Q1 Code

```
# Read in prostate cancer data
pc_data <- read.csv("prostate_cancer.csv")
# Eliminate subject number feature
pc_data <- pc_data[,-1]

# Convert gleason and treat vesinv as qualitative variables
pc_data$vesinv <- factor(pc_data$vesinv, order=F, levels = c(0, 1))
pc_data$gleason <- factor(pc_data$gleason, order=F, levels = c(6, 7, 8))

# Part a
# Preliminary findings
nrow(pc_data)
colnames(pc_data)
summary(pc_data$vesinv) # There's more people without seminal vesicle
invasion than with
summary(pc_data$gleason) # There's a mix of people with varying gleason
scores.
hist(pc_data$age) # Most people with pancreatic cancer information are older
(50-70+)
cor(pc_data[,unlist(lapply(pc_data, is.numeric))])
# age seems to have a high correlation with the amount of prostatic
hyperplasia
#, which is indicative of early stages of prostatic abnormality
# Capsular penetration, which indicates the outgrowth of cancerous tissue,
has a correlation with cancer volume


# Part b
# Examine distribution of psa to determine if it's an appropriate response
variable.
hist(pc_data[, 1])

# Since psa is not, transform it with a natural log transformation and check
again.
pc_data[, 1] <- log(pc_data[, 1])
hist(pc_data[, 1])

# Part c - For each predictor, fit a simple linear regression model to
predict the response
summary(glm(psa ~ cancervol, data = pc_data))$coefficients  #Significance
found
summary(glm(psa ~ weight, data = pc_data))$coefficients
summary(glm(psa ~ age, data = pc_data))$coefficients
summary(glm(psa ~ benpros, data = pc_data))$coefficients
summary(lm(psa ~ vesinv, data = pc_data))$coefficients      #Significance
found
summary(glm(psa ~ capspen, data = pc_data))$coefficients    #Significance
found
summary(lm(psa ~ gleason, data = pc_data))$coefficients     #Significance
found

# Plot individual features versus the response
par(mfrow=c(2,2))
plot(pc_data$cancervol, pc_data$psa, xlab = "cancervol", ylab="psa")
```

```r
plot(pc_data$vesinv, pc_data$psa, xlab = "vesinv", ylab="psa")
plot(pc_data$capspen, pc_data$psa, xlab = "capspen", ylab="psa")
plot(pc_data$gleason, pc_data$psa, xlab = "gleason", ylab="psa")

# Part d
# Compute multiple regression of all features against the response
summary(glm(psa ~ ., data = pc_data))
# We can reject the null hypothesis for cancervol, benpros, vesinv, and
gleason (linear)

# Part e
# I excluded capspen as it did not seem statistically significant when other
predictors were involved
# I excluded benpros as it did not have statistical significance to the psa
response until other predictors were accounted for

# Create final model with lm and find its coefficient estimates
final_model <- lm(psa ~ cancervol + vesinv + gleason, data = pc_data)
summary(final_model)
final_model$coefficients

# Part f - Linear equation of the final model
# psa = psa = 1.60467 + 0.05875*cancervol + 0.6259312*vesinv1 +
0.3543990*gleason7 + 0.7863444*gleason8


#Part g
# Create sample patient and predict its psa using the final model
sample_patient <- data.frame(mean(pc_data$cancervol),
names(sort(table(pc_data$vesinv)))[-1],
names(sort(table(pc_data$gleason)))[c(-1,-2)])
colnames(sample_patient) <- c("cancervol", "vesinv", "gleason")
predict(final_model, sample_patient)
```

Q2 Code

```
# Read in admission data
admit_data <- read.csv("admission.csv")
# Appropriately standardize the GPA and GMAT scores
admit_data[,1:2] <- scale(admit_data[,1:2])

# Form test data from the last 5 observations in each group
test_data <- rbind(
  tail(split(admit_data, admit_data$Group)$`1`, 5),
  tail(split(admit_data, admit_data$Group)$`2`, 5),
  tail(split(admit_data, admit_data$Group)$`3`, 5)
)

