Stats 101C Homework 3

Charles Liu (304804942)

10/27/2020

Loading Necessary Packages:

```
library(MASS)
library(ISLR)
library(class)
library(boot)
library(mclust)
library(caret)
library(e1071)
library(MLeval)
```

Problem 1 (Exercise 4.7.5) We now examine the differences between LDA and QDA.

(a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

ANSWER: If the Bayes decision boundary is linear, we expect QDA to perform better on the *training* set because of its higher flexiblity. This can give us a closer fit. We expect LDA to perform better on the *test* set because the QDA could cause it to be overfitted.

(b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

ANSWER: If the Bayes decision bounary is non-linear, we expect QDA to perform better both on the *training* and *test* sets because QDA offers more flexibility compared to the LDA. This will cause our model to be closer fit to the data.

(c) In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

ANSWER: The QDA is recommended as the (n) sample size increases. Since the *training* set is very large, we can expect that the variance won't cause us too much problems, even though it has a higher variance than LDA.

(d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.

ANSWER: False because if we have fewer sample points, the QDA model might lead us to overfit due to the higher flexibility, which in turn causes a higher variance to occur. This will give us a bad *test* error rate. We would prefer to use LDA for more linear types of decision boundaries, and we would use QDA for more non-linear types of decision boundaries.

Problem 2 (Exercise 4.7.13) (Models should be compared using 5-fold cross validation) Using the Boston data set, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN models using various subsets of the predictors. Describe your findings.

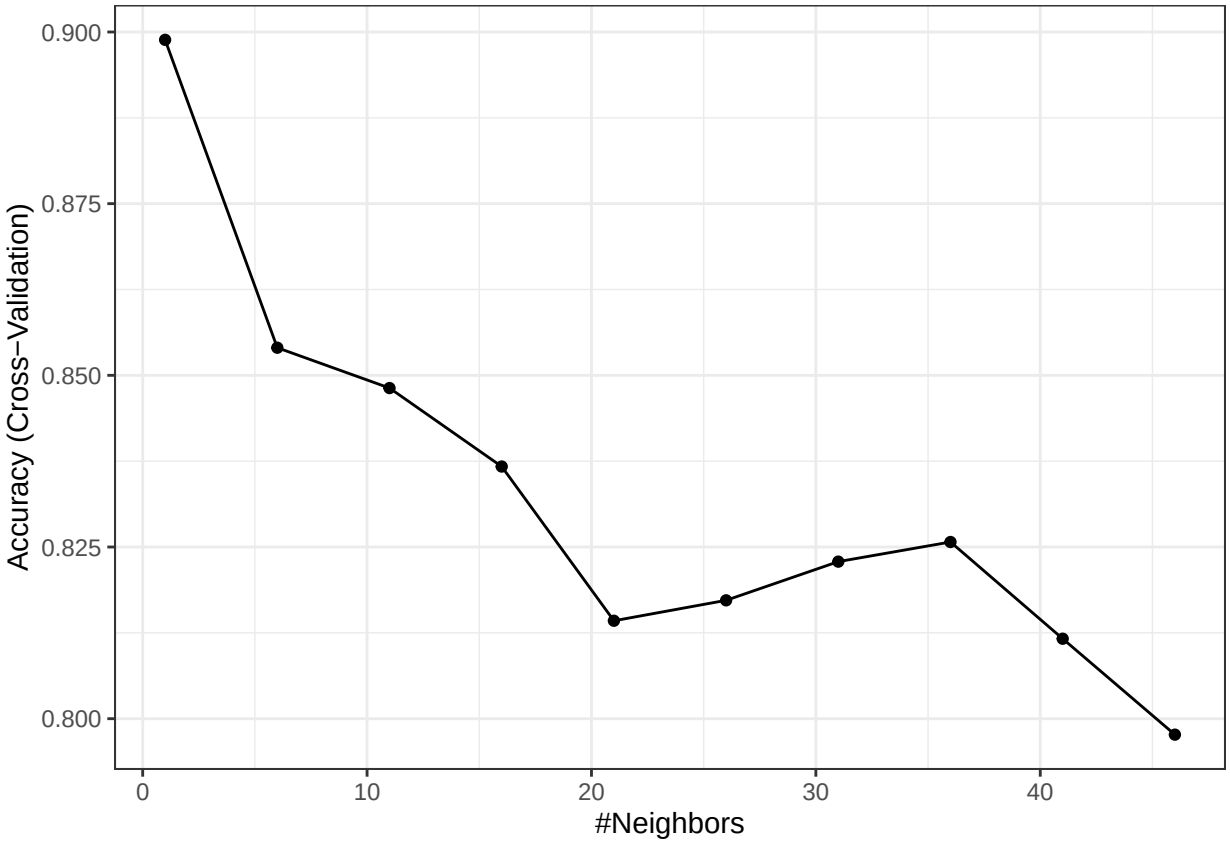
Set up our parameters, tests, and trainings data

```
# Attach Boston data and create our crim01 variable
data(Boston)
Boston$crim01 <- rep(0, length(Boston$crim))
# if the median(cri) < crim, then it is above
Boston$crim01 <- ifelse(as.integer(Boston$crim > median(Boston$crim)),
                        "Above", "Below")
Boston$crim01 <- factor(Boston$crim01, levels=c("Below", "Above"))

# Attach our crim01 with our Boston dataset &
Boston <- data.frame(Boston, Boston$crim01)

# Create data partition into training and test data.
train <- createDataPartition(Boston$crim01, p = 0.7,
                                    list = FALSE)
Boston_train <- Boston[train, ]
Boston_test <- Boston[-train, ]

# Here, we specify how we will evaluate our models
train_control <- trainControl(method="cv", number = 5,
                              classProbs = TRUE,
                              savePredictions = TRUE)
```

