# PubH 6886 Fall 2025 Assignment Two

AUTHOR
Annie Allred

```r
library(tidyverse)
library(caret)
library(pROC)
library(caret)
library(naivebayes)
library(psych)
library(class)
library(MASS)
```

## Question One

(40 pts.) The csv file `reg_data.csv` contains simulated data with 200 observations on 7 variables: `x1`, `x2`, `x3`, `x4`, `x5`, `x6`, and `y`. In this problem, you will write your own code to conduct leave-one-out cross-validation (LOOCV) and 5-fold cross-validation (CV) to compute estimates for the test error of a linear model with `y` as the response and all other variables as predictors.

### (a)

(15 pts.) Write your own code to compute the LOOCV RMSE for the linear model. Provide the LOOCV estimate for the RMSE.

The LOOCV estimate for the RMSE from my code is 1.62

```r
# setting the seed
set.seed(1234)

# reading in the data
regData <- read.csv("reg_data.csv")
# Creating an empty vector for the for loop
loocv_mseV <- numeric(200)

# looping through the data
for (i in 1:dim(regData)[1]){
  # splitting the data
  loocv_test <- regData[i,]
  loocv_train <- regData[-i,]
  # the linear model
  loocv_lm <- lm(y ~ ., data = loocv_train)
  # the predicted outcome
  loocv_lm_pred <- predict(loocv_lm,
                           newdata = loocv_test,
                           type = "response")
  # the mse
  loocv_mse <- (loocv_lm_pred - loocv_test$y)^2
  # adding the mse to a vector
  loocv_mseV[i] <- loocv_mse
}

# the rmse
(loocv_rmse <- sqrt(mean(loocv_mseV)))
```

```
[1] 1.619946
```

## (b)

(5 pts.) Use the `train()` function from the caret package to confirm that your LOOCV estimate is correct.

The LOOCV estimate from the `train()` is 1.62, which matches the LOOCV estimate from my code.

```
# setting the seed
set.seed(1234)

# using train() to find the LOOCV estimate
(loocv_lm <- train(x = regData[,1:6], y = regData$y, method = "lm",
                  trControl = trainControl(method = "LOOCV")))
```

```
Linear Regression

200 samples
  6 predictor

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 199, 199, 199, 199, 199, 199, ...
Resampling results:

  RMSE      Rsquared   MAE
  1.619946  0.6330772  1.279537

Tuning parameter 'intercept' was held constant at a value of TRUE
```

## (c)

(15 pts.) Write your own code to compute the 5-fold CV RMSE for the linear model. Provide the 5-fold CV estimate for the RMSE. Since the rows of the data set are already randomly ordered, use rows 1 - 40 for fold 1, 41 - 80 for fold 2, 81 - 120, for fold 3, 121 - 160 for fold 4, and 161 - 200 for fold 5.

The 5-fold CV estimate for the RMSE from my code is 1.62

```
# setting the seed
set.seed(1234)
# splitting the data and creating training and test sets
kfcv_train <- list(41:200, c(1:40, 81:200), c(1:80, 121:200),
                   c(1:120, 161:200),1:160)
kfcv_test  <- list(1:40,41:80,81:120,121:160,161:200)
# Creating an empty vector for the for loop
kfcv_rmseV <- numeric(5)

# Running a for loop that goes along the kfcv_train sequence of datasets
for (i in seq_along(kfcv_train)) {
  # the linear model
  kfcv_lm <- lm(y ~ ., data = regData[kfcv_train[[i]], ])
  # the predicted outcome
  kfcv_lm_pred <- predict(kfcv_lm, newdata = regData[kfcv_test[[i]], ])
  # the rmse and adding it to rmse vector
  kfcv_rmseV[i] <- sqrt(mean((regData$y[kfcv_test[[i]]] - kfcv_lm_pred)^2))
}

# the rmse
(kfcv_rmse <- mean(kfcv_rmseV))
```

```
[1] 1.623037
```

## (d)

(5 pts.) Use the `train()` function from the caret package to confirm that your 5-fold CV estimate is correct. The `trControl` argument allows you to specify which observations will be used for training and which will be held out for testing in each iteration of resampling if you do not want to use the default randomly generated folds. Use the code below for the `trControl` argument to obtain a 5-fold CV estimate that uses the same folds as those specified in part (c)

The 5-fold CV estimate from the `train()` is 1.62, which is the 5-fold CV estimate from my code.

```
# setting the seed
set.seed(1234)

# using train() to find the 5-fold CV estimate
(kfcv_lm <- train(x = regData[,1:6], y = regData$y, method = "lm",
                  trControl = trainControl(method = "cv",
                                           index = list(41:200, c(1:40,81:200),
                                                        c(1:80,121:200),
                                                        c(1:120,161:200), 1:160),
                                           indexOut = list(1:40, 41:80, 81:120,
                                                           121:160, 161:200))))
```

```
Linear Regression

200 samples
```

```
  6 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 160, 160, 160, 160, 160
Resampling results:

  RMSE      Rsquared   MAE
  1.623037  0.6368673  1.294747

Tuning parameter 'intercept' was held constant at a value of TRUE
```

## Question Two

(105 pts.) The csv file `mus14data.csv` contains data from a cross sectional survey of 3206 individuals on Medicare (source: Cameron & Trivedi 2010). Below you will construct and evaluate a set of classification models that predict whether or not an individual has private insurance (`private`) or not using the set of other variables shown in the table below as predictors.