# Take the train data as the rest of the observations
train_data <- admit_data[-as.numeric(rownames(test_data)), ]

# Partition features and responses
train_X <- train_data[,1:2]
train_y <- train_data[,3]
test_X <- test_data[, 1:2]
test_y <- test_data[, 3]

# Count the number of observations
observation_count <- nrow(admit_data)

# Display frequency of GPA and GMAT data
par(mfrow=c(1,2))
hist(admit_data$GPA)
hist(admit_data$GMAT)

#Display frequency of response
hist(admit_data$Group)

par(mfrow=c(2,1))
plot(admit_data$GPA, admit_data$Group)
#GPA appears to correlate with acceptance
plot(admit_data$GMAT, admit_data$Group)
# Students with the highest GMAT scores tended to be accepted,
# mid range students had a mix of acceptance rates
# low scoring students often did not get accepted or were borderline

# Form grid for future decision boundary making
n_grid <- 50
gpa_grid <- seq(f=min(train_X$GPA), t=max(train_X$GPA), l=n_grid)
gmat_grid <- seq(f=min(train_X$GMAT), t=max(train_X$GMAT), l=n_grid)
grid <- expand.grid(gpa_grid, gmat_grid)
colnames(grid) <- c("GPA", "GMAT")

par(mfrow=c(2,2))
#Part b
library(MASS)
# Fit an lda function with Group as a response and GPA and GMAT as predictors
admit_lda <- lda(Group ~ GPA + GMAT, data=admit_data,
subset=rownames(train_data))
# Predict class values for a grid
```

```r
predict_lda_grid <- predict(admit_lda, grid)
# Store posterior probabilities for usage later to find decision boundaries
lda_probs_1 <- matrix(predict_lda_grid$posterior[,1], nrow=n_grid,
ncol=n_grid)
lda_probs_2 <- matrix(predict_lda_grid$posterior[,2], nrow=n_grid,
ncol=n_grid)
lda_probs_3 <- matrix(predict_lda_grid$posterior[,3], nrow=n_grid,
ncol=n_grid)

# Plot each decision boundary for LDA
plot(train_X, col= ifelse(train_y == 1, "green", ifelse(train_y == 2, "red",
"blue")), main="LDA decision boundaries")
contour(gpa_grid, gmat_grid, lda_probs_1, levels=0.5, add=T)
contour(gpa_grid, gmat_grid, lda_probs_2, levels=0.5, add=T)
contour(gpa_grid, gmat_grid, lda_probs_3, levels=0.5, add=T)

# Make confusion matrices and find misclassification error for both training
and test sets under the LDA model
predict_lda_train <- predict(admit_lda, train_X)
(cfm <- table(predict_lda_train$class, train_y))
(miss_train <- 1 - ((cfm[1,1] + cfm[2,2] + cfm[3,3])/sum(cfm)))
predict_lda_test <- predict(admit_lda, test_X)
(cfm <- table(predict_lda_test$class, test_y))
(miss_test <- 1 - ((cfm[1,1] + cfm[2,2] + cfm[3,3])/sum(cfm)))

#Part c
# Fit an qda function with Group as a response and GPA and GMAT as predictors
admit_qda <- qda(Group ~ GPA + GMAT, data=admit_data,
subset=rownames(train_data))
# Predict class values for a grid
predict_qda_grid <- predict(admit_qda, grid)
# Store posterior probabilities for usage later to find decision boundaries
qda_probs_1 <- matrix(predict_qda_grid$posterior[,1], nrow=n_grid,
ncol=n_grid)
qda_probs_2 <- matrix(predict_qda_grid$posterior[,2], nrow=n_grid,
ncol=n_grid)
qda_probs_3 <- matrix(predict_qda_grid$posterior[,3], nrow=n_grid,
ncol=n_grid)

# Plot each decision boundary for QDA
plot(train_X, col= ifelse(train_y == 1, "green", ifelse(train_y == 2, "red",
"blue")), main="QDA decision boundaries")
contour(gpa_grid, gmat_grid, qda_probs_1, levels=0.5, add=T)
contour(gpa_grid, gmat_grid, qda_probs_2, levels=0.5, add=T)
contour(gpa_grid, gmat_grid, qda_probs_3, levels=0.5, add=T)

# Make confusion matrices and find misclassification error for both training
and test sets under the QDA model
predict_qda_train <- predict(admit_qda, train_X)
(cfm <- table(predict_qda_train$class, train_y))
(miss_train <- 1 - ((cfm[1,1] + cfm[2,2] + cfm[3,3])/sum(cfm)))
predict_qda_test <- predict(admit_qda, test_X)
(cfm <- table(predict_qda_test$class, test_y))
