Anna Hathaway

STAT:4540 Project Part 1

1.

i. a. rating = popularity + genre + mood

b. rating = popularity + genre + mood + popularity\*genre + popularity\*mood + genre\*mood

c. rating = popularity + genre + mood + popularity^2 + genre^2 + popularity\*genre + popularity\*mood + genre\*mood

Given mood is good and genre is 0,

1. X1 will increase by 0.372
2. X1 will increase by 0.393
3. X1 will increase by 0.390

ii. When looking at the normal QQ plot of all three models, you can see that the values do not follow the line well, especially at the ends. The consequences of this conclusion on the validity of the p-values in the output are then not valid since the assumptions are not met.

iii. see code

iv. Bootstrap confidence intervals are found by the formula (estimate +- 2\*standard error)

(0.01331494, 0.05794776) Reject the hypothesis that B1=0

(0.2369413, 0.2764655) Reject the hypothesis that B1=0

(-0.6858098, -0.6673438) Reject the hypothesis that B1=0

(-0.1415195, -0.09639295) Reject the hypothesis that B1=0

(-0.1364508, -0.09260217) Reject the hypothesis that B1=0

(-0.0329618, 0.0110919) Fail to reject the hypothesis that B1=0

v. The 10-fold cross-validation randomly divides the observations into 10 sets. The first set is the validation set and the method is fit on the remaining 9 sets. This procedure is repeated 10 times. We then can estimate the test MSE.

vi. Leave one out cross-validation removes the *i*th observation from the training data to be used for the validation. So it fits a model on training data with *n*-1 observations. Then, we can estimate the MSE.

**1.2.**

i. see code

ii. The test MSE is minimized when K=100

iii. see code

**1.3.**

**2.1**

i. Smoothness and compactness seem to be lightly correlated with the value of 0.68. Otherwise, we do not have any strongly correlated variables.

ii. Log texture mean increases by 0.63501 units for every 1 increase in rating. The 95% confidence interval is (0.48471, 0.78531) hence we reject the hypothesis that B1=0.

iii.

Y= log texture mean + log smoothness mean + log compactness mean + log texture mean\* log smoothness mean + log texture mean\* log compactness mean + log smoothness mean\* log compactness mean

For every one increase in rating, log texture mean increases by 3.6893 units when log smoothness mean and log compactness mean remain equal.

iv. see code

**2.2**

i. The LDA approach to classification of malignant and benign tumors assumes that the observatins in the *k*th class are from the Gaussian distribution of N(Muk, Sigma). The LDA will predict how many people who tested malignant were actually positive or were a false positive. The same goes for a benign tumor.

ii. The QDA approach also draws from a Gaussian distribution but it assumes that each class has its own covariance matrix. So observations from the *k*th class is of N(Muk, Sigmak). In context of the problem, it does the same thing except QDA estimates a separate covariance matrix for each class of malignant or benign.

iii. They are equal because they are meeting the same assumptions of our linear models.

2.3 & 2.4 see code

#Question 1.1

#(i)

setwd("~/Downloads")

mov\_train <- read.csv(file= "mov\_train.csv")

mdl\_a <- lm(rating ~ popularity + genre + mood, data=mov\_train)

mdl\_b <- lm(rating ~ popularity + genre + mood + popularity:genre + popularity:mood + genre:mood, data=mov\_train)

mdl\_c <- lm(rating ~ popularity + genre + mood + I(popularity^2) + I(genre^2) + popularity:genre + popularity:mood + genre:mood, data=mov\_train)

coef(mdl\_a)

coef(mdl\_b)

coef(mdl\_c)

#(ii)

plot(mdl\_a)

plot(mdl\_b)

plot(mdl\_c)

#(iii)

library(boot)

boot.fn.cor = function(data, index) {

res = cov2cor(vcov(lm(rating ~ popularity + genre + mood, data=data, subset = index)))

return(c(res[lower.tri(res)]))

}

boot.cor= boot(mov\_train, boot.fn.cor, 1000)

