STAT4540Proj_1

```
library(boot)
library(class)
library(FNN)
library(caret)
library(MASS)
library(ggplot2)
library(tidyverse)
library(knitr)


#read in data/load in packages

mov_test <- read.csv("mov_eval.csv", na.strings="NA")

mov_train <- read.csv("mov_train.csv", na.strings="NA")

wisc_train <- read.csv("wisc.csv", na.strings="NA")

wisc_test <- read.csv("wisc_eval.csv", na.strings="NA")
```

1.1

i

- Creating 3 linear regression models

```
lm1 <- lm(rating ~ ., mov_train)

summary(lm1)
```

```
##
## Call:
## lm(formula = rating ~ ., data = mov_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7954 -0.5973  0.1185  0.7810  2.7618
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.346929   0.017213 194.437   <2e-16 ***
## popularity   0.372350   0.011462  32.486   <2e-16 ***
```

```
## genre       -0.001541   0.024381  -0.063     0.95
## mood         0.377842   0.024028  15.725   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.067 on 7996 degrees of freedom
## Multiple R-squared:  0.1541, Adjusted R-squared:  0.1538
## F-statistic: 485.7 on 3 and 7996 DF,  p-value: < 2.2e-16
```

```
lm2 <- lm(rating ~ popularity + genre + mood + I(popularity * genre)
+ I(popularity * mood) + I(genre * mood), mov_train)

summary(lm2)
```

```
##
## Call:
## lm(formula = rating ~ popularity + genre + mood + I(popularity *
##     genre) + I(popularity * mood) + I(genre * mood), data = mov_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7803 -0.6080  0.1183  0.7920  2.8177
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             3.3426848  0.0178249 187.529   <2e-16 ***
## popularity              0.3936773  0.0165259  23.822   <2e-16 ***
## genre                  -0.0355675  0.0346376  -1.027   0.3045
## mood                    0.3911127  0.0255521  15.306   <2e-16 ***
## I(popularity * genre)   0.0004583  0.0229205   0.020   0.9840
## I(popularity * mood)   -0.0447837  0.0229733  -1.949   0.0513 .
## I(genre * mood)         0.0687274  0.0490437   1.401   0.1611
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.066 on 7993 degrees of freedom
## Multiple R-squared:  0.1547, Adjusted R-squared:  0.1541
## F-statistic: 243.8 on 6 and 7993 DF,  p-value: < 2.2e-16
```

```
lm3 <- lm(rating ~ popularity + genre + mood + I(popularity^2)
+ I(genre^2) + I(popularity * genre) + I(popularity *mood)
+ I(genre * mood), mov_train)

summary(lm3)
```

```
##
## Call:
## lm(formula = rating ~ popularity + genre + mood + I(popularity^2) +
##     I(genre^2) + I(popularity * genre) + I(popularity * mood) +
##     I(genre * mood), data = mov_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.7882 -0.6097  0.1063  0.7888  2.9020
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             3.331398   0.021829 152.612   <2e-16 ***
## popularity              0.390885   0.016635  23.497   <2e-16 ***
## genre                   0.032407   0.049236   0.658   0.5104
## mood                    0.388532   0.025576  15.191   <2e-16 ***
## I(popularity^2)        -0.008568   0.006804  -1.259   0.2080
## I(genre^2)              0.122816   0.063219   1.943   0.0521 .
## I(popularity * genre)   0.004278   0.023153   0.185   0.8534
## I(popularity * mood)   -0.039920   0.023330  -1.711   0.0871 .
## I(genre * mood)         0.070408   0.049046   1.436   0.1512
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.066 on 7991 degrees of freedom
## Multiple R-squared:  0.1552, Adjusted R-squared:  0.1544
## F-statistic: 183.6 on 8 and 7991 DF,  p-value: < 2.2e-16
```

