

**BUDT 758T**

**assignment #3: 100 PTS**

The goal of this homework is to give you additional familiarity with performance measure sharconcepts and introduce discriminant analysis. As with Assignment 2, you will need to create random partitions of a data set, build your model on the training data set, and then compute prediction errors using the test data set. You are required to complete this assignment in R—be sure to include the code you used and output any results you use!

**The Data**

Bikeshare systems are popular in major cities around the world and are increasingly viewed as an important mechanism to reduce auto traffic, improve air quality, reduce use of fossil fuels, and improve the health of the population. It is particularly important to be able to predict the use level of the bikesharing system in order to plan maintenance, distribution of the bikes, and make decisions on whether and where to add more bike stations.

This data is from Washington, DC’s Capital bikeshare system from the years 2011-2012. Riders can rent bikes from one location and return to a different location in the system. There are two types of users: CASUAL users, who rent the bike for a one-time fee, and REGISTERED users, who pay a yearly membership fee in exchange for unlimited bike rental.

The data in the accompanying file **bikeshare.csv** (posted on Canvas) is publicly available at the UC Irvine Machine Learning Repository:

<http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset>

The file contains daily information on the number of trips taken. The variables in the data set are:

1. MONTH: month of the year (January through December)
2. HOLIDAY: whether the day is a US holiday (YES) or not (NO)
3. WEEKDAY: whether the day is a weekday (YES) or not (NO)
4. WEATHERSIT: The values are (1) Clear/Few clouds (2) Misty (3) Light snow or light rain (4) Heavy rain, snow, or thunderstorms
5. TEMP Normalized temperature in Celsius. The values are derived as where and
6. ATEMP: Normalized “feels like” temperature in Celsius. The values are derived as where and
7. HUMIDITY: Normalized humidity on a scale of 0 to 1.
8. WINDSPEED: Normalized wind speed on a scale of 0 to 1.
9. CASUAL: count of bikes rented by casual bikeshare users.
10. REGISTERED: count of registered users.
11. COUNT: count of total rental bikes including both casual and registered.

**Assignment**

Please answer all questions in the dedicated space and upload on Canvas. Please ensure that your numbering of questions matches those below. Include any R code you used to answer each question with your response. You are welcome to include any output you wish to provide either with the appropriate question or at the end of your assignment in an appendix.

Remember: you are allowed to consult with others in the class on this assignment, but all submitted work must be your own (and don’t forget to include the names of anyone you consulted in the last question!).

Note for this assignment: there is some example code at the end of this assignment that may be helpful for plotting and running loops. The code also includes some basic LDA and QDA analysis code!

1. **10 points: Data Preparation**
   1. Read the data set in R.

bikeshare = read.csv('bikeshare.csv')

attach(bikeshare)

* 1. Change the WEATHERSIT variable to a factor variable in R. Use this new variable for the remainder of the assignment.

bikeshare$WEATHERSIT <- as.factor(bikeshare$WEATHERSIT)

* 1. Create a new variable, REG\_85, that is 1 if the percentage of registered bikes on a particular day was greater than 85% of the total count of bikes and 0 if it is not.

bikeshare$REG\_85 <- ifelse(bikeshare$REGISTERED/bikeshare$COUNT > 0.85,1,0)

* 1. Set the seed in R to 8726.

set.seed(8726)

* 1. Randomly partition the data set into a training data set and a test data set. Use 75% of the data as training data and hold out the remaining 25% as test data.

num\_obs <- nrow(bikeshare)

train\_obs <- sample(num\_obs, num\_obs\*0.75)