(miss_test <- 1 - ((cfm[1,1] + cfm[2,2] + cfm[3,3])/sum(cfm)))

#Part d
library(e1071)
```

```r
# Fit a naive Bayes model with Group as the response and GPA and GMAT as
predictors
admit_nb <- naiveBayes(Group ~ GPA + GMAT, data=admit_data,
subset=rownames(train_data))
# Find posterior probabilities and store them by predicting with type="raw"
predict_nb_grid <- predict(admit_nb, grid, type="raw")
nb_probs_1 <- matrix(predict_nb_grid[,1], nrow=n_grid, ncol=n_grid)
nb_probs_2 <- matrix(predict_nb_grid[,2], nrow=n_grid, ncol=n_grid)
nb_probs_3 <- matrix(predict_nb_grid[,3], nrow=n_grid, ncol=n_grid)

# Plot decision boundaries for naive Bayes based on the model data found
plot(train_X, col= ifelse(train_y == 1, "green", ifelse(train_y == 2, "red",
"blue")), main="Naive Bayes decision boundaries")
contour(gpa_grid, gmat_grid, nb_probs_1, levels=0.5, add=T)
contour(gpa_grid, gmat_grid, nb_probs_2, levels=0.5, add=T)
contour(gpa_grid, gmat_grid, nb_probs_3, levels=0.5, add=T)

# Find confusion matrices and calculate the misclassification error for both
training and test sets under the NB model
predict_nb_train <- predict(admit_nb, train_X)
(cfm <- table(predict_nb_train, train_y))
(miss_train <- 1 - ((cfm[1,1] + cfm[2,2] + cfm[3,3])/sum(cfm)))
predict_nb_test <- predict(admit_nb, test_X)
(cfm <- table(predict_nb_test, test_y))
(miss_test <- 1 - ((cfm[1,1] + cfm[2,2] + cfm[3,3])/sum(cfm)))

#Part e
library(class)
set.seed(5)
#Test out multiple knn classifiers
ks <- c(seq(1,30, by=1), seq(35,150, by=5))
nks <- length(ks)
err_train <- numeric(length=nks)
err_test <- numeric(length=nks)
names(err_train) <- names(err_test) <- ks

# For each k number of neighbors, find the error rate for train and test data
for(i in seq(along=ks)){
  knn_train <- knn(train_X, train_X, train_y, k=ks[i], prob=TRUE)
  cfm <- table(knn_train, train_y)
  train_acc <- (cfm[1,1] + cfm[2,2] + cfm[3,3]) / sum(cfm)

  knn_test <- knn(train_X, test_X, train_y, k=ks[i], prob=TRUE)
  cfm <- table(knn_test, test_y)
  test_acc <- (cfm[1,1] + cfm[2,2] + cfm[3,3]) / sum(cfm)

  err_train[i] <- 1 - train_acc
  err_test[i] <- 1 - test_acc
}

# Plot train and test data to see more information on optimal K values
#plot(ks, err_train, xlab="K", ylab="Error rate", type = "b", col="green",
pch=20, ylim=c(0,1))
#lines(ks, err_test, type="b", col="red", pch=20)

# Find the optimal K value
optim_find <- data.frame(ks, err_train, err_test)
```

```r
optim_find[optim_find$err_test == min(optim_find$err_test), ]
# 1 and 55 are both found to be optimal

# Train a knn to predict the grid of values from before
optimal_knn <- knn(train_X, grid, train_y, k=55, prob=TRUE)

# Using the probabilities found from the knn classifier, find subjects where
their probability is close to 1/3
optimal_probs <- attr(optimal_knn, "prob")
optimal_probs <- matrix(optimal_probs, n_grid, n_grid)
all_boundary_subjects <- which(optimal_probs - 0.333 <= 0.05)
boundary_points <- grid[all_boundary_subjects,]

# Plot the training data and draw a line through the points having close to
1/3 chance of appearing to represent a decision boundary
plot(train_X, col= ifelse(train_y == 1, "green", ifelse(train_y == 2, "red",
"blue")), main="KNN decision boundary")
abline(lm(boundary_points$GMAT ~ boundary_points$GPA))

# Find cfms for the training and testing sets under KNN
(cfm_train_KNN <- table(knn(train_X, train_X, train_y, k=55, prob=TRUE),
train_y))
(cfm_test_KNN <- table(knn(train_X, test_X, train_y, k=55, prob=TRUE),
test_y))
1 - (cfm_train_KNN[1,1] + cfm_train_KNN[2,2] +
cfm_train_KNN[3,3])/sum(cfm_train_KNN)
1 - (cfm_test_KNN[1,1] + cfm_test_KNN[2,2] +
cfm_test_KNN[3,3])/sum(cfm_test_KNN)
```