STAT 5525: Homework 2

Adam Wells

Monday, July 22

Part I: Logistic Regression

1. Set aside a 20% sample to be a test dataset

```
spambase <- read_csv("~/Desktop/STAT5525/HW2/spambase.csv")
spambase$spam<-as.factor(spambase$spam)

#The following code uses the rsample package to split the data.
set.seed(1)
train_test_split<-initial_split(spambase,prop=0.80)
train_data<-train_test_split%>%training()
test_data<-train_test_split%>%testing()
```

2. Fit a logistic model to all of the training data and display the summary.

```
logistic_model<-logistic_reg("classification")%>%
  set_engine("glm")%>%
  fit(spam~.,data=train_data)

pander(tidy(logistic_model))
```

term	estimate	std.error	statistic	p.value
(Intercept)	-1.628	0.16	-10.18	2.48e-24
make	-0.4221	0.2527	-1.67	0.09487
address	-0.1241	0.07228	-1.717	0.08601
all	0.187	0.1206	1.551	0.121
n3d	2.666	1.715	1.554	0.1201
our	0.6583	0.1146	5.745	9.177e-09
over	1.092	0.2943	3.712	0.0002059
remove	2.044	0.332	6.158	7.382e-10
internet	0.4934	0.1658	2.976	0.002917
order	0.5806	0.3057	1.899	0.05756
mail	0.09621	0.07424	1.296	0.195
receive	-0.1569	0.3205	-0.4895	0.6245
will	-0.1453	0.08309	-1.748	0.08041
people	0.01837	0.2621	0.07009	0.9441
report	0.06552	0.1492	0.4393	0.6605
addresses	1.187	0.7126	1.665	0.09583
free	1.109	0.163	6.806	1.006e-11

term	estimate	std.error	statistic	p.value
business	0.8226	0.2335	3.522	0.0004276
email	0.2005	0.1257	1.595	0.1107
you	0.07249	0.03892	1.863	0.06251
credit	0.9956	0.5924	1.68	0.09287
your	0.2162	0.05672	3.811	0.0001383
font	0.2836	0.2133	1.33	0.1836
n000	2.307	0.5198	4.438	9.088e-06
money	0.4324	0.1611	2.684	0.007283
hp	-2.035	0.3644	-5.584	2.346e-08
hpl	-0.8561	0.4479	-1.911	0.05594
george	-11.21	2.332	-4.807	1.53e-06
n650	0.5395	0.3012	1.791	0.07323
lab	-2.131	1.435	-1.485	0.1376
labs	-0.3869	0.3799	-1.018	0.3085
telnet	-0.1151	0.3689	-0.312	0.755
n857	-81.46	4403	-0.0185	0.9852
data	-1.072	0.3957	-2.708	0.006766
n415	1.262	1.78	0.7089	0.4784
n85	-2	0.8497	-2.354	0.01857
technology	0.783	0.3465	2.26	0.02383
n1999	0.1033	0.1912	0.54	0.5892
parts	1.729	0.9482	1.823	0.06825
pm	-0.6896	0.4487	-1.537	0.1243
direct	-0.3056	0.3733	-0.8186	0.413
cs	-46.89	25.62	-1.83	0.06718
meeting	-3.472	1.235	-2.811	0.004934
original	-0.706	0.7227	-0.9769	0.3286
project	-1.814	0.637	-2.847	0.004411
re	-0.699	0.1553	-4.503	6.716e-06
edu	-1.388	0.2872	-4.833	1.345e-06
table	-1.936	1.484	-1.305	0.1919
conference	-3.9	1.675	-2.328	0.01992
cf.semicol	-1.666	0.5989	-2.783	0.00539
cf.lparen	-0.177	0.3094	-0.5719	0.5674
cf.lbrack	-0.5813	1.036	-0.561	0.5748
cf.exclaim	0.2434	0.06868	3.544	0.0003938
cf.dollar	4.455	0.7421	6.003	1.938e-09
cf.pound	2.955	1.173	2.52	0.01174
crl.avg	0.01183	0.02017	0.5865	0.5575
crl.longest	0.007875	0.002762	2.851	0.004357
crl.total	0.001206	0.0002502	4.822	1.423e-06

3. Evaluate the fit and simplify the model by eliminating some predictors. Fit the simplification. (You may use techniques from the Advanced Regression course.) Explain why you chose to eliminate certain predictors and the approach or any technique(s) you used to do so. (Hint: a simple approach based on significance is adequate for this homework.) Display the summary.