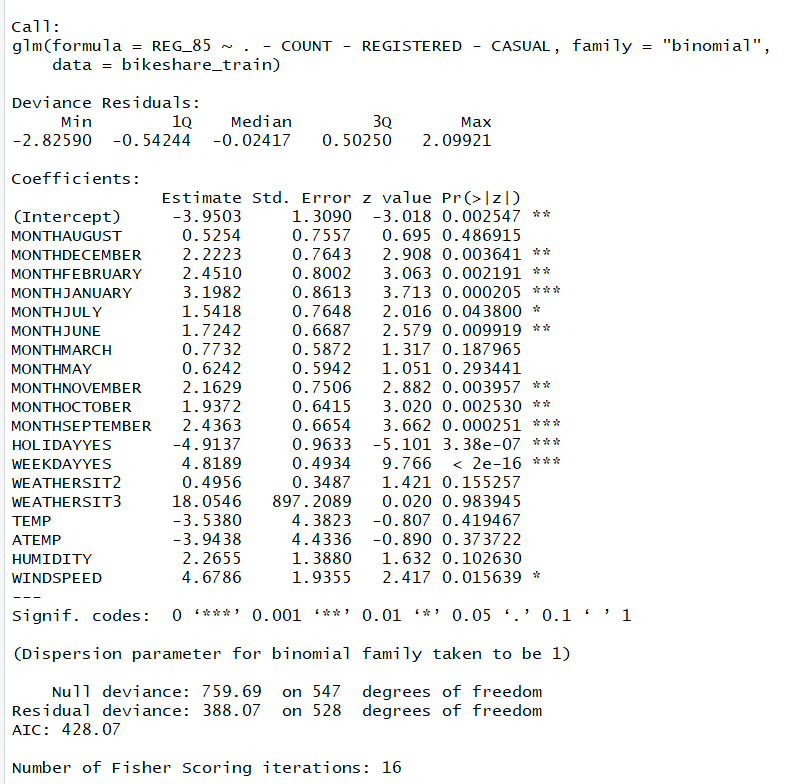
bikeshare\_train <- bikeshare[train\_obs,]

bikeshare\_test <- bikeshare[-train\_obs,]

1. **40 points: Run a logistic regression model to predict REG\_85using all other variables except COUNT, REGISTERED, and CASUAL. Use only the training data set for this. Report the summary of the model.**

fit1 <- glm(REG\_85~.-COUNT-REGISTERED-CASUAL, data=bikeshare\_train, family='binomial')

summary(fit1)



* 1. In looking at the summary, you notice WEATHERSIT3 has an unusually high p-value. Should we eliminate WEATHERSIT3 and rerun the model? Why or why not?

No, because our model has many other variables that are statistically significant and also a low AIC value. So, it doesn’t matter even if we had accidentally inserted some terrible variables into our model, our model still works fine.

Also, it’s not possible to remove WEATHERSIT3 level alone, we will have to remove the whole WEATHERSIT variable.

* 1. Calculate the accuracy, TPR, and TNR of your predictions (on the test set) using 13 different cutoff values: 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, and 0.99.

cutoffs=c(0.01,0.05,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95,0.99)

log\_acc=rep(0,13)

log\_TPR=rep(0,13)

log\_TNR=rep(0,13)

log\_preds <- predict(fit1,newdata=bikeshare\_test,type='response')

for (i in 1:13) {

log\_class <- ifelse(log\_preds>cutoffs[i],1,0)

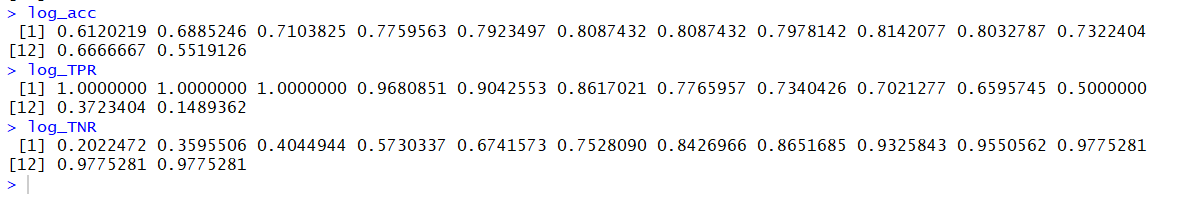
confuse\_test=table(bikeshare\_test$REG\_85,log\_class)

log\_acc[i]=(confuse\_test[1,1]+confuse\_test[2,2])/sum(confuse\_test)

log\_TPR[i]=confuse\_test[2,2]/(confuse\_test[2,1]+confuse\_test[2,2])

log\_TNR[i]=confuse\_test[1,1]/(confuse\_test[1,1]+confuse\_test[1,2])