Variables from the `mus14data.csv` Dataset

| Variable | Coding |
|---|---|
| `private` | 1 if has private supplemental insurance; 0 otherwise |
| `retire` | 1 if retired; 0 otherwise |
| `age` | age in years |
| `hstatusg` | 1 if good health status; 0 otherwise |
| `hhincome` | annual household income in thousands of dollars |
| `educyear` | years of education |
| `married` | 1 if married; 0 otherwise |
| `hisp` | 1 if Hispanic ethnicity; 0 otherwise |
| `chronic` | number of chronic health conditions |

Below, you will use both the validation set approach and k-fold cross-validation to assess the predictive ability of several different classification methods.

```r
# reading in the data, and factoring all the binary predictors
mus14data <- read.csv("mus14data.csv") %>%
  mutate(retire = factor(retire, levels = c(0,1)),
         hstatusg = factor(hstatusg, levels = c(0,1)),
         married = factor(married, levels = c(0,1)),
         hisp = factor(hisp, levels = c(0,1)))
```

## Part One (Validation Set Approach)

### (a)

(16 pts.) Split the data in to training and validation sets with 70% in the training set and 30% in the test/validation set. Fit a logistic model on the training set and use the estimated model with a threshold of 50% to predict whether or not a subject has private insurance. Take a look at the p-values for the slopes. What do these p-values suggest about the majority of the predictors in the model? Justify your response. Provide an estimate for the test misclassification error. Construct the confusion matrix and compute the true positive rate for classifying an observation as having private insurance and also compute the false positive rate. Use the code below to obtain the row indices for the training and test/validation sets (these ensure a 70/30 split).

- According to the p-values from the logistic model, most of the variables have a significant effect as to whether the individual will have private insurance or not, controlling for all other variables. Age and the number of chronic illnesses an individual has are the only variables that do not have a significant effect on whether or not a person will have private insurance as they are the only variables with p-value larger than 0.05.
- The estimate for the test misclassification error rate is 0.4016649 or 40.17%.
- The true positive rate for classifying as observation as having private insurance is $\frac{93}{93+284} = 0.2467$ or 24.67%. The false positive rate is $\frac{102}{102+482} = 0.1747$ or 17.47%.

```r
# setting the seed
set.seed(1234)
# the code provided to split the data
train_ids <- sample(1:3206, size = ceiling(3206*0.70))
test_ids <- setdiff(1:3206, train_ids)

# splitting the data
q2train <- mus14data[train_ids,]
q2test <- mus14data[test_ids,]

# the logistic regression
summary(q2logit <- glm(private ~ retire + age + hstatusg + hhincome +
                             educyear + married + hisp + chronic,
                   family = binomial(link = "logit"),
                   data = q2train))
```

```
Call:
glm(formula = private ~ retire + age + hstatusg + hhincome +
    educyear + married + hisp + chronic, family = binomial(link = "logit"),
    data = q2train)
```

```
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.3798011  0.8986160  -2.648 0.008090 **
retire1      0.2935871  0.1019006   2.881 0.003963 **
age         -0.0115992  0.0133771  -0.867 0.385891
hstatusg1    0.3420388  0.1179896   2.899 0.003745 **
hhincome     0.0017598  0.0008677   2.028 0.042547 *
educyear     0.1269961  0.0174162   7.292 3.06e-13 ***
married1     0.7052222  0.1150118   6.132 8.69e-10 ***
hisp1       -0.9057101  0.2447861  -3.700 0.000216 ***
chronic      0.0662149  0.0358695   1.846 0.064893 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2992.1  on 2244  degrees of freedom
Residual deviance: 2758.8  on 2236  degrees of freedom
AIC: 2776.8

Number of Fisher Scoring iterations: 4
```

```r
# the predicted probabilities of the test set
q2logit_predprob_test <- predict(q2logit,
                                    newdata = q2test,
                                    type = "response")
# a threshold of 0.5 to assign predicted class.
  # <= 0.5, no private insurance (1), > 0.5, has private insurance (0)
q2logit_predclass_test <- 1*I(q2logit_predprob_test > 0.5)

# the test error rate
mean(q2logit_predclass_test != q2test$private)
```

```
[1] 0.4016649
```

```r
# the confusion matrix
confMat <- table(Predicted = q2logit_predclass_test, Observed = q2test$private)
confMat[c(2,1), c(2,1)]
```

```
         Observed
Predicted   1   0
        1  93 102
        0 284 482
```