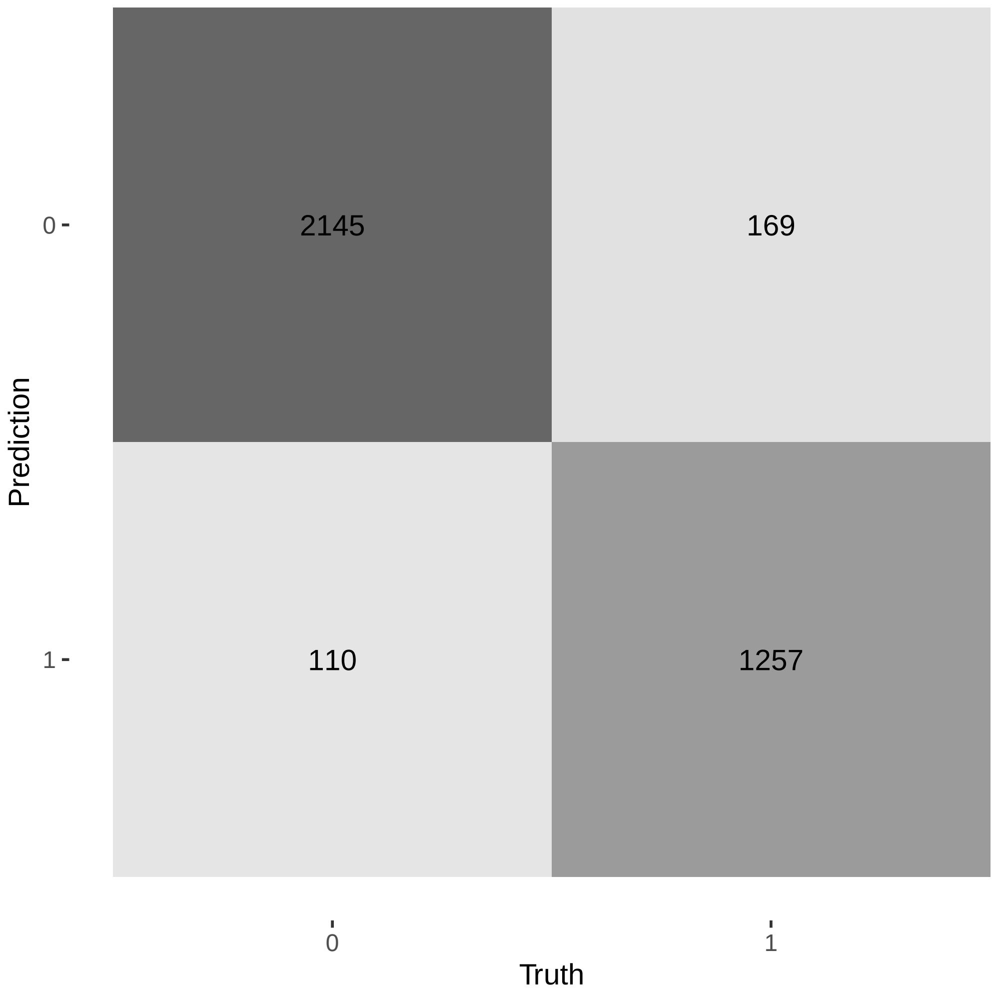
All predictors with a p-value greater than 0.05 have been eliminated from the following model.

```r
formula<-(spam~our+over+remove+internet+free+business+your+n000+money+hp+george+
        data+n85+technology+meeting+project+re+edu+conference+cf.semicol+cf.exclaim+
        cf.dollar+cf.pound+crl.longest+crl.total)

logistic_model_reduced<-logistic_reg("classification")%>%
  set_engine("glm")%>%
  fit(formula,data=train_data)

pander(tidy(logistic_model_reduced))
```

term	estimate	std.error	statistic	p.value
(Intercept)	-1.57	0.1126	-13.94	3.471e-44
our	0.7203	0.1156	6.229	4.704e-10
over	1.104	0.2972	3.715	0.0002028
remove	2.217	0.3349	6.619	3.604e-11
internet	0.5088	0.1508	3.375	0.0007389
free	1.205	0.1605	7.508	6.011e-14
business	0.9067	0.2153	4.211	2.54e-05
your	0.1759	0.04623	3.805	0.0001416
n000	2.287	0.5282	4.33	1.489e-05
money	0.5921	0.191	3.101	0.001932
hp	-2.445	0.3025	-8.083	6.336e-16
george	-12.5	2.114	-5.911	3.392e-09
data	-1.083	0.3537	-3.062	0.002198
n85	-2.187	0.8601	-2.543	0.011
technology	0.6019	0.3265	1.844	0.06525
meeting	-3.887	1.281	-3.036	0.002399
project	-1.92	0.6742	-2.848	0.004398
re	-0.7425	0.1569	-4.732	2.219e-06
edu	-1.648	0.305	-5.404	6.513e-08
conference	-4.767	1.857	-2.567	0.01026
cf.semicol	-1.042	0.3352	-3.109	0.001876
cf.exclaim	0.292	0.07791	3.749	0.0001779
cf.dollar	5.213	0.7637	6.827	8.683e-12
cf.pound	3.724	0.8727	4.267	1.977e-05
crl.longest	0.01195	0.002042	5.852	4.844e-09
crl.total	0.000823	0.0002031	4.052	5.083e-05

4. Using the refitted model, the estimated fitted values and a threshold of 0.50 for deciding spam, display a confusion matrix and calculate the overall error rate and the false positive rate.

```r
prediction_glm <- logistic_model_reduced %>%
  predict(new_data = train_data)%>%
  bind_cols(train_data[,58])

conf_mat<-prediction_glm%>%
  conf_mat(spam,.pred_class)

autoplot(conf_mat, type = "heatmap")
```




```
overall_error_rate<-
  sum(prediction_glm$spam!=prediction_glm$.pred_class)/
  nrow(prediction_glm)
false_positive_rate<-
  sum((prediction_glm$.pred_class[prediction_glm$spam==0])==1)/
  sum(prediction_glm$spam==0)
error<-
  as.data.frame(cbind(overall_error_rate,false_positive_rate))
pander(error,caption="")
```

overall_error_rate	false_positive_rate
0.07579	0.04878

5. Zero false positives are the goal. Using the fitted values find the threshold value that produces approximately zero false positives and re-display the confusion matrix

