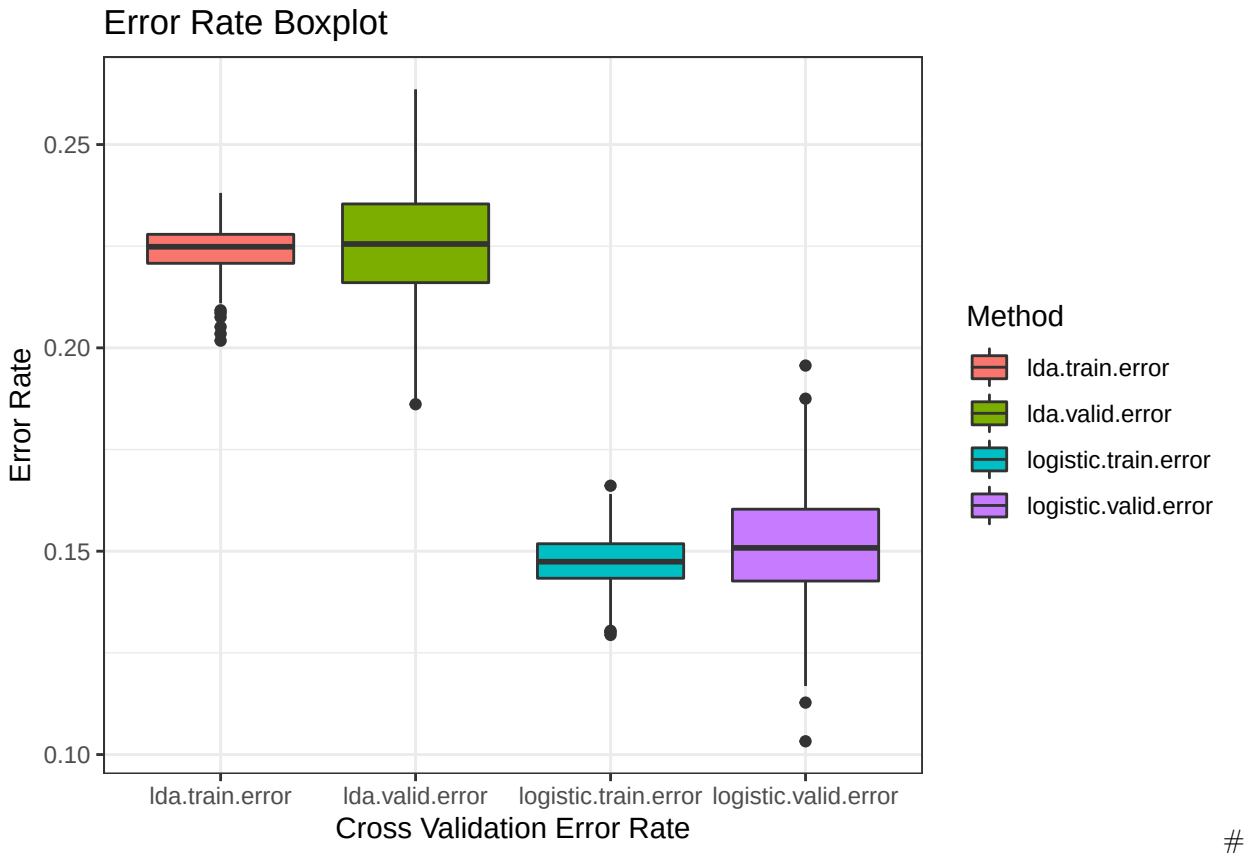
```
#Bind output of 100 5 fold cross validation outputs using both model

error_df<-suppressWarnings(cbind(logistic.k.fold.validator(spam_train,5,100),lda.k.fold.validator(spam_

#GGplot showing variance of error rates on train data vs. holdout data.
```

```
error_df%>%
  gather(Method, Value)%>%
  ggplot()+
  geom_boxplot(aes(x=Method, y=Value, fill=Method))+
  labs(x="Cross Validation Error Rate", y="Error Rate", title="Error Rate Boxplot")+
  theme_bw()
```

```
## Warning: Removed 2 rows containing non-finite values (stat_boxplot).
```



#

Introduction of Test Data

Next we introduce our test data (held out in the first steps in the assignment), and compare the fits of our models to the values of our test data.