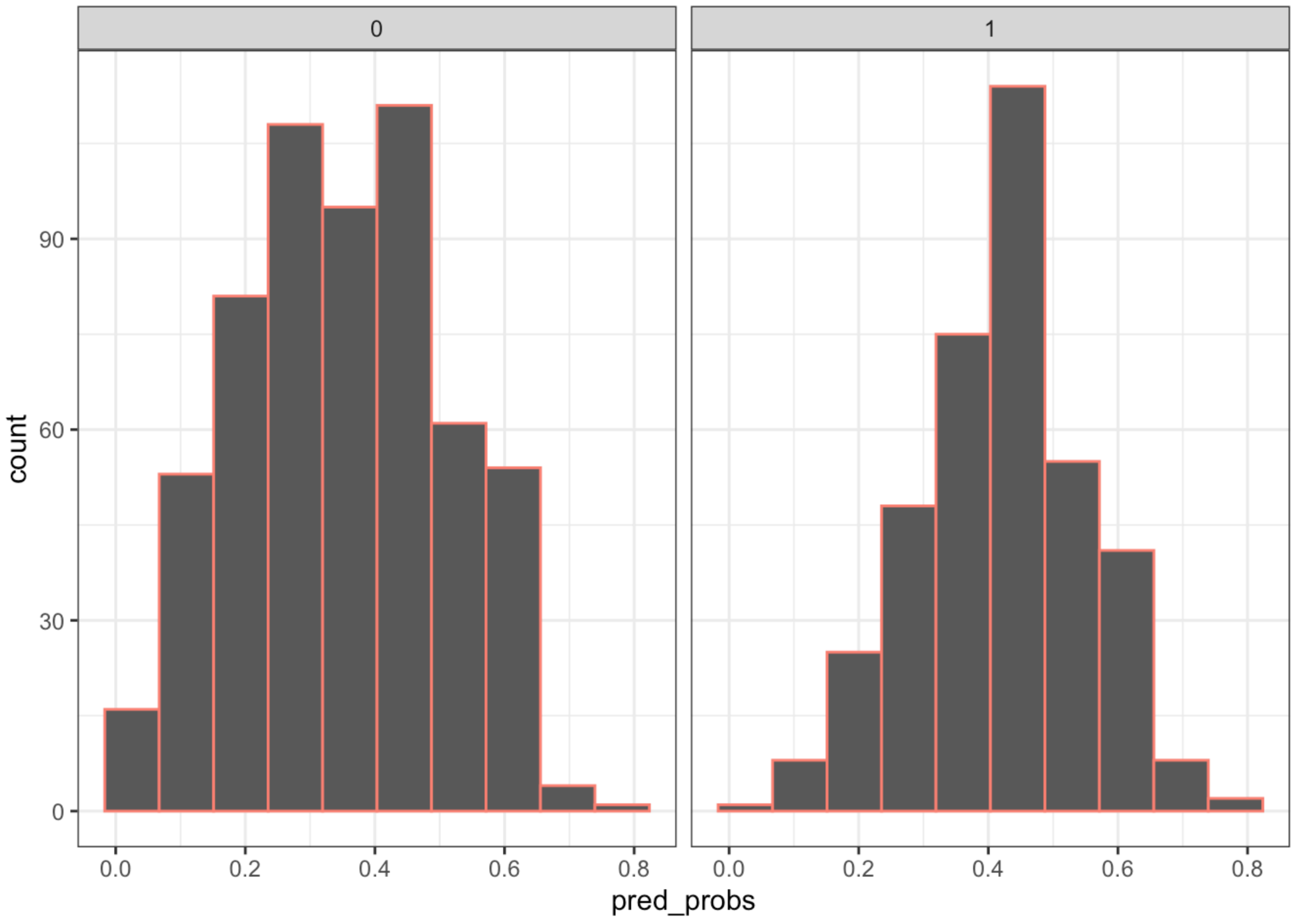
## (b)

(15 pts.) Using the validation data from part (a), provide histograms of the predicted probabilities in each outcome class, construct a calibration plot (bin the predicted probabilities into bins of length 0.10), and compute Cohen's $\kappa$. Collectively, do these diagnostics suggest that the classifier that you fit is part (a) performs well or performs poorly?