Fitting our Models

```
# Fit different classification models.
KNNfit <- train(crim01 ~ zn + indus + chas + nox + rm + age + dis +
                 rad + tax + ptratio + black + lstat + medv,
                data = Boston_train, method = 'knn',
                preProc = c("center", "scale"),
```

```
                trControl = train_control,
                tuneGrid = expand.grid(k = seq(1, 50, by = 5)))
ggplot(KNNfit) + theme_bw()
```




```
LRfit <- train(crim01 ~ zn + indus + chas + nox + rm + age + dis +
                 rad + tax + ptratio + black + lstat + medv,
               data = Boston_train, method = "glm",
               family = "binomial",
               preProc = c("center", "scale"),
               trControl = train_control)

LDAfit <- train(crim01 ~ zn + indus + chas + nox + rm + age + dis +
                 rad + tax + ptratio + black + lstat + medv,
                data = Boston_train, method = "lda",
                preProc = c("center", "scale"),
                trControl = train_control)

QDAfit <- train(crim01 ~ zn + indus + chas + nox + rm + age + dis +
                 rad + tax + ptratio + black + lstat + medv,
                data = Boston_train, method = "qda",
                preProc = c("center", "scale"),
                trControl = train_control)


# Evaluate different models using K-fold Cross-Validation
```

```
KNNfit
```

```
## k-Nearest Neighbors
##
## 356 samples
##  13 predictor
##   2 classes: 'Below', 'Above'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 284, 285, 285, 284, 286
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.8988576  0.7976469
##    6  0.8540208  0.7078299
##   11  0.8481478  0.6961960
##   16  0.8367203  0.6734503
##   21  0.8142622  0.6284798
##   26  0.8172379  0.6343179
##   31  0.8228728  0.6455878
##   36  0.8257311  0.6511851
##   41  0.8116454  0.6230840
##   46  0.7976783  0.5949535
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```
LRfit
```

```
## Generalized Linear Model
##
## 356 samples
##  13 predictor
##   2 classes: 'Below', 'Above'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 285, 285, 284, 285, 285
## Resampling results:
##
##   Accuracy   Kappa
##   0.8932316  0.7864395
```

```
LDAfit
```