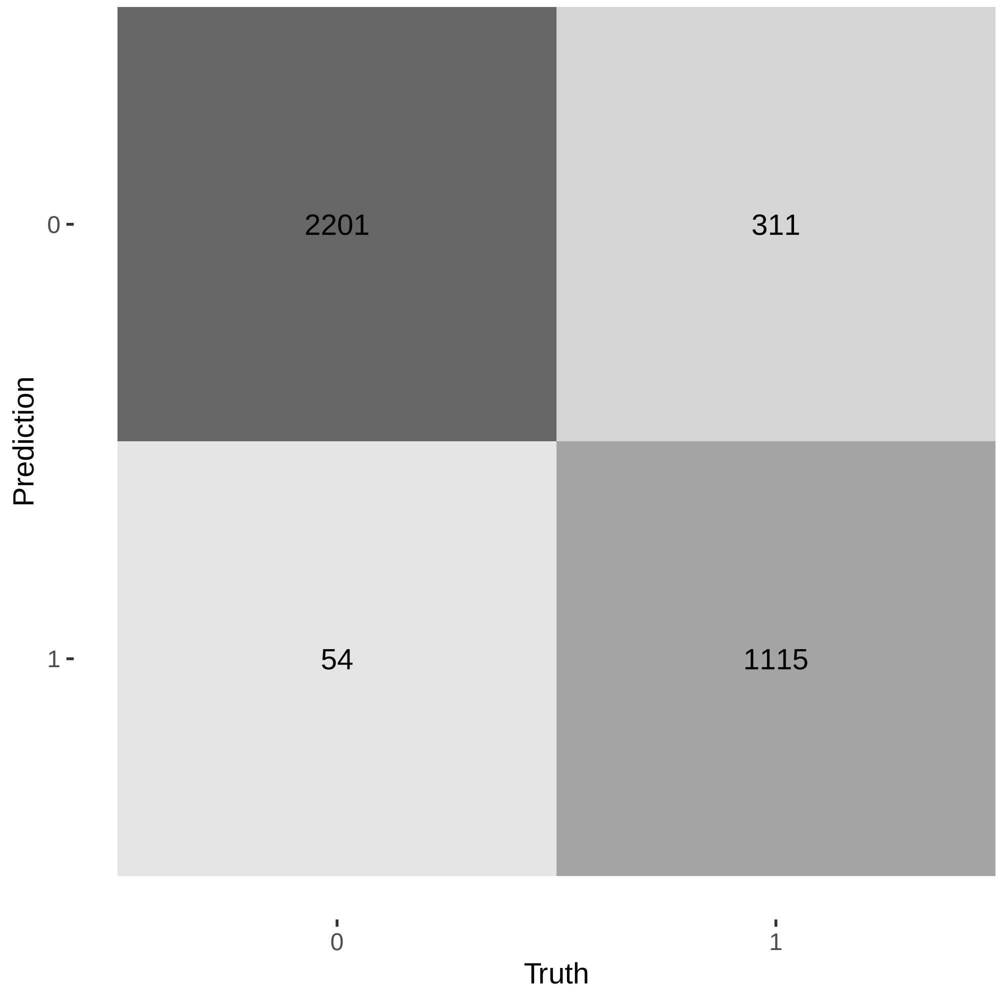
Zero false positives can only be achieved by letting every email message through. At that point, we would cease to have a spam filter! I am going to assume that a false positive rate of 2.5% or less is acceptable. Lowering the false positive rate also increases the overall error rate. The goal is to find a good balance. The confusion matrix below shows that the overall error rate increases to about 10% (from about 8%) when false positives are held below 2.5%. In this case, the threshold is 0.73.

```
#The following function takes a columns of probabilities (given by parsnip's predict function)
#and sorts them into categories (1,0) based on wether the probability meets a certain threshold)
prediction<-function(data,probability1,threshold){
  prediction<-rep(0,nrow(data))
  for(i in 1:nrow(data))
    {
    if(probability1[i]>=threshold)
      {
      prediction[i]<-1
      }
    else
      {
      prediction[i]<-0
      }
  }
  data<-cbind(data,prediction)
  return(data)
}
```

```
#this function finds a prediction threshold based on a desired rate of false positives.
#"Data" must be in the format of a parsnip prediction.  This function is specific to the
#spam dataset.

get_threshold<-function(data,prediction_column,false_pos_rate)
{
thresh<-c()
for (i in rev(1:100))
  {
  #generate predictions
  p<-prediction(data,prediction_column,threshold=i/100)
  #calculate false positive rate
  fp<-sum((p$prediction[p$spam==0])==1)/sum(p$spam==0)
  if(fp<=false_pos_rate)
      {
      thresh<-c(thresh,i/100)
      }
  }
  return(tail(thresh,1))
}
```

```
#generate predictions as probabilities
prediction_glm <- logistic_model_reduced %>%
  predict(new_data = train_data,type="prob")%>%
  bind_cols(train_data[,58])

#find a threshold for predictions based on logistic regression with
#a maximum false positive rate of 2.5%
threshold_log<-get_threshold(prediction_glm,prediction_glm$.pred_1,0.025)

#generate predictions using new threshold
prediction_glm_threshold<-tibble(prediction(prediction_glm,prediction_glm$.pred_1,threshold_log))%>%
  mutate(prediction=as.factor(prediction))
```

```
#produce confusion matrix
conf_mat<-prediction_glm_threshold%>%
  conf_mat(spam,prediction)

autoplot(conf_mat, type = "heatmap")
```