- Based on the calibration plot, the model seems to perform well. The points corresponding to the observed event percentage and midpoints of the bins seem to line up decently to the gray dotted line with a slope of one, which is what we want. However, based on the histograms and Cohen's $\kappa$, the classifier that we fit in part (a) performs poorly. If the model performed well, the histograms would be more heavily skewed toward either zero or one, depending on which classification the histogram refers to. In histograms of the classifier from part (a) appear to be almost normal, which is not what we would expect if the classifier performed well. In addition, Cohen's $\kappa = 0.079$, which is indicative that there is little agreement between the observed and predicted classes. This means that that model is a poor classifier of the data.

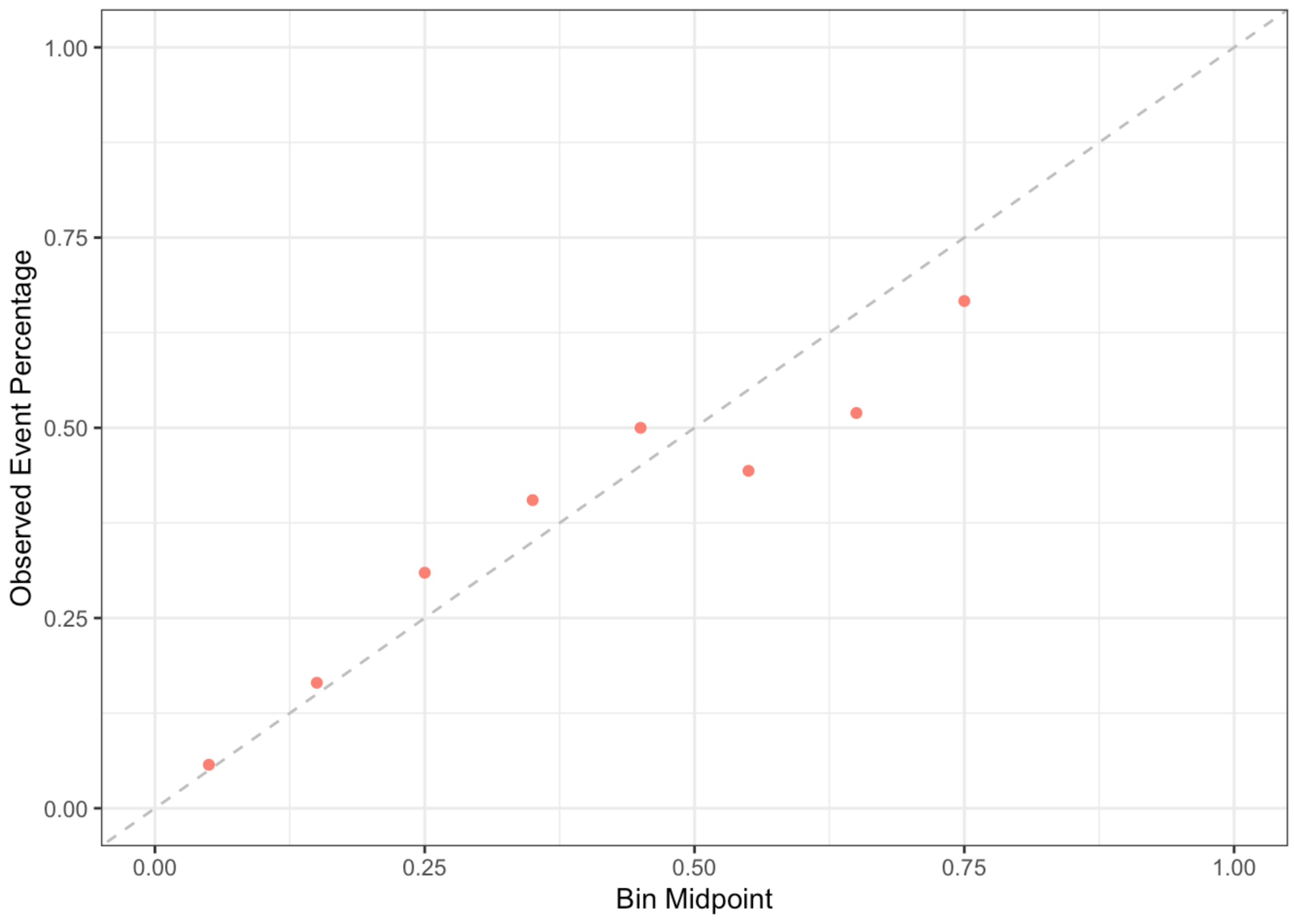
```r
# setting the seed
set.seed(1234)

# creating a data frame of the predicted probabilities and their class
q2pred_data <- data.frame(pred_probs = q2logit_predprob_test,
                          obs_class = q2test$private)
# the histogram
q2pred_data %>%
  ggplot(aes(x = pred_probs)) +
  geom_histogram(bins = 10, col = "salmon")+
  facet_grid(~ obs_class) + theme_bw()
```



```
# the calibration plot
  # binning the predictions
q2pred_data$pred_bin <- cut(q2logit_predprob_test,
                            breaks = c(0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8),
                            include.lowest = TRUE)
  # computing event probability in each bin
q2cal_data <- q2pred_data %>%
  group_by(pred_bin) %>%
  summarise(event_prob = mean(obs_class))

  # adding bin midpoints for plotting
q2cal_data$bin_mid <- seq(0.05, 0.75, by = 0.1)
  # the calibration plot
q2cal_data %>%
  ggplot(aes(x = bin_mid, y = event_prob)) +
  geom_point(col = "salmon")+
  geom_abline(slope = 1, intercept = 0, lty = 2, col = "grey") +
  ylab("Observed Event Percentage") + ylim(0,1) +
  xlab("Bin Midpoint") + xlim(0,1) + theme_bw()
```



```
# Cohen's kappa
cohen.kappa(cbind(q2test$private, q2logit_predclass_test))
```

```
Call: cohen.kappa1(x = x, w = w, n.obs = n.obs, alpha = alpha, levels = levels,
    w.exp = w.exp)

Cohen Kappa and Weighted Kappa correlation coefficients and confidence boundaries
                 lower estimate upper
unweighted kappa 0.021    0.079  0.14
weighted kappa   0.021    0.079  0.14

 Number of subjects = 961
```