Logistic Regression Confusion Matrix on Test Data

The below confusion matrix is of our logistic fit model on our test data at a tolerance of .88. We find an overall error rate for our logistic fit model against our test data set as (5+127)*100/931=14.3%

```
#Revert predicted values from log odds to odds
log_predict<-predict(spamfit_reduced, newdata=spam_test)
log_predict<-(exp(log_predict))/(1+exp(log_predict))

#Output confusion matrix we p > .88
spam_observed_log<-spam_test$spam
spam_pred_log<-rep("No", nrow(spam_test))
```

```
spam_pred_log[log_predict >.88]<- "Yes"

table(spam_pred_log, spam_observed_log)
```

```
##              spam_observed_log
## spam_pred_log   0   1
##           No  562 127
##           Yes   5 227
```

LDA Confusion Matrix on Test Data

Below is a confusion matrix of our LDA model on our test data at a tolerance of .84. We find an overall error rate of (7+172)*100/921= 19.4%

```
#Fit lda model
lda_fit <- lda(spam~our+over+remove+internet+order+free+
                    business+you+your+n000+money+hp+hpl+george+data+n85+technology+
                    meeting+project+re+edu+conference+cf.semicol+cf.exclaim+cf.dollar+cf.pound+crl.lon
                    data=spam_train)

#Predict posterior values vs. test data
lda_pred_test<-predict(lda_fit, newdata=spam_test)
lda_pred_test_post<- lda_pred_test$posterior

spam_test_post<-spam_test%>%
  bind_cols(post0=lda_pred_test_post[,1], post1=lda_pred_test_post[,2])

#Output where posterior1 value > .84
spam_observed_lda_test<-spam_test$spam
spam_pred_lda_test<-rep("No", nrow(spam_test))
spam_pred_lda_test[spam_test_post$post1 > .84]<- "Yes"

table(spam_pred_lda_test, spam_observed_lda_test)
```

```
##                   spam_observed_lda_test
## spam_pred_lda_test   0   1
##                No  560 172
##                Yes   7 182
```

Recommendation and Future Steps

Our goal in this assignment is to choose between a logistic regression model and a linear discriminant model for the goal of minimizing false positive spam result, while also remaining stable.

Initially, we found that even with a tolerance level of .99 (meaning to be assessed as spam, we require a 100% probability of being spam as assessed by our model), we have some false positives. This may be a weakness in our model and requires the inclusion of other variables in our data set to prevent false positives even at a tolerance level of .99. I recommend future data gathering to assess whether a model can be built that allows for no false positives. *Using a tolerance of .99, I suggest use of the LDA model to minimize false positive count, as it produces 7 false positives vs. 12 false positives for our logistic fit model using our training data set.*

Next, we use respective "reasonable" tolerance levels of .88 for our logistic regression model and .84 for our linear discriminant analysis model to find a false positive count of no greater than 36. *To minimize overall error rate at our reasonable tolerance levels, it is preferred to use our logistic regression model. We found associated error rates of 15.1% for logistic fit and 20.8% for LDA, respectively, using our training data set.*

Next, we used iterated 5 fold cross validation to visualize the stability of our model over training data and validation data, not having yet exposed our model to our testing data. We find similar stability of both of our models as we proceed from training data sets to holdout data sets in our iterated CV technique. *An iterated 5 fold validation model shows that each of our model is approximately equally stable against our validation data, however the logistic fit model has a lower error rate as compared to our LDA model (approximately 15% vs. 22.5%) as assessed against our train data, split into training and validation subsets.*

Finally, we expose our model to our test data set, and assess the goodness of fit of both models against unseen data. *We find our logistic fit model has a lower error rate and lower false positive count as compared to our linear discriminant analysis model using our reasonable tolerance level, assessed on our test data set.*

Final Thoughts:
* To minimize error rates, my recommendation is to use our logistic fit model, even though our logistic fit model produces more false positives at a given tolerance level.
* To minimize false positives, my recommendation is to use a linear discriminant analysis model, though it produces higher error rates at a given tolerance level.
* At our "reasonable" tolerance levels, our logistic regression model produces both a lower error rate and a lower false positive count as compared to our linear discriminant analysis model, and therefore is considered our preferred final model.