MiniProject 3:

STAT 6340

Raheel Ahmed

Rsa170130

Question 1

1. **Perform an exploratory analysis of data.**

In commented portion of the code. We can see that there are 1000 subjects in the database and 20 features, with the one binary response being Default. Checking status, history, savings, and other plans were found to have a noticeable negative correlation with Default, whereas duration and amount had a strong positive correlation.

1. **Build a “reasonably good” logistic regression model for these data. There is no need to explore interactions. Carefully justify all the choices you make in building the model.**

I checked to see correlations between all the variables and default and the ones that I chose in my logistic regression model had both a high magnitude of correlation and a lower p-value in our full and partial logistic regression model

1. **Write the final model in equation form. Interpret the estimated coefficients of at least two predictors. Provide training error rate for the model.**
   1. Score = 1.60477871 - 0.58096911\*checkingstatus1 - 0.37347583\*history - 0.22999377 \*savings + 0.17344665\*installment + 0.03794233\*duration - 0.26281146\*otherplans

Final Equation: 1/(1 + exp(-Score))

* 1. For every 1 unit increase in installment the log odds of Defaulting increase by 0.173. For every 1 unit of increase in savings the log odds of Defaulting decrease by 0.22999.
  2. Training Error Rate: 0.245

Question 2

Note: Major points excluding evaluations of performance have been commented in the code below, such as implementations of LOOCV, KNN Tuning, etc.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model Type | Error Rate | Sensitivity | Specificity | LOOCV |
| Logistic Regression with all predictors | 0.214 | 0.5333333 | 0.8942857 | 0.249000  0.248446(bias adjusted) |
| Custom Logistic Regression Model | 0.245 | 0.4333333 | 0.8928571 | 0.24900  0.248907(bias adjusted) |
| LDA Model | 0.223 | 0.5400000 | 0.8785714 | 0.239 |
| QDA Model | 0.177 | 0.6966667 | 0.8257143 | 0.278 |
| KNN | 0.288 | 0.08333333 | 0.98428571 | 0.288 |

|  |  |  |
| --- | --- | --- |
| Model Type | ROC Curve | AUC |
| Logistic Regression with all predictors |  | 0.8338 |
| Custom Logistic Regression Model |  | 0.8031 |
| LDA Model |  | 0.8322 |
| QDA Model |  | 0.8907 |
| KNN |  | 0.65 |

1. **Compare the results from various classifiers. Which classifier would you recommend? Justify your answer.**

It seems in our case that QDA seemed to perform best when considering most of our evaluation methods. In terms of the ROC curves, KNN performed the worst, with the lowest AUC. Logistic regression of both the custom model and the full model were fairly similar to LDA and QDA, but QDA did have the highest AUC.

Similarly, in terms of error rate, KNN performed least well. Both models of logistic regression were close to LDA and QDA, but QDA did have the lowest error rate. Additionally, for all these models, specificity was noticeably higher than sensitivity, most likely because these models were optimizing for accuracy, but the underlying pattern of QDA outperforming remained, although the other models excluding KNN were close in performance, even while considering these metrics of evaluation.

Code

library**(**boot**)**

library**(**e1071**)**

library**(**pROC**)**

library**(**class**)**

library**(**MASS**)**

# Read the data into a dataframe variable

german\_data **<-** read.csv**(**"germancredit.csv", stringsAsFactors **=** T**)**

# Print number of observations in the german data and provide info on its features / response

nrow**(**german\_data**)**

str**(**german\_data**)**

# Proportion of observations that are defaulting

pie**(**table**(**german\_data**$**Default**))**

# Check to see which variables tend to have a strong correlation with Default

cor**(**german\_data**)[**,1**]**

# Produce a logistic regression model using all the predictors

all\_lr **<-** glm**(**Default **~** ., data **=** german\_data, family **=** "binomial"**)**

summary**(**all\_lr**)**

# Produce a logistic regression model that uses predictors that appeared to have a very low p-value in our previous model

partial\_lr **<-** glm**(**Default **~** checkingstatus1 **+** history **+** savings **+** installment **+** duration **+** otherplans, data **=** german\_data, family **=** "binomial"**)**