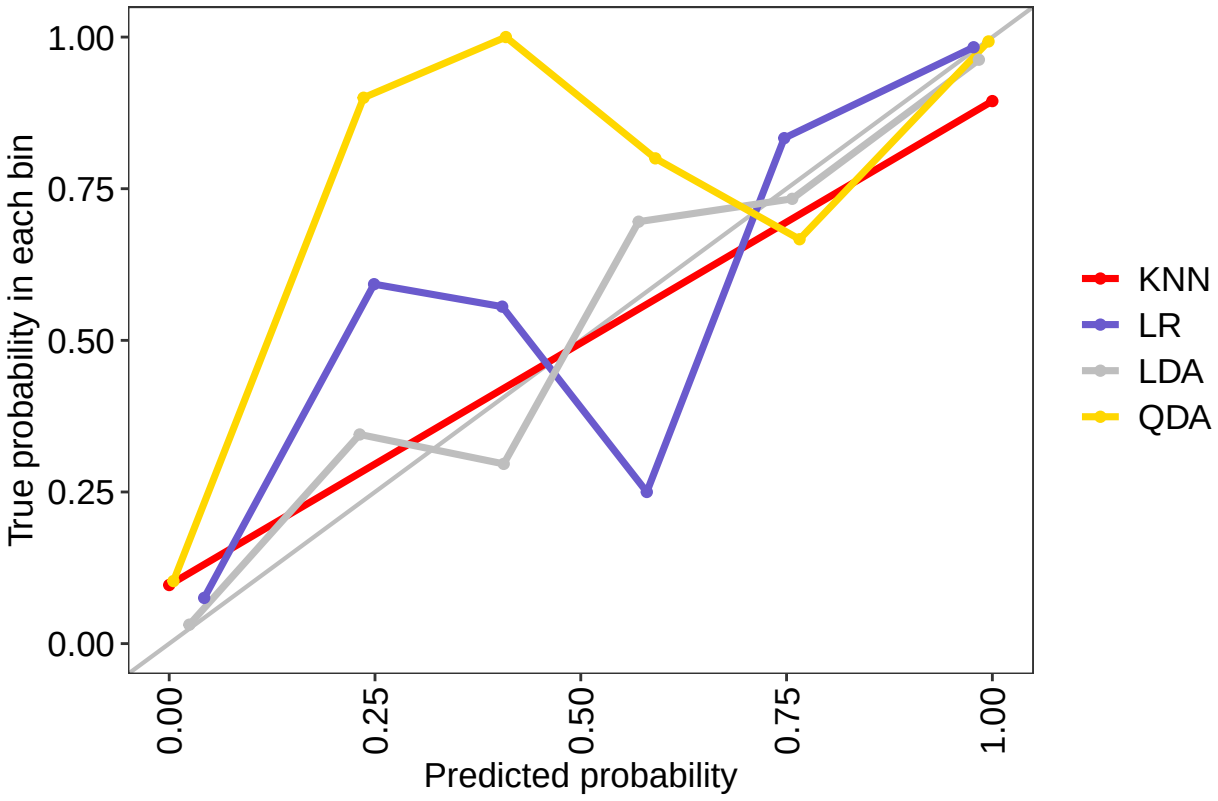
```
## Linear Discriminant Analysis
##
## 356 samples
##  13 predictor
##   2 classes: 'Below', 'Above'
##
```

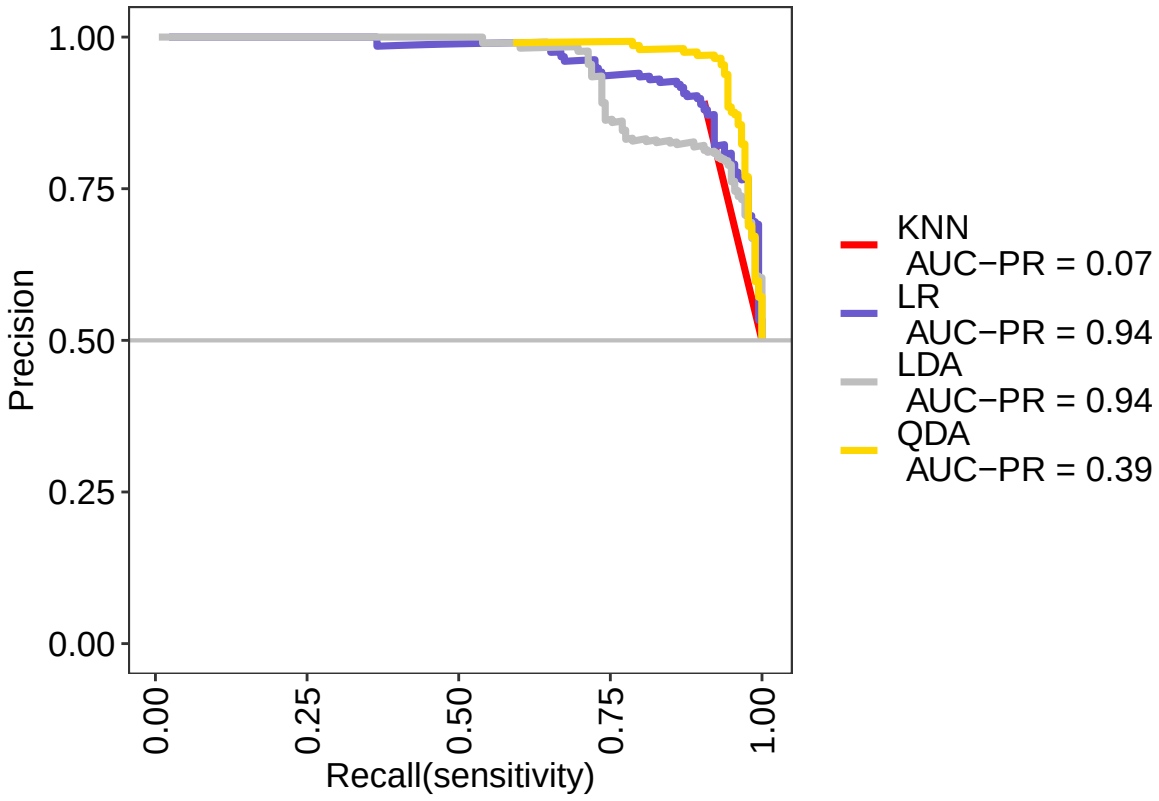
```
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 286, 284, 284, 285, 285
## Resampling results:
##
##   Accuracy   Kappa
##   0.8233065  0.6465795
```