## (c)

(12 pts.) Suppose that you want to determine if there is a better threshold than 50%. Construct an ROC curve and compute the AUC (based on the test/validation data) for the logistic model that you fit in part (a). What is the optimal threshold if we want the sum of the sensitivity and specificity to be maximized? Does the optimal threshold produce a good predictive model based on sensitivity and specificity? Justify your response.

- The AUC of the logistic model fit in part (a) is 0.6324. The threshold that maximizes the sum of sensitivity and specificity is 0.375. The test misclassification error rate with the new threshold is still 0.4016649 or 40.17%. The optimal threshold does not produce a good predictive model based on sensitivity and specificity. A specificity of 0.541 is barely better than chance. The sensitivity of 0.690 is better, but still not great.

```r
# setting the seed
set.seed(1234)
# creating the ROC curve
(q2logit_roc <- roc(response = q2pred_data$obs_class,
                   predictor = q2pred_data$pred_probs))
```

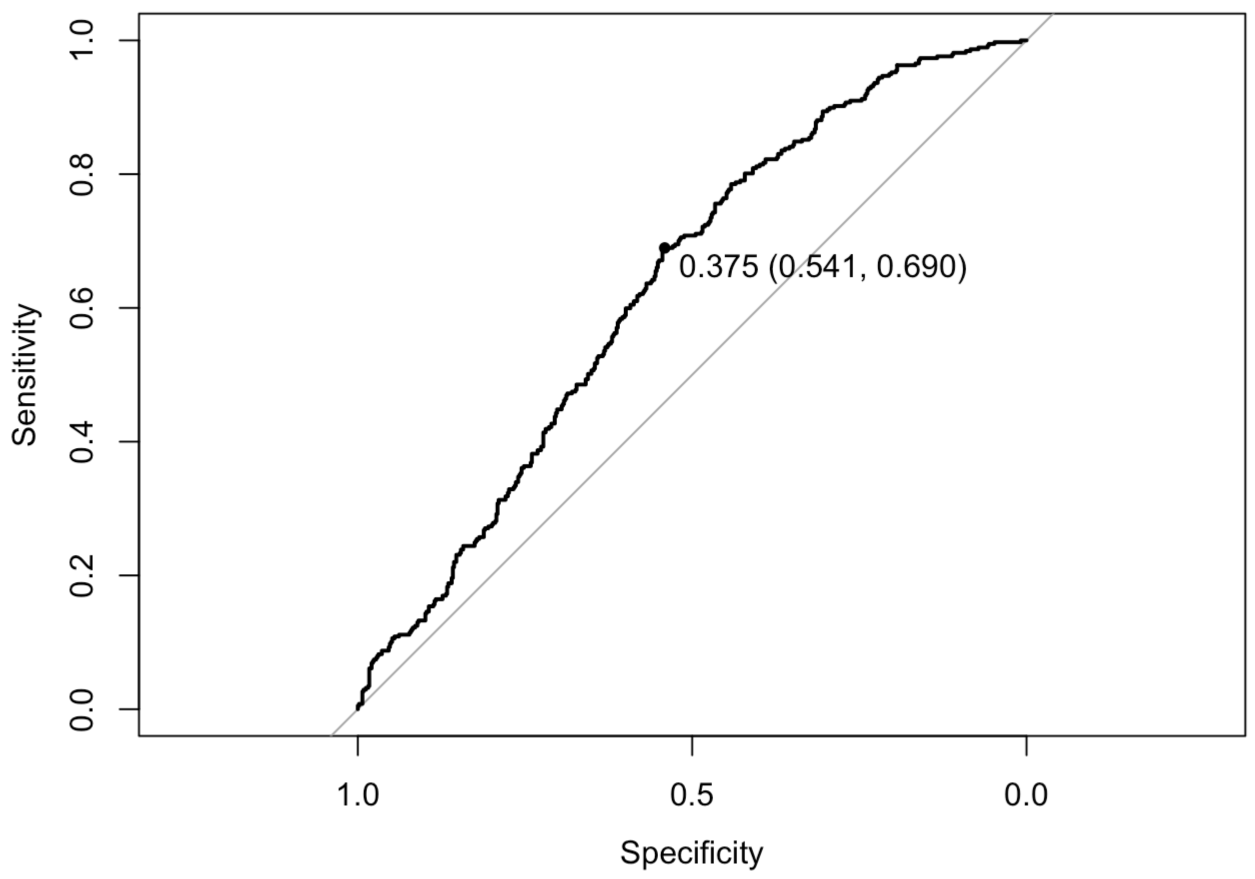
```
Setting levels: control = 0, case = 1

Setting direction: controls < cases


Call:
roc.default(response = q2pred_data$obs_class, predictor = q2pred_data$pred_probs)

Data: q2pred_data$pred_probs in 584 controls (q2pred_data$obs_class 0) < 377 cases
(q2pred_data$obs_class 1).
Area under the curve: 0.6324
```

```r
# plotting the ROC curve
plot(q2logit_roc, print.thres = TRUE)
```



```r
# a threshold of 0.375 to assign predicted class.
  # <= 0.375, no private insurance (1), > 0.375, has private insurance (0)
q2logit_predclass_test375 <- 1*I(q2logit_predprob_test > 0.375)

# the test error rate
mean(q2logit_predclass_test375 != q2test$private)
```

```
[1] 0.4016649
```