Part 2: LDA

1. Perform an LDA on the data using only the predictors you decided upon for the simplified logistic regression model. Display the summary results.

```
LDA_model<-discrim_linear("classification")%>%
  set_engine("MASS")%>%
  fit(formula,data=train_data)

LDA_model$fit
```

```
Call:
lda(spam ~ our + over + remove + internet + free + business +
    your + n000 + money + hp + george + data + n85 + technology +
    meeting + project + re + edu + conference + cf.semicol +
```

```
    cf.exclaim + cf.dollar + cf.pound + crl.longest + crl.total,
    data = data)

Prior probabilities of groups:
        0         1
0.6126053 0.3873947

Group means:
        our       over     remove   internet       free   business      your
0 0.1855033 0.04341463 0.01075831 0.04044346 0.06565854 0.05162749 0.4494102
1 0.5304698 0.17396914 0.26617111 0.21204067 0.51214586 0.28830996 1.3888920
         n000      money         hp     george       data         n85
0 0.006722838 0.01842572 0.88510421 1.25889135 0.15891353 0.180829268
1 0.238688640 0.22071529 0.01786816 0.00141655 0.01669705 0.006353436
  technology     meeting     project        re        edu  conference
0 0.15211086 0.224359202 0.123277162 0.4244967 0.28653659 0.054944568
1 0.02788219 0.002131837 0.005476858 0.1234222 0.01523142 0.002223001
  cf.semicol cf.exclaim  cf.dollar   cf.pound crl.longest crl.total
0 0.04974279  0.1087539 0.01202882 0.02372151    18.31885  159.8359
1 0.02072440  0.5123296 0.16811431 0.07187588   108.64656  468.2952

Coefficients of linear discriminants:
                      LD1
our          0.3688542422
over         0.5642088109
remove       1.1164593899
internet     0.3837343347
free         0.4113557819
business     0.2226522226
your         0.2551651982
n000         0.8690227796
money        0.4123905904
hp          -0.1618001047
george      -0.0478990063
data        -0.1954021594
n85         -0.1144865782
technology   0.0846461394
meeting     -0.1905952914
project     -0.1692157115
re          -0.1476753215
edu         -0.1555023002
conference  -0.2544450674
cf.semicol  -0.3377978322
cf.exclaim   0.2687165792
cf.dollar    1.1978614886
cf.pound     0.1860320845
crl.longest  0.0001089044
crl.total    0.0004825070
```

2. Interpret the summary results as best you can. We did not discuss this in class so this is a challenge step. A round of brews or "mocktail" to those who try this, once we can get back together.

The LDA output provides "prior probabilities," which indicate that approximately 38.7% of the training data is spam, while 61.3% of the training data is not spam. The output also provides "group means," which are the average of each predictor within each class. Consider the predictor "free." The group means suggest that spam messages are more likely to contain "free" (0.51 times per message on average) than non-spam messages (0.066 times per message on average). The output also contains "coefficients of linear discrimination" which specify the coefficients in the linear equation that is used to predict whether a message is spam or not.

3. As with the logistic model, use the fitted probabilities to establish a threshold that achieves near zero false positives.

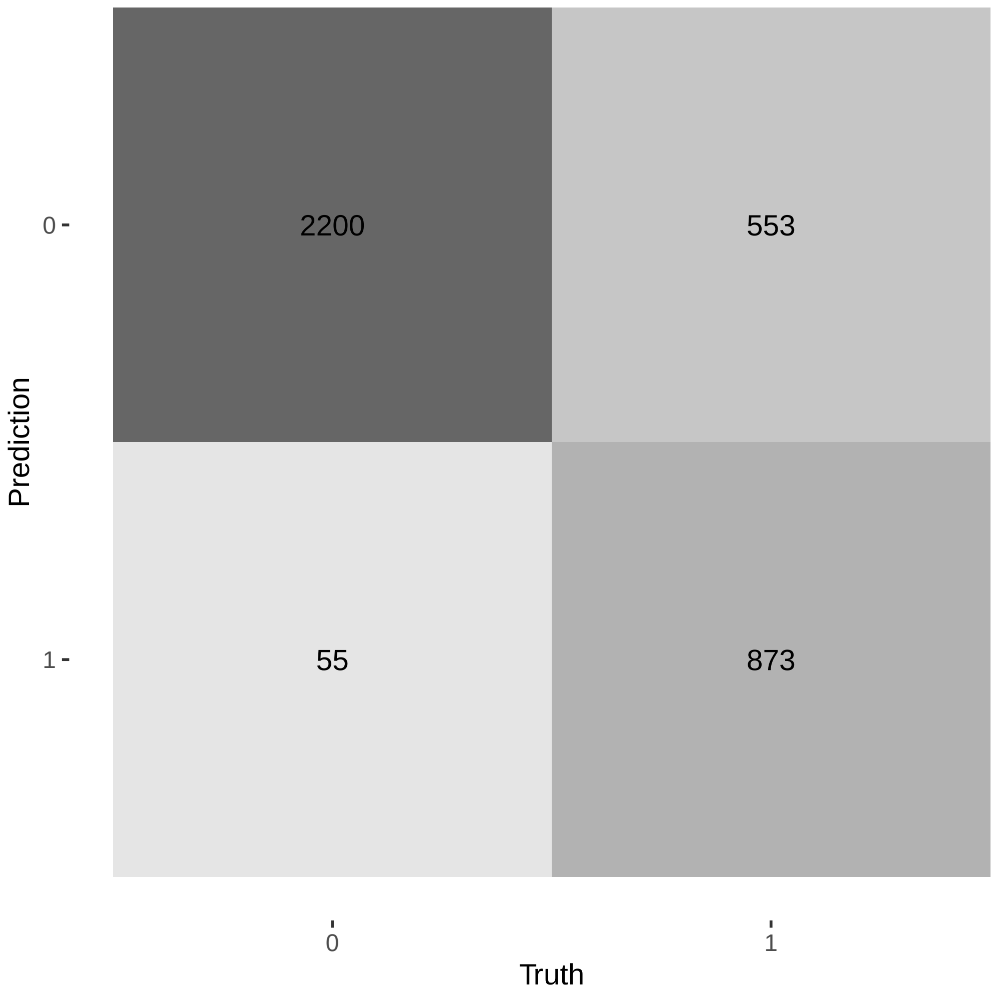
```
predictions_lda <- LDA_model %>%
  predict(new_data = train_data,type="prob")%>%
  bind_cols(train_data[,58])

#find a threshold for predictions based on lda with a maximun false positive rate of 2.5%
threshold_lda<-get_threshold(predictions_lda,predictions_lda$.pred_1,0.025)
```