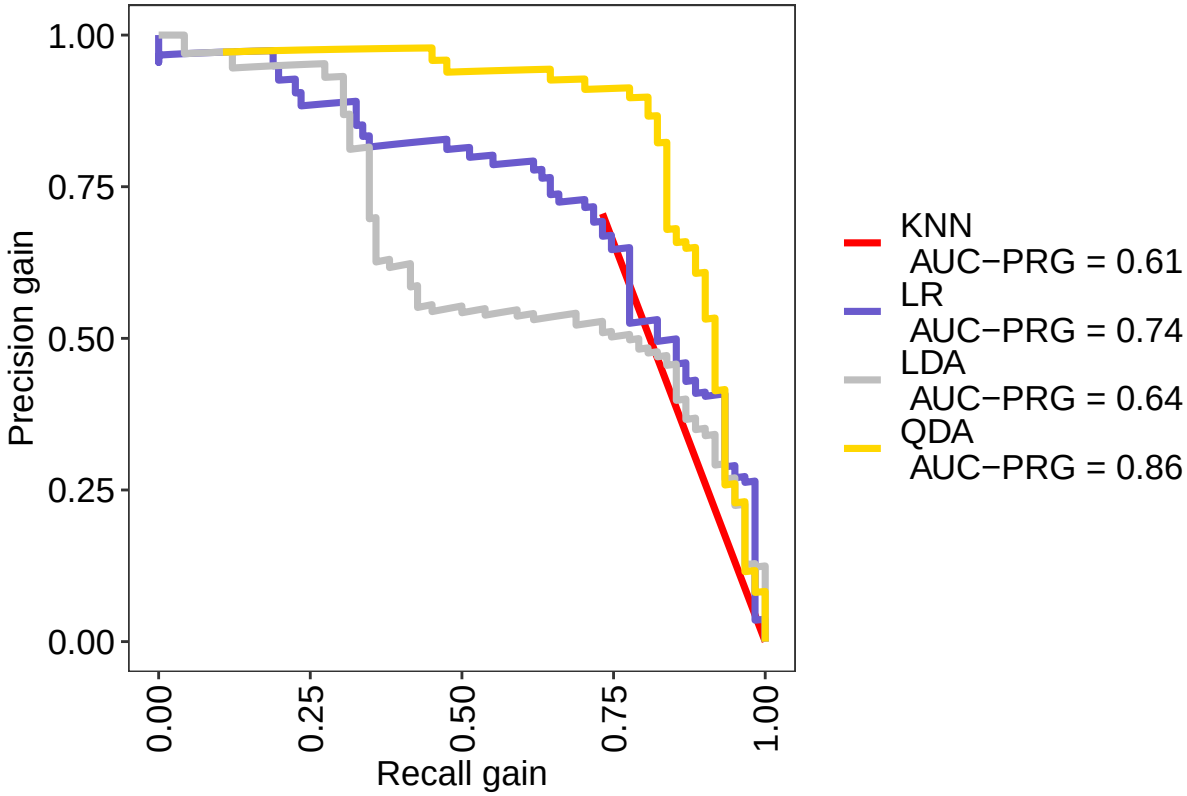
```
QDAfit
```

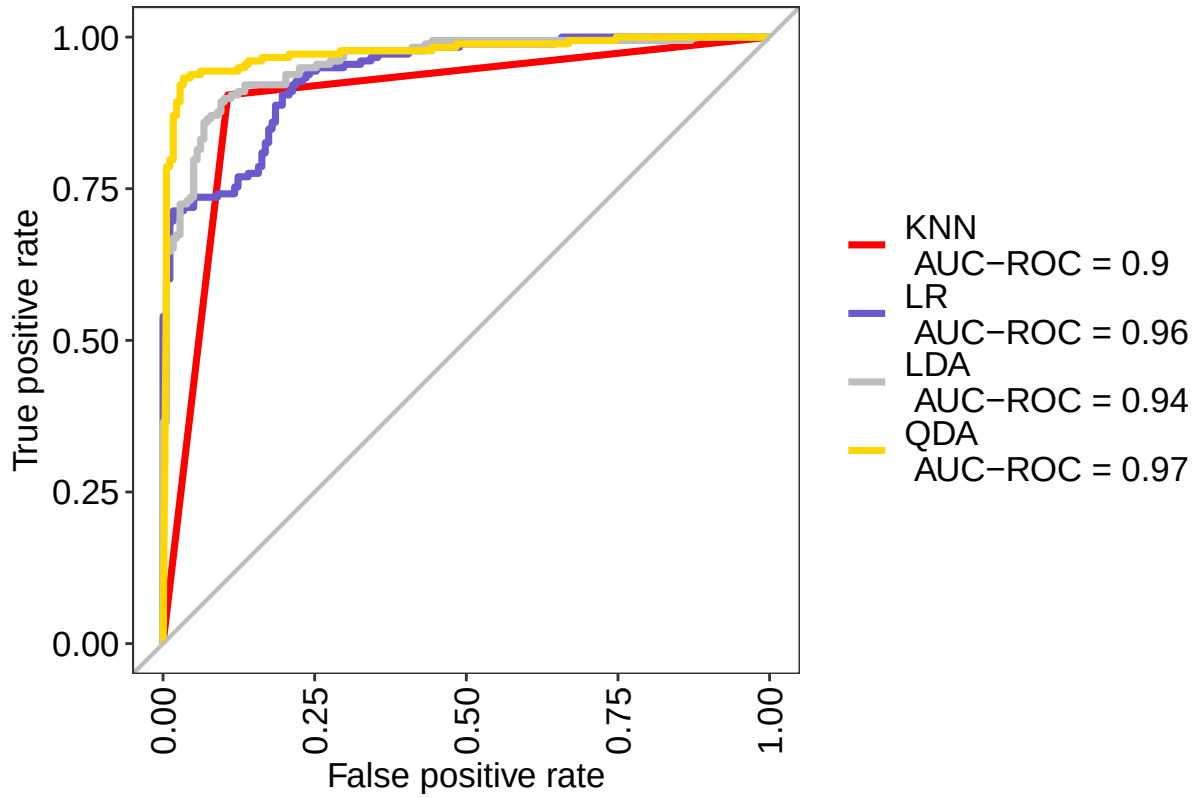
```
## Quadratic Discriminant Analysis
##
## 356 samples
##  13 predictor
##   2 classes: 'Below', 'Above'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 284, 286, 286, 284, 284
## Resampling results:
##
##   Accuracy   Kappa
##   0.8960317  0.7920635
```

```
# Evaluate using ROC curves.
res <- evalm(list(KNNfit,LRfit,LDAfit,QDAfit),gnames=c('KNN','LR',
                                                       'LDA', 'QDA'))
```


1.00
0.75
0.50
0.25
0.00
True probability in each bin
0.00
0.25
0.50
0.75
1.00
Predicted probability
KNN
LR
LDA
QDA


1.00
0.75
0.50
0.25
0.00
Precision
0.00
0.25
0.50
0.75
1.00
Recall(sensitivity)
KNN
AUC−PR = 0.07
LR
AUC−PR = 0.94
LDA
AUC−PR = 0.94
QDA
AUC−PR = 0.39


1.00
0.75
0.50
0.25
0.00
Precision gain
0.00
0.25
0.50
0.75
1.00
Recall gain
KNN
AUC−PRG = 0.61
LR
AUC−PRG = 0.74
LDA
AUC−PRG = 0.64
QDA
AUC−PRG = 0.86



Find our predictions & misclassifications

```
# Create a function for misclassifications accuracy
misclassfication_model <- function(prediction, observation, n) {
  misclass_risk <- sum((as.integer(prediction) -
                          as.integer(observation))^2)/nrow(n)
  return(misclass_risk)