}



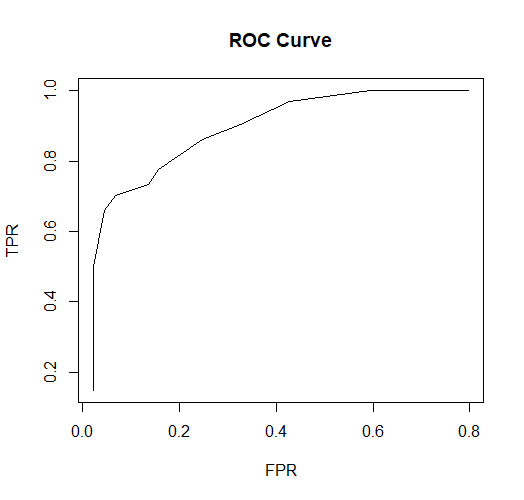
**Note**: You could speed up these calculations by doing a loop, like we discussed in class the first week. There is a simplified example of this kind of code at the end of this assignment. If you aren’t comfortable with that, though, you may do individual calculations for each cutoff instead. But it will take longer!

* + 1. What is the best cutoff to use for maximum accuracy in this data?

The best cutoff to use is 0.7 for maximum accuracy of 0.8142077 in this data.

* + 1. Use your TPR and TNR values to plot an ROC curve (note that you can plot a curve instead of points by using the type = “l” argument in the plot() function, if you wish) and report it below. Does this model appear to be performing better than having no model at all? How do you know?

plot(1-log\_TNR, log\_TPR, type='l', xlab='FPR', ylab='TPR', main='ROC Curve')



This model performs better than having no model at all. Because according to the above plot AUROC of this model is greater than the diagonal/random guessing.

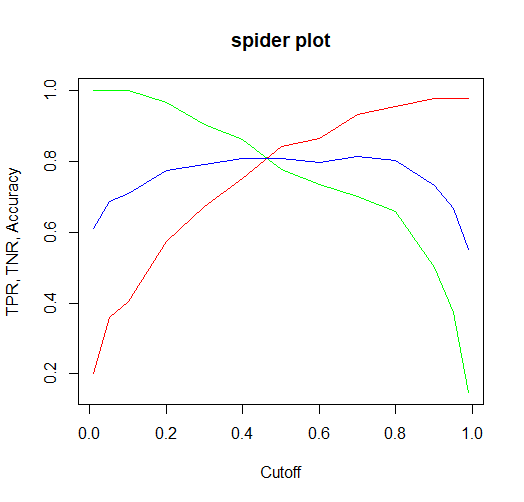
* + 1. Plot the TPR, TNR, and Accuracy values all on the same plot, using green for TPR, red for TNR, and blue for Accuracy to create a “spider plot” similar to the ones we saw in class and report it below. What cutoff value would you suggest using for this data? Why?

*Note that it does not need to be exact. You may estimate this value—which means you may have a different answer than your classmates!*

plot(cutoffs, log\_TPR, col='green', type='l', xlab='Cutoff', ylab='TPR, TNR, Accuracy', main='spider plot')

lines(cutoffs, log\_TNR, col='red')

lines(cutoffs, log\_acc, col='blue')



The cutoff value is 0.45. Because it is the intersection/trade off point of the TPR, TNR and Accuracy from the above spider plot.

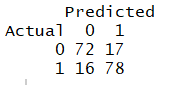
* 1. Calculate a new confusion matrix using the cutoff you suggested in b, part iii above. Report it below. Did your accuracy improve using this cutoff compared to the cutoffs you used to calculate accuracy in part (b) above?

cutoff = 0.45

log\_class <- ifelse(log\_preds>cutoff,1,0)

confuse\_test=table(bikeshare\_test$REG\_85,log\_class,dnn=c('Actual','Predicted'))

logistic\_acc=(confuse\_test[1,1]+confuse\_test[2,2])/sum(confuse\_test)



The accuracy using this cutoff is 0.8196721 which is more than the accuracy in part(b). So, the accuracy improved.