For LDA, the threshold necessary to achieve a false positive rate below 2.5% is 0.72.

4. Display the confusion matrix from step 3. Using only this and the confusion matrix from step 5 above, which model would you recommend to use? Explain.

When we compare the confusion matrices from step 3 and 4, we see that the overall error rate for the lda model (~17%) is much higher than that of the logistic regression model (~10%). I would therefore recommend the logistic regression model.

```
#generate predictions using new threshold
prediction_lda_threshold<-tibble(prediction(predictions_lda,predictions_lda$.pred_1,threshold_lda))%>%
  mutate(prediction=as.factor(prediction))

#produce confusion matrix
conf_mat<-prediction_lda_threshold%>%
  conf_mat(spam,prediction)

autoplot(conf_mat, type = "heatmap")
```



Part 3: Cross-Validation

1. Perform an iterated 5-fold cross validation of 500 iterations each using both models only the original training data as follows. Use total error rate as the measure or "cost" function, not false positives.

The following tables show samples of the training and validation error rates generated by 5-fold cross validation. For each model, I use the prediction thresholds established above.

```
#This function is a modification of code from labs 4 and 5.  It takes a parsnip model
#object and makes predictions based on a preset threshold.  It then perfroms k-fold
#cross validation and returns the total error rate for the training and validation
#sets. This function is specific to the spam dataset.

k.fold.validator <- function(df, K,parsnipModel,threshold) {

  # 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, ]

    #clean probability predictions from parsnip model for the training set
```

```r
    prediction_train <- parsnipModel %>%
      predict(new_data = train.data,type="prob")%>%
      dplyr::select(-.pred_0)%>%
      rename(pred1=.pred_1)%>%
      bind_cols(train.data[,58])

    #uses the "prediction" function (see Part I, problem 5) to make predictions based on
    #a preset threshold
    train.predict<-as.data.frame(prediction(prediction_train,
                                            probability1=prediction_train$pred1,
                                            threshold))%>%
      mutate(prediction=as.factor(prediction))

    #calculates total prediction error
    train.error <- sum(train.predict$spam!=train.predict$prediction)/nrow(train.predict)

    #clean probability predictions from parsnip model for the holdout set
    prediction_holdout <- parsnipModel %>%
      predict(new_data = holdout.data,type="prob")%>%
      dplyr::select(-.pred_0)%>%
      rename(pred1=.pred_1)%>%
      bind_cols(holdout.data[,58])

    #uses the "prediction" function (see Part I, problem 5) to make predictions based on
    #a preset threshold.
    holdout.predict<-as.data.frame(prediction(prediction_holdout,
                                              probability1=prediction_holdout$pred1,
                                              threshold))%>%
      mutate(prediction=as.factor(prediction))

    holdout.error <- sum(holdout.predict$spam!=holdout.predict$prediction)/nrow(holdout.predict)
    tibble(train.error = train.error, valid.error = holdout.error)
  }

  # shuffle the data and create the folds
  indices <- sample(1:nrow(df))
  # if argument K == 1 we want to do LOOCV
  if (K == 1) {
    K <- nrow(df)
  }
  folds <- cut(indices, breaks = K, labels = F)
  # set error to 0 to begin accumulation of fold error rates
  errors <- tibble()
  # iterate on the number of folds
  for (i in 1:5) {
    holdout.indices <- which(folds == i, arr.ind = T)
    folded.errors <- fold.errors(df, holdout.indices)
    errors <- errors %>%
      bind_rows(folded.errors)
  }
 return(errors)
}
```

```r
#five-fold cross validation of LDA model repeated 500 times
lda_errors<-replicate(500,k.fold.validator(test_data, 5,LDA_model,threshold_lda),simplify = F)%>%
  purrr::reduce(rbind)

#five-fold cross validation of the logistic regression model repeated 500 times
log_errors<-replicate(500,k.fold.validator(test_data, 5,logistic_model_reduced,threshold_log),
                      simplify = F)%>%
  purrr::reduce(rbind)

#sample of the log_errors dataset
pander(head(log_errors,5),
       caption="Sample of error rates from cross-validation of the logistic model")
```