}

# KNN prediction
predKNN <- predict(KNNfit, newdata = Boston_test)
CM_KNN <- confusionMatrix(data = predKNN, reference = Boston_test$crim01)
misclass_KNN <- misclassfication_model(prediction = KNNfit$pred$pred,
                        observation = KNNfit$pred$obs,
                        n = Boston_train)

# Logistic Regression prediction
predLR <- predict(LRfit, newdata = Boston_test)
CM_LR <- confusionMatrix(data = predLR, reference = Boston_test$crim01)
misclass_LR <- misclassfication_model(prediction = LRfit$pred$pred,
                        observation = LRfit$pred$obs,
                        n = Boston_train)

# LDA prediction
```

```
predLDA <- predict(LDAfit, newdata = Boston_test)
CM_LDA <- confusionMatrix(data = predLDA, reference = Boston_test$crim01)
misclass_LDA <- misclassfication_model(prediction = LDAfit$pred$pred,
                       observation = LDAfit$pred$obs,
                       n = Boston_train)

# QDA prediction
predQDA <- predict(QDAfit, newdata = Boston_test)
CM_QDA <- confusionMatrix(data = predQDA, reference = Boston_test$crim01)
misclass_QDA <- misclassfication_model(prediction = QDAfit$pred$pred,
                       observation = QDAfit$pred$obs,
                       n = Boston_train)
```