* 1. Consider the cost matrix below.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | ***Predicted*** | |
| ***Actual*** |  | REG\_85 = 1 | REG\_85 = 0 |
| REG\_85 = 1 | 0 | 135 |
| REG\_85 = 0 | 72 | 0 |

* + 1. Which is worse in this problem, a false negative or a false positive? How do you know?

In this problem, a false negative is worse because it has a higher cost associated in the cost matrix compared to a false positive.

* + 1. Using this cost matrix and the confusion matrix you created in part (c), calculate the total cost for the testing data.

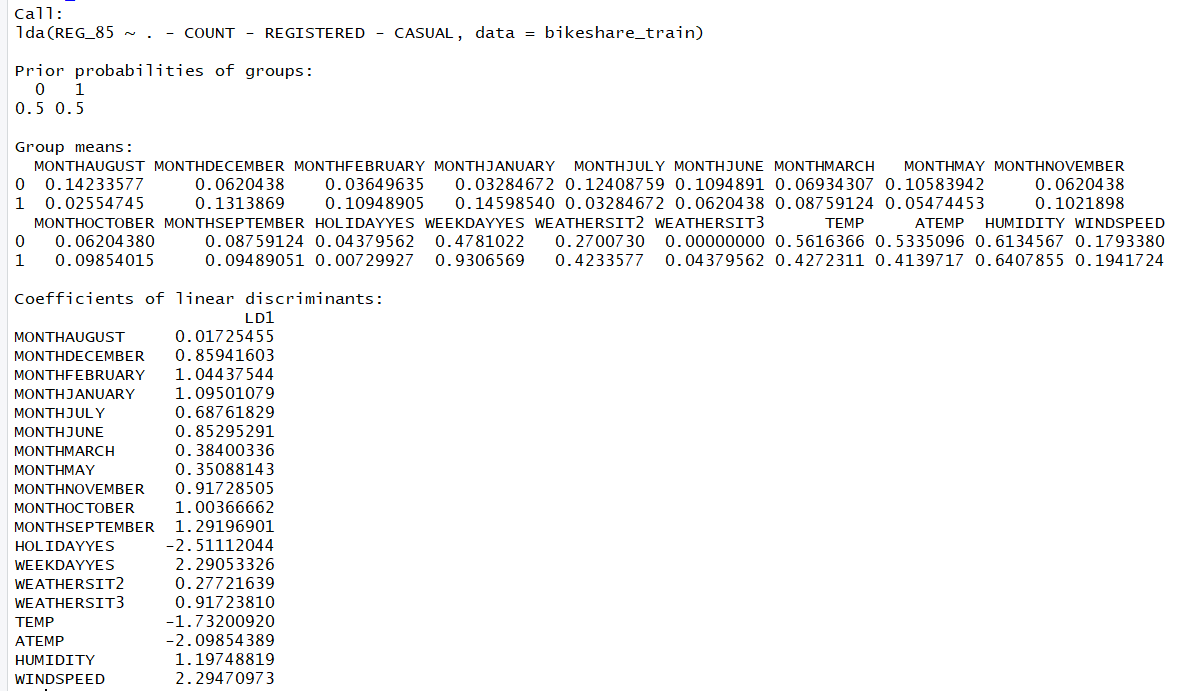
total\_cost=confuse\_test[1,2]\*72+confuse\_test[2,1]\*135

The total cost for the testing data is 3384.

1. **25 points: Run a linear discriminant analysis model to predict REG\_85using all other variables except COUNT, REGISTERED, and CASUAL. Use only the training data set for this. (Reminder: there is example code for running LDA/QDA and getting predictions at the end of this assignment!)**

library(MASS)

lda\_model <- lda(REG\_85~.-COUNT-REGISTERED-CASUAL, data=bikeshare\_train)



* 1. Use your LDA model and your cutoff from 2(b) part iii above to create a confusion matrix for the testing data. Report the confusion matrix below and calculate the accuracy.

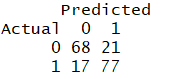
lda\_predict=predict(lda\_model,newdata=bikeshare\_test)

lda\_preds <- lda\_predict$posterior[,2]

lda\_class <- ifelse(lda\_preds > cutoff,1,0)

confuse\_test2 <- table(bikeshare\_test$REG\_85,lda\_class,dnn=c('Actual','Predicted'))

accuracy2 <- (confuse\_test2[1,1]+confuse\_test2[2,2])/sum(confuse\_test2)



The accuracy here is 0.7923497.