Table 4: Sample of error rates from cross-validation of the logistic model

train.error	valid.error
0.1128	0.1304
0.1182	0.1087
0.1182	0.1087
0.1141	0.125
0.1182	0.1087

```r
#sample of the log_errors dataset
pander(head(lda_errors,5),
       caption="Sample of error rates from cross-validation of the lda model")
```

Table 5: Sample of error rates from cross-validation of the lda model

train.error	valid.error
0.1957	0.125
0.1848	0.1685
0.1821	0.1793
0.1658	0.2446
0.1793	0.1902

2. Similar to what I did in the lab calculate the error rate on the non-fold data (i.e. training error rate) and the validation error on the fold for each fold for each of the 500 iterations. However, at the end do not take the mean of the iterations. Rather keep the separate results. Therefore, at the end you should have 500 values each for non-fold training errors and validation (fold) error rates for both the logistic regression and the LDA fit. In other words, you should have produced 500 estimates of 2 overall error rates for each of 2 models. If you build a data frame from this you should have one consisting of 500 rows and 4 columns.

The following table displays a sample of combined lda and logistic regression errors. I have transformed it into a "long" format to make it easier to plot.

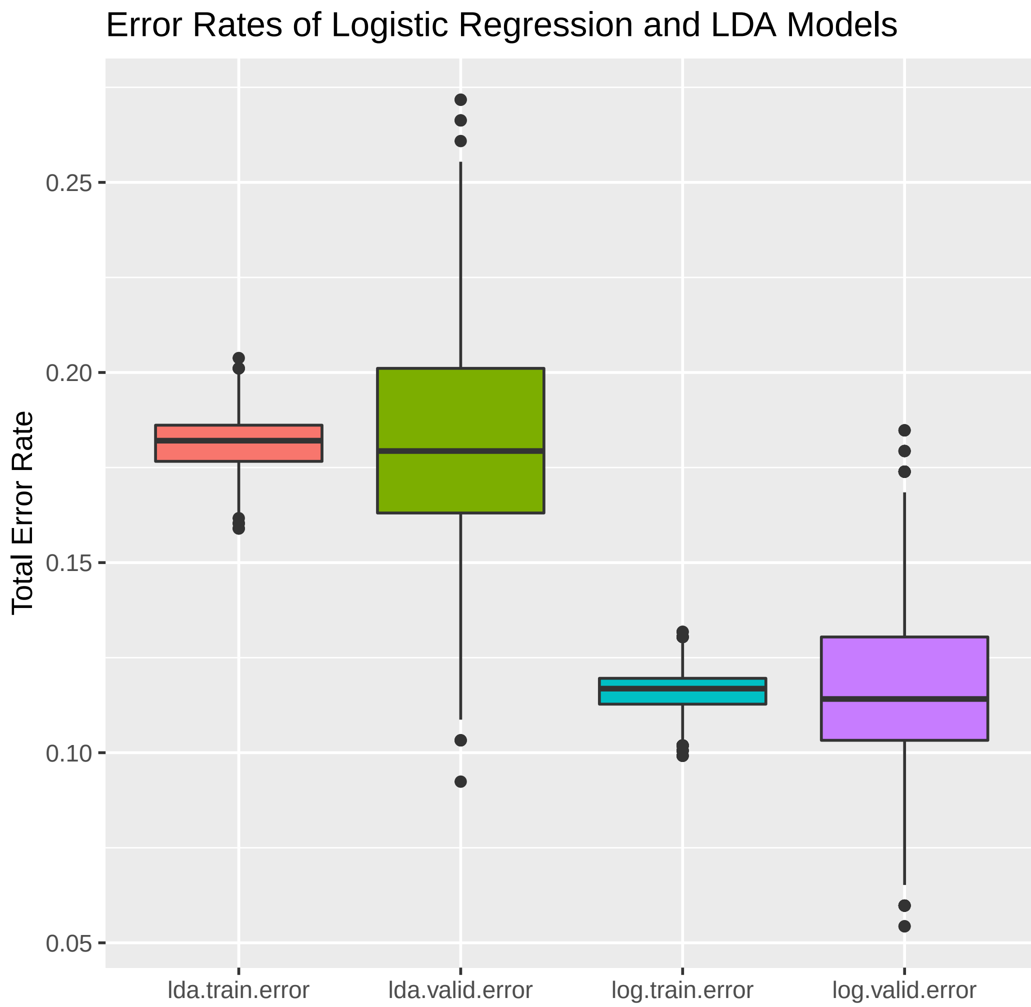
```r
#combine results from cross validation procedures.  Tidy the data for graphing.
df2 <- cbind(log_errors%>%rename(log.train.error=train.error,log.valid.error=valid.error),
             lda_errors%>%rename(lda.train.error=train.error,lda.valid.error=valid.error))%>%
  pivot_longer(c(log.train.error,log.valid.error, lda.train.error, lda.valid.error),
               names_to = "model",
               values_to = "error rate")

#a sample of the tidy data set
pander(head(df2,4),caption="Sample of combined log and lda error rates")
```

Table 6: Sample of combined log and lda error rates

model	error rate
log.train.error	0.1128
log.valid.error	0.1304
lda.train.error	0.1957
lda.valid.error	0.125

3. Display a ggplot consisting of 4 side-by-side boxplots from the data frame described in step 2.

```r
ggplot(data=df2,mapping=aes(x=as.factor(model), y=`error rate`, fill=model))+
  geom_boxplot()+
  labs(x="",y="Total Error Rate",title = "Error Rates of Logistic Regression and LDA Models")+
  theme(legend.position="none")
```



a. Challenge: Try discussing the similarities between this and doing a bootstrap. Is this like doing a bootstrap?