Create matrix with all the models to compare

```
# Results of our models & their accuracy's (average)
KNNresults <- KNNfit$results[1,-c(1,3:5)]
LRresults <- LRfit$results[,-c(1,3:5)]
LDANresults <- LDAfit$results[,-c(1,3:5)]
QDAresults <- QDAfit$results[,-c(1,3:5)]

results_comparison <- matrix(c(KNNresults, LRresults,
                               LDANresults, QDAresults))
colnames(results_comparison) <- "Average Accuracy of Model"
rownames(results_comparison) <- c("KNN", "LR", "LDA", "QDA")
results_comparison
```

```
##     Average Accuracy of Model
## KNN                 0.8988576
## LR                  0.8932316
## LDA                 0.8233065
## QDA                 0.8960317
```

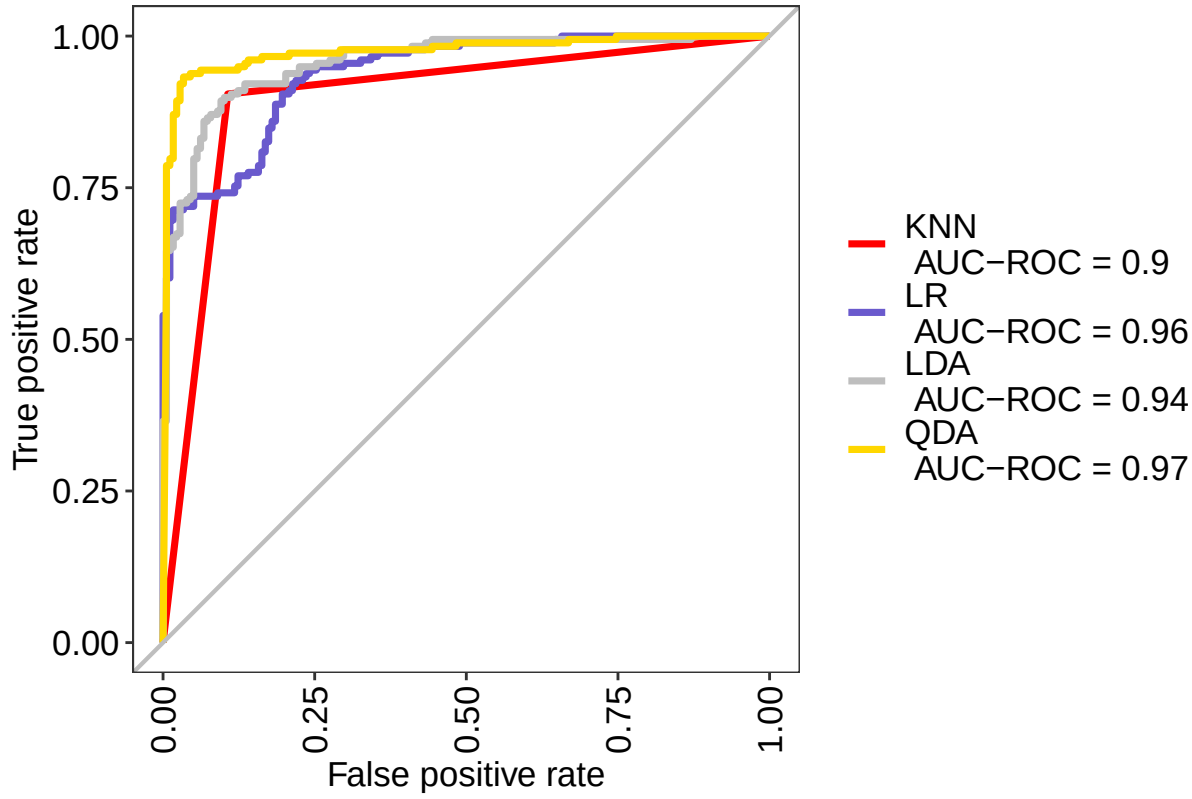
```
# Test accuracy
KNNtest_results <- CM_KNN$overall[-c(2:7)]
LRtest_results <- CM_LR$overall[-c(2:7)]
LDAtest_results <- CM_LDA$overall[-c(2:7)]
QDAtest_results <- CM_QDA$overall[-c(2:7)]
test_comparison <- matrix(c(KNNtest_results, LRtest_results,
                            LDAtest_results, QDAtest_results))
colnames(test_comparison) <- "Test Accuracy of Model"
rownames(test_comparison) <- c("KNN", "LR", "LDA", "QDA")
test_comparison
```

```
##     Test Accuracy of Model
## KNN              0.9200000
## LR               0.9266667
## LDA              0.8533333
## QDA              0.9266667
```

```r
# Misclassification Risk
misclass_risk_comparison <- matrix(c(misclass_KNN,
                                     misclass_LR, misclass_LDA, misclass_QDA))
colnames(misclass_risk_comparison) <- "Misclassification Risk of Model"
rownames(misclass_risk_comparison) <- c("KNN", "LR", "LDA", "QDA")
misclass_risk_comparison
```

```
##     Misclassification Risk of Model
## KNN                       1.6713483
## LR                        0.1067416
## LDA                       0.1769663
## QDA                       0.1039326
```

```r
res$roc
```