#(iv)

boot.cor

0.03563135 - 2\*0.011158205

0.03563135 + 2\*0.011158205

0.25670339 - 2\*0.009881046

0.25670339 + 2\*0.009881046

-0.67657681 - 2\*0.004616508

-0.67657681 + 2\*0.004616508

-0.11895624 - 2\*0.011281647

-0.11895624+ 2\*0.011281647

-0.11452647 - 2\*0.010962150

-0.11452647+ 2\*0.010962150

-0.01093495 - 2\*0.011013426

-0.01093495 + 2\*0.011013426

#10 fold CV

rm(list = ls())

library(ISLR)

library(FNN)

K = c(1, )

trainMseList = list()

testMseList = list()

foldId <- sample(1:10, nrow(mov\_train), replace = TRUE)

testMse = matrix(NA, 10, 3)

for (cc in 1:10) {

cat("Fold: ", cc, "\n")

testIdx <- which(foldId == cc)

trainIdx <- which(foldId != cc)

testdf = mov\_train[testIdx, ]

traindf = mov\_train[trainIdx, ]

mdl1 <- lm(rating ~ popularity + genre + mood, data = traindf)

mdl2 <- lm(rating ~ popularity + genre + mood + popularity:genre + popularity:mood + genre:mood, data=traindf)

mdl3 <- lm(rating ~ popularity + genre + mood + I(popularity^2) + I(genre^2) + popularity:genre + popularity:mood + genre:mood, data=traindf)

testMse[cc, 1] = mean((testdf$rating - predict(mdl1, testdf))^2)

testMse[cc, 2] = mean((testdf$rating - predict(mdl2, testdf))^2)

testMse[cc, 3] = mean((testdf$rating - predict(mdl3, testdf))^2)

}

colMeans(testMse)

#LOOCV

for (cc in 1:nrow(mov\_train)) {

cat("cc:", cc, "\n")

trainIdx <- (1:nrow(mov\_train))[-cc]

traindf <- mov\_train[trainIdx]

testdf <- mov\_train[cc]

Klist[[cc]] <- seq(1, nrow(traindf), by = 1)

K <- Klist[[cc]]

testMse <- numeric(length(K))

trainMse <- numeric(length(K))

for (ii in K) {

kresTest <- knn.reg(train = as.matrix(traindf),

test = as.matrix(testdf),

y = traindf$rating, k = K[ii])

testMse[ii] <- ((testdf$rating - kresTest$pred)^2)

kresTrain <- knn.reg(train = as.matrix(traindf),

test = as.matrix(traindf),

y = traindf$rating, k = K[ii])

trainMse[ii] <- mean((traindf$rating - kresTrain$pred)^2)

}

trainMseList[[cc]] <- trainMse

testMseList[[cc]] <- testMse

}

#Question 1.2

#(i)

testMse <- matrix(NA, 10, length(K))

foldId <- sample(1:10, nrow(mov\_Train), replace = TRUE)

for (cc in 1:10) {

cat("Fold: ", cc, "\n")

testIdx <- which(foldId == cc)

trainIdx <- which(foldId != cc)

traindf <- mov\_Train[trainIdx, ]

testdf <- mov\_Train[testIdx, ]

for (ii in 1:length(K)) {

kresTest <- knn.reg(train = as.matrix(traindf[ , -1]),

test = as.matrix(testdf[ , -1]),

y = traindf$rating,

k = K[ii])

testMse[cc, ii] <- mean((testdf$rating - kresTest$pred)^2)

}

}

plot(rev(1 / K), colMeans(testMse), type = "l")

#(ii)

knn1.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn1.tab3 <- table(knn1.class, testdf)

knn1.tab <- knn1.tab3

colnames(knn1.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn1.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn1.tab)

knn10.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn10.tab3 <- table(knn1.class, testdf)

knn10.tab <- knn1.tab3

colnames(knn10.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn10.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn10.tab)

knn50.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn50.tab3 <- table(knn1.class, testdf)

knn50.tab <- knn1.tab3

colnames(knn50.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn50.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn50.tab)

knn100.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn100.tab3 <- table(knn1.class, testdf)

knn100.tab <- knn1.tab3

colnames(knn100.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn100.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn100.tab)

knn200.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn200.tab3 <- table(knn1.class, testdf)

knn200.tab <- knn1.tab3

colnames(knn200.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn200.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn200.tab)