It is similar to a bootstrap in the sense that a calculation is repeated and the distribution of the results is then examined. There are, however, a few important differences: First, the bootstrap involves sampling *with* replacement, which allows for the repetition of individual observations within the sample. By contrast, K-fold cross validation involves sampling *without* replacement. The sample is split into folds, and each fold is used in turn to validate a model that is trained on the other folds. If we sampled with replacement, the folds would cease to be distinct from one another. Second, the bootstrap function would return one value. If, for example, we wanted to estimate a population mean, we could use the bootstrap to sample from a group of observations multiple times. The function would then return the mean of all the samples. By contrast, in iterated K-fold validation, we return the errors from each iteration so that we can examine their distribution across iterations.

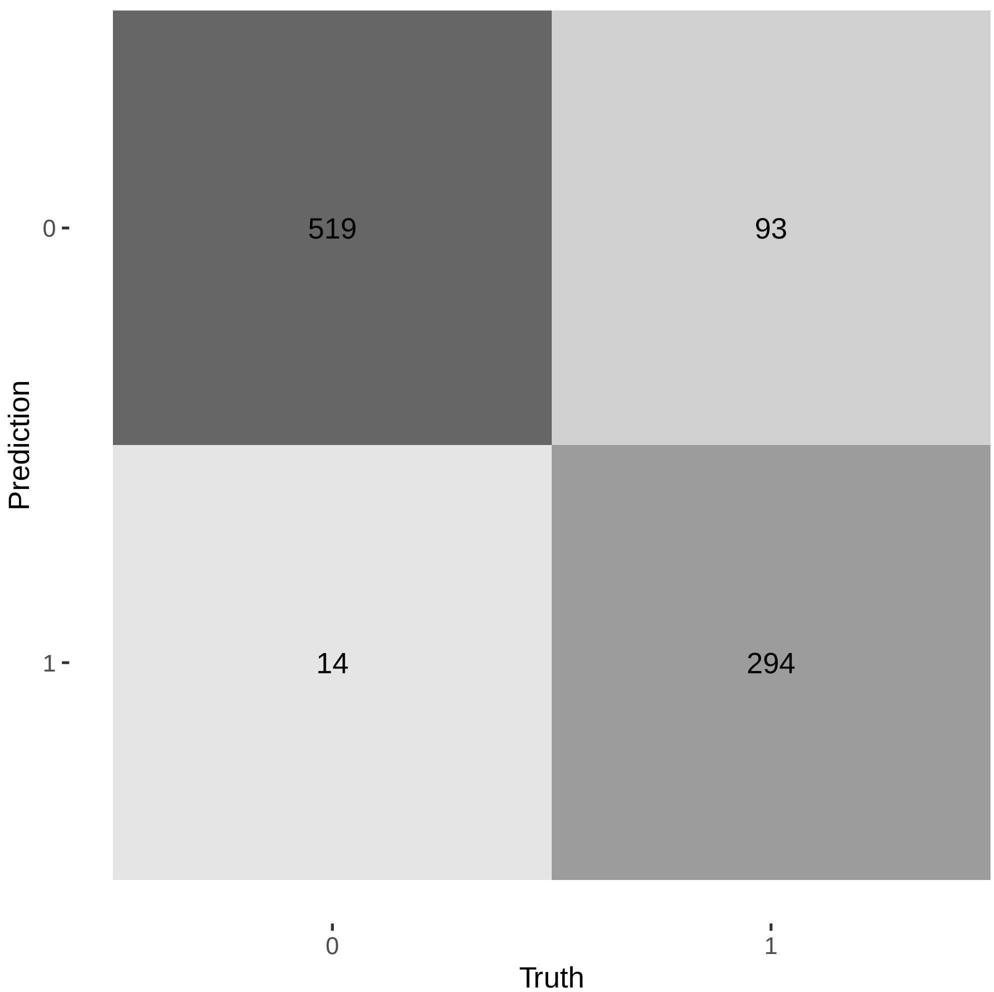
4. Based on the boxplots and consideration of the false positive goal, which model would recommend? Explain.

The boxplots confirm what we saw in the confusion matrices above: the logistic regression model has lower overall error rates when false positives are held below 2.5%. For that reason, I would recommend the logistic regression model over the lda model.

5. Using the fits from both models, classify the observations in the held-out test data and display the corresponding confusion matrices. (You should have 2 matrices.) Based on this do you stand by your recommendation in step 4? Explain.

The following confusion matrices show the results of the logistic regression and lda models when applied to the test data. Here again, the total error rate for the logistic model is lower. In the end, I would recommend the logistic regression model.