## (d)

(10 pts.) Fit KNN classifiers on the training data set for $K = 5, 10, 15, 20, 25, 30, 35,$ and 40 using all 8 predictors. Use the validation set to select the optimal value of $K$ based on misclassification error. How does the best-performing KNN model compare the the logistic model from part (a) with respect to misclassification error?

- The knn classifier with the lowest test misclassification error was $K = 10$, which was 0.3798. This classifier is incorrect 37.98% of the time, which is better than the logistic model from part (a) which had a test misclassification error rate of 40.17%.

```r
# setting the seed
set.seed(1234)
```

```
# the knn classifiers
knn_5 <- knn(train = q2train[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
              test = q2test[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
             cl = q2train$private, k = 5, prob = TRUE)
knn_10 <- knn(train = q2train[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
              test = q2test[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
             cl = q2train$private, k = 10, prob = TRUE)
knn_15 <- knn(train = q2train[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
              test = q2test[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
             cl = q2train$private, k = 15, prob = TRUE)
knn_20 <- knn(train = q2train[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
              test = q2test[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
             cl = q2train$private, k = 20, prob = TRUE)
knn_25 <- knn(train = q2train[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
              test = q2test[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
             cl = q2train$private, k = 25, prob = TRUE)
knn_30 <- knn(train = q2train[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
              test = q2test[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
             cl = q2train$private, k = 30, prob = TRUE)
knn_35 <- knn(train = q2train[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
              test = q2test[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
             cl = q2train$private, k = 35, prob = TRUE)
knn_40 <- knn(train = q2train[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
              test = q2test[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
             cl = q2train$private, k = 40, prob = TRUE)

# the misclassification test error rates
mean(knn_5 != q2test$private)
```

```
[1] 0.3891779
```

```
mean(knn_10 != q2test$private)
```

```
[1] 0.3798127
```

```
mean(knn_15 != q2test$private)
```

```
[1] 0.3829344
```

```
mean(knn_20 != q2test$private)
```

```
[1] 0.3995838
```

```
mean(knn_25 != q2test$private)
```

```
[1] 0.3985432
```

```
mean(knn_30 != q2test$private)
```

```
[1] 0.3850156
```

```
mean(knn_35 != q2test$private)
```

```
[1] 0.4016649
```

```
mean(knn_40 != q2test$private)
```

```
[1] 0.4068678
```

## Part Two (Cross-Validation Approach)

For any question below that requires use of the `train()` function, set the seed to 1234 ( `set.seed(1234)` ) before running the `train()` function.

### (e)

(10 pts.) Using the complete data set with all 3206 observations, compute the 10-fold CV misclassification error for the logistic classifier based on all 8 predictors and using a threshold of 50%. How does this compare with the estimate for the test misclassification error that you obtained in part (a)? On average, should we expect this value to be larger or smaller than the misclassification error obtained in part (a)? Justify your response.