* 1. Predicting from an LDA results in two separate predictions: $posterior, which gives the Y=0 and Y=1 probabilities, and $class, which classifies points for you into Y=0 and Y=1 directly. If you had used the classifications from $class, would you have gotten the same confusion matrix as you did in 3(a)? Why or why not?

We would have got a different confusion matrix than from 3(a) because $class from predict uses 0.5 as the cutoff but we have used 0.45 as the cutoff above.RE

* 1. Using the cost matrix from 2(d) above, what is the total cost of your classifications in 3(a)?

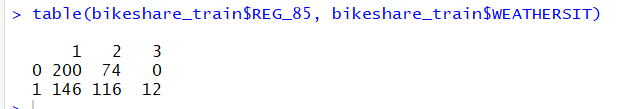
total\_cost2=confuse\_test2[1,2]\*72+confuse\_test2[2,1]\*135

The total cost here is 3807.

* 1. We could try to run QDA on this same problem, but if we do, we will get an error in R telling us that QDA could not run due to one of the variables not having enough information to create a model. This usually happens when a particular level of a categorical variable is missing information for one of the classes.
     1. Look at the distribution of each of the four categorical variables (MONTH, WEEKDAY, HOLIDAY, and WEATHERSIT) compared to REG\_85. Which variable is causing the error in R?

WEATHERSIT

* + 1. Why does this cause an error for QDA but not for LDA?



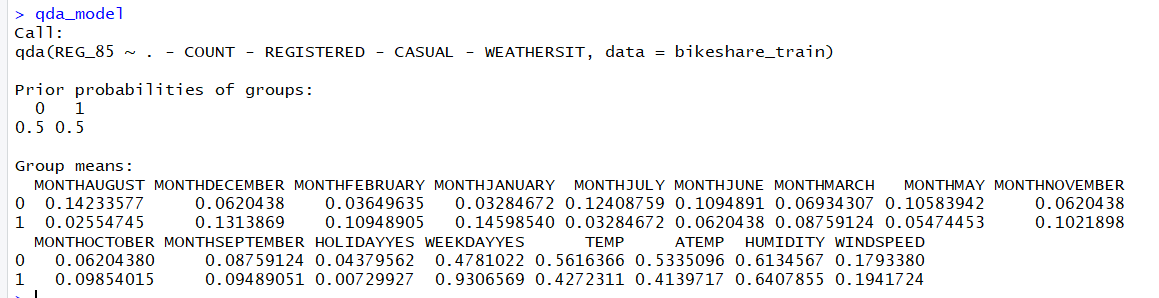
WEATHERSIT3 has no data points in the category of REG\_85=0.

The issue is that QDA fits a separate distribution for each Y=1 and Y=0 category. If there is no value in some category for a particular X variable, QDA will not be able to estimate the distribution.

But incase of LDA as long as there's information for either Y=1 or Y=0, LDA can use that to create the distributions since it isn't constrained by a single value of Y.

1. **10 points: Run a QDA model to predict REG\_85using all other variables except COUNT, REGISTERED, CASUAL, and the variable you identified in 3(d) above. Use only the training data set for this. (Reminder: there is example code for running LDA/QDA and getting predictions at the end of this assignment!)**

qda\_model <- qda(REG\_85~.-COUNT-REGISTERED-CASUAL-WEATHERSIT, data=bikeshare\_train)



* 1. Use your QDA model and your cutoff from 2(b) part iii above to create a confusion matrix for the testing data. Report the confusion matrix below and calculate the accuracy.

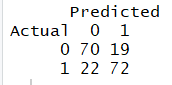
qda\_predict <- predict(qda\_model,newdata=bikeshare\_test)

qda\_preds <- qda\_predict$posterior[,2]

qda\_class <- ifelse(qda\_preds > cutoff,1,0)

confuse\_test3 <- table(bikeshare\_test$REG\_85,qda\_class,dnn=c('Actual','Predicted'))

accuracy3 <- (confuse\_test3[1,1]+confuse\_test3[2,2])/sum(confuse\_test2)



The accuracy here is 0.7759563.