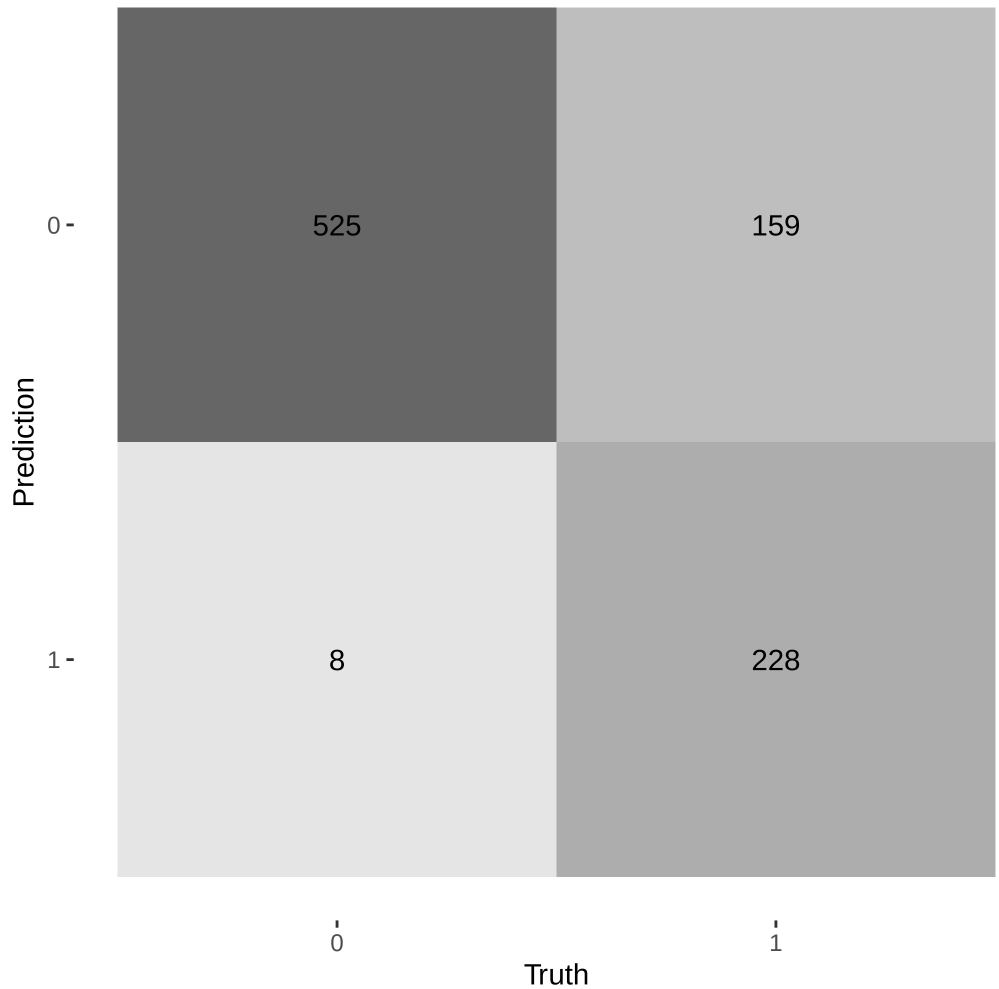
```
prediction_glm <- logistic_model_reduced %>%
  predict(new_data = test_data,type="prob")%>%
  bind_cols(test_data[,58])

#generate predictions using preset threshold
prediction_glm_threshold<-tibble(prediction(prediction_glm,prediction_glm$.pred_1,threshold_log))%>%
  mutate(prediction=as.factor(prediction))

#produce confusion matrix
conf_mat<-prediction_glm_threshold%>%
  conf_mat(spam,prediction)

autoplot(conf_mat, type = "heatmap")
```



```
overall_error_rate<-
  sum(prediction_glm_threshold$spam!=prediction_glm_threshold$prediction)/
  nrow(prediction_glm_threshold)
false_positive_rate<-
  sum((prediction_glm_threshold$prediction[prediction_glm_threshold$spam==0])==1)/
  sum(prediction_glm_threshold$spam==0)
error_logistic<-
  as.data.frame(cbind(overall_error_rate,false_positive_rate))
pander(error_logistic,caption="")
```

overall_error_rate	false_positive_rate
0.1163	0.02627

```
predictions_lda <- LDA_model %>%
  predict(new_data = test_data,type="prob")%>%
  bind_cols(test_data[,58])

prediction_lda_threshold<-tibble(prediction(predictions_lda,predictions_lda$.pred_1,threshold_lda))%>%
  mutate(prediction=as.factor(prediction))

#produce confusion matrix
conf_mat<-prediction_lda_threshold%>%
  conf_mat(spam,prediction)

autoplot(conf_mat, type = "heatmap")
```




```
overall_error_rate<-
  sum(prediction_lda_threshold$spam!=prediction_lda_threshold$prediction)/
  nrow(prediction_lda_threshold)
false_positive_rate<-
  sum((prediction_lda_threshold$prediction[prediction_lda_threshold$spam==0])==1)/
  sum(prediction_lda_threshold$spam==0)
error_lda<-
  as.data.frame(cbind(overall_error_rate,false_positive_rate))
pander(error_lda,caption="")
```

overall_error_rate	false_positive_rate
0.1815	0.01501