knn500.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn500.tab3 <- table(knn1.class, testdf)

knn500.tab <- knn1.tab3

colnames(knn500.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn500.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn500.tab)

knn1000.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn1000.tab3 <- table(knn1.class, testdf)

knn1000.tab <- knn1.tab3

colnames(knn1000.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn1000.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn1000.tab)

knn1500.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn1500.tab3 <- table(knn1.class, testdf)

knn1500.tab <- knn1.tab3

colnames(knn1500.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn1500.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn1500.tab)

knn2000.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn2000.tab3 <- table(knn1.class, testdf)

knn2000.tab <- knn1.tab3

colnames(knn2000.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn2000.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn2000.tab)

knn3000.class <- knn(train = as.matrix(kresTest[ ,2:3]),

test = as.matrix(kresTrain[ ,2:3]),

cl = kresTrain, k = 1)

knn3000.tab3 <- table(knn1.class, testdf)

knn3000.tab <- knn1.tab3

colnames(knn3000.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn3000.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn3000.tab)

K <- sort(c(1, 10, 50, 100, 200, 500, 1000, 1500, 2000, 3000))

kres <- list()

for (ii in seq\_along(K)) {

kres[[ii]] <- knn.reg(train = as.matrix(mov\_train$popularity, mov\_train$genre, mov\_train$mood),

test = as.matrix(hpgrid),

y = mov\_train$rating, k = K[ii])

}

#(iii)

library(boot)

variance.fn <- function(data, index) {

knn.fit <- knn.reg(train = as.matrix(data[index, c("popularity", "genre", "mood")]),

test = as.matrix(data[ , c("popularity", "genre", "mood")]),

y = data[index, "rating"],

k = 200)

# get variance from predicted values

variance <- var(data$rating - knn.fit$pred)

return(variance)

}

boot.var <- boot(movTrain, variance.fn, 200)

boot.var.ci <- quantile(as.numeric(boot.var$t), prob = c(0.025, 0.975))

#Question 2

wisc <- read.csv(file="wisc.csv")

#(i)

pairs(wisc[,2:4])

cor(wisc[,2:4])

#(ii)

mdl2.1 <- glm(as.numeric(diagnosis == "M") ~ log\_texture\_mean +log\_smoothness\_mean + log\_compactness\_mean, data=wisc)

summary(mdl2.1)

0.63501 - 2\*0.07515

0.63501 + 2\*0.07515

#(iii)

mdl2.2 <- glm(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean + log\_texture\_mean:log\_smoothness\_mean +log\_texture\_mean:log\_compactness\_mean + log\_smoothness\_mean:log\_compactness\_mean, data=wisc)

summary(mdl2.2)

#(iv)

mdl2.3 <- glm(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean + I(log\_texture\_mean)^2 + I(log\_smoothness\_mean)^2 + I(log\_compactness\_mean)^2 + log\_texture\_mean:log\_smoothness\_mean + log\_texture\_mean:log\_compactness\_mean + log\_smoothness\_mean:log\_compactness\_mean, data=wisc)

#Question 2.2

#(iii)

library(MASS)

lda.fit <- lda(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean, data = wisc)

lda.fit

lda\_pred = predict(lda.fit, wisc)

lda\_class = lda\_pred$diagnosis

table(lda\_class,wisc$diagnosis)

qda\_fit = qda(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean, data = wisc)

qda\_pred = predict(qda\_fit, wisc)

qda\_class = qda\_pred$diagnosis

table(qda\_class,wisc$diagnosis)

#Question 2.3

#(i)

library(boot)

boot.fn.cor2 = function(data, index) {

res = cov2cor(vcov(lm(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean, data = wisc, subset = index)))

return(c(res[lower.tri(res)]))

}

boot.cor= boot(wisc\_eval, boot.fn.cor2, 1000)

rm(list = ls())