* 1. Using the cost matrix from 2(d) above, what is the total cost of your classifications in 4(a)?

total\_cost3=confuse\_test3[1,2]\*72+confuse\_test3[2,1]\*135

The total cost here is 4338.

1. **15 points: Model Comparison**
   1. Based on accuracy, which model would you choose here: logistic, LDA, or QDA? Is it better than the accuracy associated with having no model at all?

Based on accuracy, I would choose the logistic model because it has got the highest accuracy (0.8196721) among all the three models.

countOf1 = sum(ifelse(bikeshare\_test$REG\_85==1,1,0))/nrow(bikeshare\_test)

baseline\_test <- ifelse(bikeshare\_test$REG\_85==1,1,1)

baseline\_cf <- table(bikeshare\_test$REG\_85, baseline\_test, dnn=c('Actual', 'Predicted'))

baseline\_accuracy <- baseline\_cf[2,1]/(baseline\_cf[1,1]+baseline\_cf[2,1])

Most common class is 1 as there are more 1’s then 0’s in the testing data.

Baseline accuracy is 0.5136612 which is lower than accuracy of logistic model. Hence, logistic model is better than having no model at all based on accuracy.

* 1. Based on cost, which model would you choose here: logistic, LDA, or QDA? Is it better than the cost associated with having no model at all?

Based on cost, I would choose the logistic model because it has got the least cost (3384) among all the three models.

baseline\_cost <- baseline\_cf[1,1]\*72

The cost associated with having no model at all is 6408 which is higher than the cost associated with logistic model. Hence, logistic model is better than having no model at all based on cost.

**Example Code for running loops using accuracy:**

# First, create a vector with your cutoff values and another vector to store the calculated accuracy values

## Note that they should have the same length!

cutoffs=c(0.2,0.4,0.6,0.8)

log\_acc=rep(0,4)

## Next, run predicted probabilities from your model using the testing data.

## You only need to do this once, since the probability predictions don’t change

log\_preds=predict(model1,newdata=test\_data,type="response")

## Now, consider a for loop. It should run through all your cutoff values and calculate an accuracy for each one

for(i in 1:4){

## Classify probabilities according to the appropriate cutoff

log\_class=ifelse(log\_preds>cutoffs[i],1,0)

## This is just one way to calculate accuracy, but you can save a confusion matrix and then use the entries in the matrix for whatever you need. So since true positives are in the 2nd row and 2nd column of the confusion matrix ([2,2]), and true negatives are in the 1st row and 1st column of the confusion matrix ([1,1]), I can call the table like a matrix to get an overall accuracy calculation:

confuse\_test=table(test\_data$REG\_85,log\_class)

log\_acc[i]=(confuse\_test[1,1]+confuse\_test[2,2])/sum(confuse\_test)

}

**Example Code for plotting multiple curves on a single plot:**

## First, plot one curve, then add other curves to it:

plot(X, Y, type=“l”)

lines(X, Z)

## This will plot the X-Z curve on the same plot as the X-Y curve

**Example Code for running LDA and predicting:**

## LDA and QDA are in the MASS package, so first install/load MASS

library(MASS)

## LDA code runs the same way as any other model

lda\_model = lda(Y~X1+X2, data=training\_data)

## Predictions are slightly different, because LDA and QDA includes classifications and probability predictions when you do predict()

lda\_predict=predict(lda\_model,newdata=test\_data)

## the $class option gives you classifications for every point in the test data here:

lda\_predict$class

## the $posterior option gives you probabilities both for Y=0 and Y=1 (note that “posterior” comes from Bayesian probability calculations; it essentially just means after fitting the distributions, what is the probability a point is Y=0 instead of Y=1)

lda\_predict$posterior

## Since we always predict for Y=1, we want the second column of posterior probabilities:

lda\_preds=lda\_predict$posterior[,2]

## Then we can use these predicted probabilities just like any other predicted probabilities, for example from the logistic regression model.

## QDA works exactly the same as LDA. You simply use the qda() function instead of lda()