summary**(**partial\_lr**)**

coef**(**partial\_lr**)**

# Dropped variables are still significant, but model still seems to be fair in its interpretation

anova**(**partial\_lr, all\_lr, test **=** "Chisq"**)**

# Calculate probabilities and predictions for our partial logistic regression model

partial\_lr.prob **<-** predict**(**partial\_lr, type**=**"response"**)**

partial\_lr.pred **<-** ifelse**(**partial\_lr.prob **>=** 0.5, 1, 0**)**

# Error rate of partial logistic regression model

1 **-** mean**(**partial\_lr.pred **==** german\_data**$**Default**)**

# Confusion matrix and sensitivity, specificity provided

table**(**partial\_lr.pred, german\_data**$**Default**)**

c**(**130**/(**130**+**170**)**, **(**625**/(**625**+**75**)))**

# Create roc variable, print to obtain AUC and plot to show curve

roc.part\_lr **<-** roc**(**german\_data**$**Default, partial\_lr.prob**)**

roc.part\_lr

plot**(**roc.part\_lr, legacy.axes**=**T**)**

# Calculate LOOCV estimate of test error for the partial log-reg model

cost **<-** **function(**r, pi **=** 0**){**mean**(**abs**(**r **-** pi**)** **>** 0.5**)}**

part\_lr.err **<-** cv.glm**(**german\_data, partial\_lr, cost, K**=**nrow**(**german\_data**))**

part\_lr.err**$**delta

# Calculate probabilities and predictions for our full logistic regression model

all\_lr.prob **<-** predict**(**all\_lr, type**=**"response"**)**

all\_lr.pred **<-** ifelse**(**all\_lr.prob **>=** 0.5, 1, 0**)**

# Error rate of full logistic regression model

1 **-** mean**(**all\_lr.pred **==** german\_data**$**Default**)**

# Confusion matrix and sensitivity, specificity provided

table**(**all\_lr.pred, german\_data**$**Default**)**

c**(**160**/(**160**+**140**)**, **(**626**/(**626**+**74**)))**

# Create roc variable, print to obtain AUC and plot to show curve

roc.all\_lr **<-** roc**(**german\_data**$**Default, all\_lr.prob**)**

roc.all\_lr

plot**(**roc.all\_lr, legacy.axes**=**T**)**

#2b. Our own estimate of LOOCV estimation for full logistic regression model

sum\_misclass **<-** 0

**for(**i **in** 1**:**nrow**(**german\_data**)){**

# Fit to training excluding one subject, predict the subject, then find error

excluded\_subject **<-** german\_data**[**i,**]**

remaining\_train **<-** german\_data**[-**i,**]**

log\_fit **<-** glm**(**Default **~** ., data **=** remaining\_train, family**=**"binomial"**)**

prob **<-** predict**(**log\_fit, excluded\_subject, type**=**"response"**)**

**if(**prob **>=** 0.5**){**

pred **=** T

**}**

**else{**

pred **=** F

**}**

err **<-** 1 **-** **(**pred **==** excluded\_subject**$**Default**)**

# Sum error

sum\_misclass **<-** sum\_misclass **+** err

**}**

# Find average of summed error to obtain LOOCV estimate

**(**sum\_misclass **/** nrow**(**german\_data**))**

#2c. Use package to estimate LOOCV for full log-reg model

cost **<-** **function(**r, pi **=** 0**){**mean**(**abs**(**r **-** pi**)** **>** 0.5**)}**

all\_lr.err **<-** cv.glm**(**german\_data, all\_lr, cost, K**=**nrow**(**german\_data**))**

all\_lr.err**$**delta

################################################################################

################################################################################

# Fit an LDA model with all predictors

lda.fit **<-** lda**(**Default **~** ., data**=**german\_data**)**

lda.pred **<-** predict**(**lda.fit, german\_data**)**

# Find error rate for LDA model

1 **-** mean**(**lda.pred**$**class **==** german\_data**$**Default**)**

# Print confusion matrix and sensitivity, specificity

table**(**lda.pred**$**class, german\_data**$**Default**)**

c**(**162**/(**162**+**138**)**, 615**/(**615**+**85**))**

# Calculate roc variable for LDA then print it to find AUC, then plot to show curve

roc.lda **<-** roc**(**german\_data**$**Default, lda.pred**$**posterior**[**,1**])**