COMMENTS:
i) For the KNN model, there is a 90.147% average accuracy over the 5-folds, a test accuracy of 92.667%, and a misclassification risk of 1.657. The ROC curve the area under the curve of 0.90.

ii) For the LR model, there is a 89.871% average accuracy over the 5-folds, a test accuracy of 91.333%, and a misclassification risk of 0.101. The ROC curve the area under the curve of 0.96.

iii) For the LDA model, there is a 84.873% average accuracy over the 5-folds, a test accuracy of 84.667%, and a misclassification risk of 0.152. The ROC curve the area under the curve of 0.94.

iv) For the QDA model, there is a 87.930% average accuracy over the 5-folds, a test accuracy of 90.667%, and a misclassification risk of 0.121 The ROC curve the area under the curve of 0.97.

v) We see that (from least to greatest) the *average accuracy* over the 5-folds are ranked **LDA, QDA, LR, KNN**. For the *test accuracy* (from least to greatest), we have **LDA, QDA, LR, KNN**. For the *Misclassification Risk* (from least to greatest), we have **LR, QDA, LDA, KNN**. We can conclude our Losigistic Regression model has the best of *Misclassification Risk*, which tells us this model is best for classifying. As for both *test & average accuracy*, we see that KNN offers the best results for it, however one thing to note is that there is more variation in this model. The reason for this is because we can see simply that our *test accuracy* is slightly higher than our *average accuracy*. As a follow up to that, it was mentioned that our best usage of our K is when $K = 1$.

Problem 3 (Exercise 5.4.8) We will now perform cross-validation on a simulated data set.

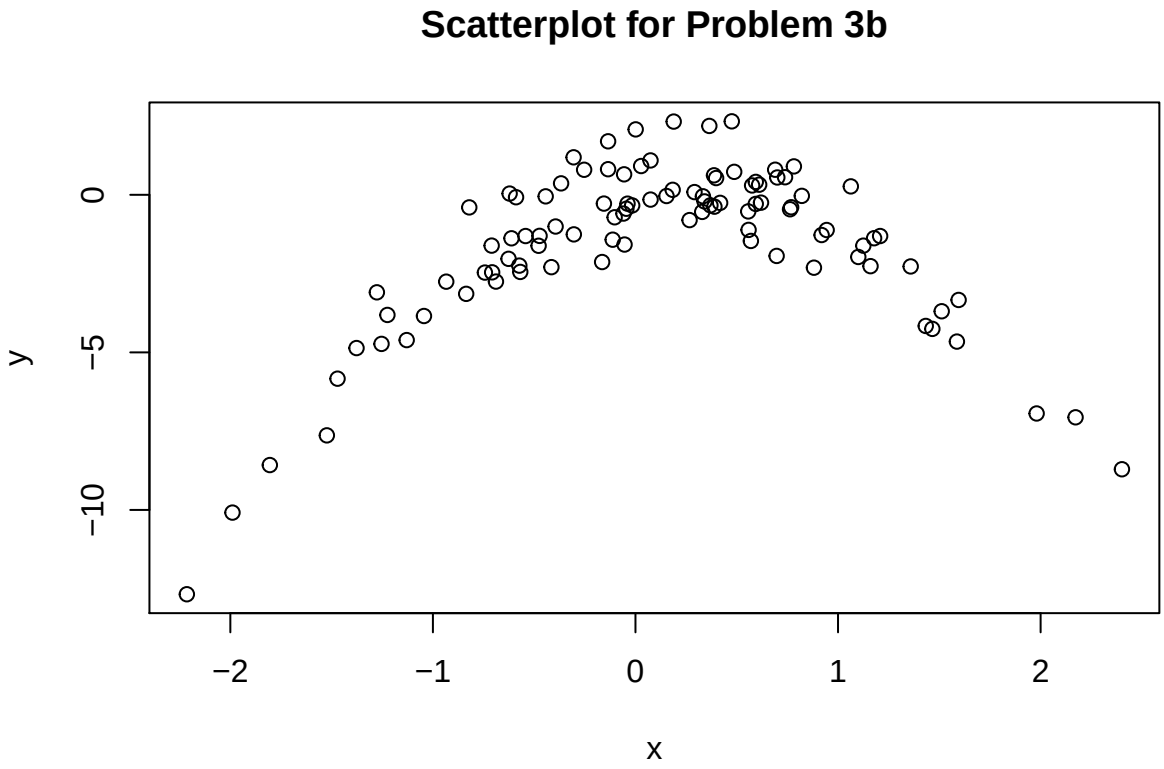
(a) In this data set, what is n and what is p? Write out the model used to generate the data in equation form. Generate a simulated data set as follows:

```
set.seed (1)
x = rnorm(100)
y = x - 2*x^2 + rnorm(100)
```