- The accuracy of the 10-fold CV is 0.6220, so the misclassification error for the logistic classifier based on all 8 predictors and using a threshold of $50\%$ is $1 - 0.6220 = 0.3780$ This classifier is incorrect $37.80\%$ of the time, which is better than the logistic model from part (a) which had a test misclassification error rate of $40.17\%$. This implies that the 10-fold CV is at least slightly better than the predictive model from part (a). Another way to determine whether or not the 10-fold CV model performed better than the model from part (a) is via Cohen's $\kappa$. Cohen's $\kappa = 0.079$ for the model in part (a) while it is 0.13 for this predictive model. Since Cohen's $\kappa$ for this model is bigger, this is the better model. However, it still might not be considered a good model.

- You can’t really say whether or not the misclassification error from the 10-fold CV model will be larger or smaller on average compared to the validation set approach. What can be said is that the variance of the estimate will be smaller for the 10-fold CV model than for the validation set approach.

```r
# setting the seed
set.seed(1234)
# factors private outcome variable
mus14data1 <- mus14data %>%
  mutate(private = factor(private, levels = c(0,1), labels = c("No", "Yes")))
# the 10-fold CV with a threshold of 0.50
(q2_10fcv <- train(x = mus14data[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
                  y = mus14data1$private,
                  method = "glm", family = binomial(link = "logit"),
                  trControl = trainControl(method = "cv", number = 10,
                                          classProbs = TRUE,
                                          savePredictions = TRUE)))
```

```
Generalized Linear Model 

3206 samples
   8 predictor
   2 classes: 'No', 'Yes' 

No pre-processing
Resampling: Cross-Validated (10 fold) 
Summary of sample sizes: 2885, 2886, 2884, 2885, 2885, 2886, ... 
Resampling results:

  Accuracy  Kappa    
  0.621957  0.1273671
```

## (f)

(15 pts.) Using the complete data set with all 3206 observations, determine the optimal threshold for the logistic model fit in part (e) based on 10-fold CV misclassification error. Consider threshold values of 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90%. Is the optimal threshold based on 10-fold CV misclassification error the same as the optimal threshold based on 10-fold CV Cohen’s $\kappa$? Justify your response.

- The optimal threshold based on the 10-fold CV predictive model depends which statistic your are optimize the model with respect to. The optimal threshold based on misclassification error is not the same as the optimal threshold based on Cohen’s $\kappa$. The optimal threshold based on the 10-fold CV misclassification error is 40%, which has an misclassification error rate of 0.3778. The optimal threshold based on the 10-fold CV Cohen’s $\kappa$ is 60%, which has $\kappa = 0.2329$.

```r
# setting the seed
set.seed(1234)
# the sequence of threshold values
prob_thresh <- seq(0.1, 0.9, by = 0.1)
```

```
# use thresholder() function to obtain accuracy measures at each threshold
(q2_10fcv_thresh <- thresholder(q2_10fcv,
                               threshold = prob_thresh,
                               final = TRUE,
                               statistics = c("Sensitivity", "Specificity",
                                            "Accuracy", "Kappa")))
```

```
  parameter prob_threshold Sensitivity  Specificity  Accuracy         Kappa
1      none            0.1  0.99898477 0.0008064516 0.6126015 -0.0002557865
2      none            0.2  0.99796954 0.0040322581 0.6132265  0.0024430841
3      none            0.3  0.99593650 0.0120838710 0.6150947  0.0097731869
4      none            0.4  0.95472651 0.0958838710 0.6222714  0.0596776668
5      none            0.5  0.84073345 0.2755483871 0.6219570  0.1273671169
6      none            0.6  0.59853672 0.6462580645 0.6169930  0.2328738259
7      none            0.7  0.36594064 0.8573677419 0.5561537  0.1928644578
8      none            0.8  0.14912462 0.9734258065 0.4681931  0.0986685503
9      none            0.9  0.04579923 0.9983870968 0.4145355  0.0346235504
```

```
# finds the best model wrt to accuracy and kappa
max(q2_10fcv_thresh$Accuracy)
```

```
[1] 0.6222714
```

```
max(q2_10fcv_thresh$Kappa)
```

```
[1] 0.2328738
```

## (g)

(10 pts.) Using the complete data set with all 3206 observations, fit KNN classifiers with $K = 5, 10, 15, 20, 25, 30, 35,$ and 40 using all 8 predictors. What is the optimal value of K based on 10-fold CV misclassification error? Justify your response.