roc.lda

plot**(**roc.lda, legacy.axes**=**T**)**

# Calculate LOOCV for LDA using same principles in 2b.

lda\_misclass **<-** 0

**for(**i **in** 1**:**nrow**(**german\_data**)){**

l\_fit **<-** lda**(**Default **~** ., data**=**german\_data**[-**i, **])**

l\_pred **<-** predict**(**l\_fit, german\_data**[**i,**])**

err **<-** 1 **-** **(**l\_pred**$**class **==** german\_data**[**i,**]$**Default**)**

lda\_misclass **<-** lda\_misclass **+** err

**}**

**(**lda\_misclass**/**nrow**(**german\_data**))**

################################################################################

################################################################################

# Fit a QDA model on all predictors in the german data

qda.fit **<-** qda**(**Default **~** ., data **=** german\_data**)**

qda.pred **<-** predict**(**qda.fit, german\_data**)**

# Calculate error rate for QDA model on training data

1 **-** mean**(**qda.pred**$**class **==** german\_data**$**Default**)**

# Print confusion matrix and sensitivity, specificity

table**(**qda.pred**$**class, german\_data**$**Default**)**

c**(**209**/(**209**+**91**)**, **(**578**/(**578**+**122**)))**

# Make roc variable for QDA model and print to show AUC, then plot to show curve

roc.qda **<-** roc**(**german\_data**$**Default, qda.pred**$**posterior**[**,1**])**

roc.qda

plot**(**roc.qda, legacy.axes**=**T**)**

# Find LOOCV for QDA model using same principles as in 2b.

qda\_misclass **<-** 0

**for(**i **in** 1**:**nrow**(**german\_data**)){**

q\_fit **<-** qda**(**Default **~** ., data**=**german\_data**[-**i, **])**

q\_pred **<-** predict**(**q\_fit, german\_data**[**i,**])**

err **<-** 1 **-** **(**q\_pred**$**class **==** german\_data**[**i,**]$**Default**)**

qda\_misclass **<-** qda\_misclass **+** err

**}**

**(**qda\_misclass**/**nrow**(**german\_data**))**

###############################################################################

###############################################################################

# Turn factors into numerical data for each qualitative feature

german\_data**[**,c**(**2,4,5,7,8,10,11,13,15,16,18,20,21**)]** **<-**

sapply**(**german\_data**[**,c**(**2,4,5,7,8,10,11,13,15,16,18,20,21**)]**, as.numeric**)**

# Tune knn with cross validation for n folds

knn.cross **<-** tune.knn**(**x **=** german\_data**[**,**-**1**]**,

y **=** as.factor**(**german\_data**[**,1**])**,

k **=** 1**:**200,

tunecontrol**=**tune.control**(**sampling **=** "cross"**)**,

cross**=**nrow**(**german\_data**))**

# Print to find optimal K

summary**(**knn.cross**)**

optim\_K **<-** 38

# Make a KNN classifier using the optimal K

classifier\_knn **<-** knn**(**train **=** german\_data**[**,**-**1**]**,

test **=** german\_data**[**,**-**1**]**,

cl **=** german\_data**$**Default,

k **=** optim\_K, prob**=**T**)**

# Store probabilities for ROC curve

knn\_probs **<-** attr**(**classifier\_knn, "prob"**)**

# Print error rate for KNN

1 **-** mean**(**classifier\_knn **==** german\_data**$**Default**)**

# Print confusion matrix and sensitivity, specificity for KNN model

table**(**classifier\_knn, german\_data**$**Default**)**

c**(**25**/(**25**+**275**)**, **(**689**/(**689**+**11**)))**

# Produce ROC curve for KNN model of german dataset

roc.knn **<-** roc**(**german\_data**$**Default, knn\_probs**)**

# Print variable to find AUC then plot the curve

roc.knn

plot**(**roc.knn, legacy.axes**=**T**)**