ANSWER: We have (n = 100) and (p = 2). The equation we use is: $Y = X - 2X^2 + \varepsilon$

(b) Create a scatterplot of X against Y. Comment on what you find.

```
plot(x, y, main = "Scatterplot for Problem 3b")
```



COMMENT: We can see clearly that the relationship between X and Y is that of a curved relationship. It is almost upside-down parabola-like curve.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

i) $Y = \beta_0 + \beta_1 X_1 + \epsilon$
ii) $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$
iii) $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$
iv) $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \epsilon$

Note: you may find it helpful to use the data.frame() function to create a single data set containing both X and Y .

```
set.seed(1)
df_problem3 <- data.frame(x, y)


m1A_glm <- glm(y ~ x)
m1A_cv_glm <- cv.glm(df_problem3, m1A_glm)$delta[1]

m1B_glm <- glm(y ~ poly(x, 2))
m1B_cv_glm <- cv.glm(df_problem3, m1B_glm)$delta[1]

m1C_glm <- glm(y ~ poly(x, 3))
```

```
m1C_cv_glm <- cv.glm(df_problem3, m1C_glm)$delta[1]

m1D_glm <- glm(y ~ poly(x, 4))
m1D_cv_glm <- cv.glm(df_problem3, m1D_glm)$delta[1]
```

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```
set.seed(2)
df_problem3 <- data.frame(x, y)


m2A_glm <- glm(y ~ x)
m2A_cv_glm <- cv.glm(df_problem3, m2A_glm)$delta[1]

m2B_glm <- glm(y ~ poly(x, 2))
m2B_cv_glm <- cv.glm(df_problem3, m2B_glm)$delta[1]

m2C_glm <- glm(y ~ poly(x, 3))
m2C_cv_glm <- cv.glm(df_problem3, m2C_glm)$delta[1]

m2D_glm <- glm(y ~ poly(x, 4))
m2D_cv_glm <- cv.glm(df_problem3, m2D_glm)$delta[1]

# Compare answers from part (c) and part (d)
compare_matrix <- matrix(c(m1A_cv_glm, m1B_cv_glm, m1C_cv_glm, m1D_cv_glm,
                           m2A_cv_glm, m2B_cv_glm, m2C_cv_glm, m2D_cv_glm),
                         ncol = 2, byrow = FALSE)
colnames(compare_matrix) <- c("Seed 1", "Seed 2")
rownames(compare_matrix) <- c("Model A", "Model B",
                              "Model C", "Model D")
compare_matrix
```

```
##            Seed 1    Seed 2
## Model A 7.2881616 7.2881616
## Model B 0.9374236 0.9374236
## Model C 0.9566218 0.9566218
## Model D 0.9539049 0.9539049
```

ANSWER: The reason why the results are the same results as Exercise 8c's is because the Leave-One-Out Cross-Validation (LOOCV) evaluates (n)-folds for a single observation.

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

ANSWER: We may see that the LOOCV estimate for the test MSE is minimum for "m1B_glm", this is not surprising since we saw clearly in (b) that the relation between "x" and "y" is quadratic.

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

```r
summary(m1D_glm)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

```r
summary(m1D_glm)$coefficients
```

```
##                Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)  -1.5500226 0.09590514 -16.1620379 5.169227e-29
## poly(x, 4)1   6.1888256 0.95905143   6.4530695 4.590732e-09
## poly(x, 4)2 -23.9483049 0.95905143 -24.9708243 1.593826e-43
## poly(x, 4)3   0.2641057 0.95905143   0.2753822 7.836207e-01
## poly(x, 4)4   1.2570950 0.95905143   1.3107691 1.930956e-01
```

ANSWER: The p-values show that the linear and quadratic terms are statistically significant. However, the cubic and 4th degree terms are not statistically significant. This agrees with our cross-validation results because the function was the minimum for the quadratic model.