- The optimal value of K nearest neighbors based on 10-fold CV misclassification error is $K = 35$. The accuracy of the models is given and the misclassification error is found by subtracting the accuracy from one. Thus, the smallest misclassification error is $1 - 0.6304 = 0.3696$, which occurs when $K = 35$.

```
# setting the seed
set.seed(1234)
# the tune grid object with the K neighbor values
tg_01 <- data.frame(k = seq(5,40, by = 5))
# the 10-fold CV measures for each model
(q2_10fcv_knn <- train(x = mus14data[c("retire", "age", "hstatusg", "hhincome",
                                    "educyear", "married", "hisp", "chronic")],
                      y = mus14data1$private,
                      method = "knn",
                      trControl = trainControl(method = "cv", number = 10),
                      tuneGrid = tg_01))
```

```
k-Nearest Neighbors

3206 samples
   8 predictor
   2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2885, 2886, 2884, 2885, 2885, 2886, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.6207264  0.1957841
  10  0.6138670  0.1758760
  15  0.6210060  0.1906329
  20  0.6247744  0.1922348
  25  0.6300636  0.2020231
  30  0.6263291  0.1930542
  35  0.6303897  0.2015085
  40  0.6266494  0.1917884

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 35.
```

## (h)

(10 pts.) Using the complete data set with all 3206 observations, fit LDA and QDA classifiers using all 8 predictors. Compute the 10-fold CV misclassification error for each classifier. For these data, why might one argue that that LDA and QDA methods are inappropriate?

- The 10-fold CV misclassification error for the LDA classifier is $0.3793$ and the 10-fold CV misclassification error for the QDA classifier is $0.4033$. The classifiers incorrectly classify the data $37.93\%$ and $40.33\%$ of the time, respectively. The LDA and QDA classifiers might be inappropriate due to the fact that some of the predictors are categorical. As they are not normal, they're not appropriate for LDA and QDA methods.

```r
# setting the seed
set.seed(1234)
# the LDA classifier
(q2lda <- train(private ~ retire + age + hstatusg + hhincome +
                 educyear + married + hisp + chronic,
               data = mus14data1,
               method = "lda",
               trControl = trainControl(method = "cv", number = 10)))
```

```
Linear Discriminant Analysis

3206 samples
   8 predictor
   2 classes: 'No', 'Yes'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2885, 2886, 2884, 2885, 2885, 2886, ...
Resampling results:

  Accuracy   Kappa
  0.6207138  0.1210944
```

```
# the QDA classifier
(q2qda <- train(private ~ retire + age + hstatusg + hhincome +
                 educyear + married + hisp + chronic,
               data = mus14data1,
               method = "qda",
               trControl = trainControl(method = "cv", number = 10)))
```

```
Quadratic Discriminant Analysis

3206 samples
   8 predictor
   2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2885, 2886, 2885, 2885, 2886, 2885, ...
Resampling results:

  Accuracy   Kappa
  0.5966978  0.2128437
```

## (i)

(5 pts.) Using the complete data set with all 3206 observations, fit a naive Bayes classifier (using all 8 predictors). Assume that the numerical variables are normally distributed. Use `method = "naive_bayes"` and `tuneGrid = data.frame(laplace = 0, usekernel = FALSE, adjust = FALSE)` in the `train()` function. Also, if you have not already, be sure to convert the binary 0/1 predictors to factor variables, otherwise these predictors will be treated as numerical and therefore will be assumed to be normally distributed. Compute the 10-fold CV misclassification error for this classifier.

- The 10-fold CV misclassification error for the naive Bayes classifier is 0.4. Thus, the classifier gets 40% of its predictions incorrect.

```
# setting the seed
set.seed(1234)
# the naive Bayes classifier
(q2bayes <- train(private ~ retire + age + hstatusg + hhincome +
                             educyear + married + hisp + chronic,
                 data = mus14data1,
                 method = "naive_bayes",
                 trControl = trainControl(method = "cv", number = 10),
                 tuneGrid = data.frame(laplace = 0, usekernel = FALSE,
                                       adjust = FALSE)))
```

```
Naive Bayes 

3206 samples
   8 predictor
   2 classes: 'No', 'Yes' 

No pre-processing
Resampling: Cross-Validated (10 fold) 
Summary of sample sizes: 2885, 2886, 2884, 2885, 2885, 2886, ... 
Resampling results:

  Accuracy   Kappa    
  0.6004645  0.2116564

Tuning parameter 'laplace' was held constant at a value of 0
Tuning
 parameter 'usekernel' was held constant at a value of FALSE
Tuning
 parameter 'adjust' was held constant at a value of FALSE
```

## (j)

(2 pts.) Considering all of the classifiers from parts (e) - (h), which is best with respect to 10-fold CV misclassification error?

- From parts (e) - (h), the best classifier is the KNN classifier where $K = 35$ with a threshold of 0.5. This is because that classifier has the lowest 10-fold CV misclassification error, which is 0.3696.