K = c(1, )

trainMseList = list()

testMseList = list()

foldId <- sample(1:10, nrow(wisc\_eval), replace = TRUE)

testMse = matrix(NA, 10, 3)

for (cc in 1:10) {

cat("Fold: ", cc, "\n")

testIdx <- which(foldId == cc)

trainIdx <- which(foldId != cc)

testdf = wisc\_eval[testIdx, ]

traindf = wisc\_eval[trainIdx, ]

mdl1 <- lm(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean, data = wisc)

mdl2 <- lm(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean + log\_texture\_mean:log\_smoothness\_mean +log\_texture\_mean:log\_compactness\_mean + log\_smoothness\_mean:log\_compactness\_mean)

mdl3 <- lm(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean + I(log\_texture\_mean)^2 + I(log\_smoothness\_mean)^2 + I(log\_compactness\_mean)^2 + log\_texture\_mean:log\_smoothness\_mean + log\_texture\_mean:log\_compactness\_mean + log\_smoothness\_mean:log\_compactness\_mean)

testMse[cc, 1] = mean((testdf$rating - predict(mdl1, testdf))^2)

testMse[cc, 2] = mean((testdf$rating - predict(mdl2, testdf))^2)

testMse[cc, 3] = mean((testdf$rating - predict(mdl3, testdf))^2)

}

colMeans(testMse)

#Question 2.4

#logistic regression confusion matrix

wisc\_eval <- read.csv(file= "wisc\_eval.csv")

logistic.fit <- glm(das.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean, data = wisc, family = binomial)

summary(logistic.fit)

tab3 <- table(logistic.pred1, wisc\_eval$diagnosis)

tab <- tab3

colnames(tab3) <- c("true-1", "true-2", "true-3")

rownames(tab3) <- c("pred-1", "pred-2", "pred-3")

conf1 <- matrix(NA, 2, 2)

colnames(conf1) <- c("true-1", "true-0")

rownames(conf1) <- c("pred-1", "pred-0")

conf1[1, 1] <- tab3[1, 1]

conf1[1, 2] <- tab3[1, 2] + tab3[1, 3]

conf1[2, 1] <- tab3[2, 1] + tab3[3, 1]

conf1[2, 2] <- tab3[2, 2] + tab3[2, 3] + tab3[3, 2] + tab3[3, 3]

#LDA

lda.fit <- lda(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean, data = wisc)

lda.pred <- predict(lda.fit, wisc\_eval)

lda.prob <- lda.pred$posterior

lda.class <- lda.pred$class

lda.tab3 <- table(lda.class, wisc\_eval$diagnosis)

lda.tab <- lda.tab3

colnames(lda.tab3) <- c("true-1", "true-2", "true-3")

rownames(lda.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(lda.tab)

#QDA

qda.fit <- qda(as.numeric(diagnosis == "M") ~ log\_texture\_mean + log\_smoothness\_mean + log\_compactness\_mean, data = wisc)

qda.pred <- predict(qda.fit, wisc\_eval)

qda.prob <- qda.pred$posterior

qda.class <- qda.pred$diagnosis

qda.tab3 <- table(qda.class, qualityTest$category)

qda.tab <- qda.tab3

colnames(qda.tab3) <- c("true-1", "true-2", "true-3")

rownames(qda.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(qda.tab)

#KNN

knn1.class <- knn(train = as.matrix(qualityTrain[ , 2:3]),

test = as.matrix(qualityTest[ , 2:3]),

cl = qualityTrain$category, k = 1)

knn1.tab3 <- table(knn1.class, wisc\_eval$diagnosis)

knn1.tab <- knn1.tab3

colnames(knn1.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn1.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn1.tab)

knn5.class <- knn(train = as.matrix(qualityTrain[ , 2:3]),

test = as.matrix(qualityTest[ , 2:3]),

cl = qualityTrain$category, k = 5)

knn5.tab3 <- table(knn1.class, wisc\_eval$diagnosis)

knn5.tab <- knn5.tab3

colnames(knn5.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn5.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn5.tab)

knn10.class <- knn(train = as.matrix(qualityTrain[ , 2:3]),

test = as.matrix(qualityTest[ , 2:3]),

cl = qualityTrain$category, k = 10)

knn10.tab3 <- table(knn1.class, wisc\_eval$diagnosis)

knn10.tab <- knn5.tab3

colnames(knn10.tab3) <- c("true-1", "true-2", "true-3")

rownames(knn10.tab3) <- c("pred-1", "pred-2", "pred-3")

confusionMatrix(knn10.tab)