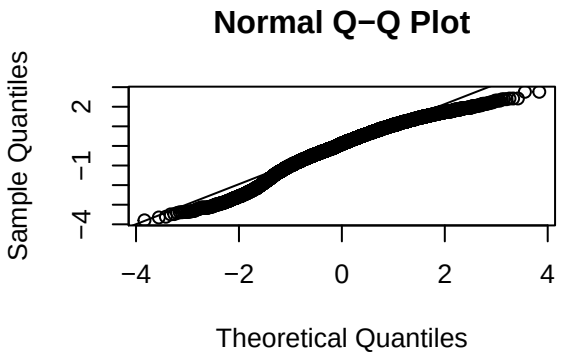
ii

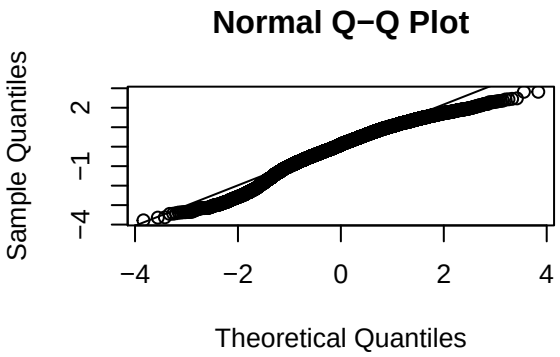
- QQnorm plots to show that the irreducible errors in all 3 models are not normally distributed.

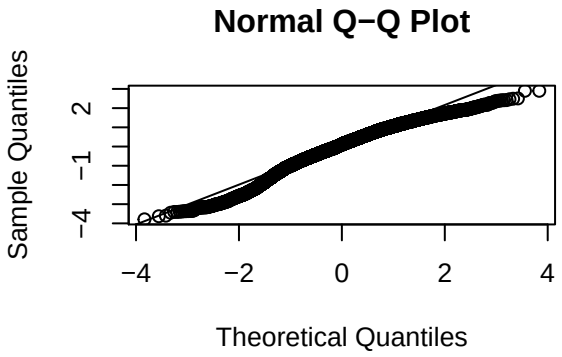
```
par(mfrow=c(2,2))
qqnorm(lm1$residuals)
qqline(lm1$residuals)

qqnorm(lm2$residuals)
qqline(lm2$residuals)

qqnorm(lm3$residuals)
qqline(lm3$residuals)
```







iii

- Bootstrap to construct 95% confidence intervals for Beta1 and elements of the correlation matrix of (beta0,beta1,beta2,beta3) in lm1.

```
set.seed(4540)
# boot function for \beta_1
fct_coe <- function(data, index){
            coef(lm(rating ~ popularity + genre + mood, data, subset
            = index))[2]
}

# boot function for cor
# The boot.cor object has rows corresponding to
# cor(beta_0, beta_1)
# cor(beta_0, beta_2)

# cor(beta_0, beta_3)
# cor(beta_1, beta_2)
# cor(beta_1, beta_3)
# cor(beta_2, beta_3)
boot.fn.cor = function(data, index) {
  res = cov2cor(vcov(lm(rating ~ popularity + genre + mood,
                        data=data, subset = index)))
```

```
  return(c(res[lower.tri(res)]))
}
# bootstrap replicates
boot.beta1 <- boot(mov_train, fct_coe, 1e3)
boot.cor <- boot(mov_train, boot.fn.cor, 1e3)
# 95% bootstrap percentile CI
boot.ci(boot.beta1, type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.beta1, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.3522,  0.3963 )
## Calculations and Intervals on Original Scale
```

```
library(knitr)

boot_CI_cor <- apply(boot.cor$t, 2, quantile, probs =
                c(0.025,0.975))
colnames(boot_CI_cor) <- c("cor(beta_0, beta_1)", "cor(beta_0, beta_2)",
          "cor(beta_0, beta_3)", "cor(beta_1, beta_2)", "cor(beta_1, beta_3)",
          "cor(beta_2, beta_3)")

kable(boot_CI_cor, caption = "95% bootstrap percentile CI for
Correlations")
```

Table 1: 95% bootstrap percentile CI for Correlations

	cor(beta_0, beta_1)	cor(beta_0, beta_2)	cor(beta_0, beta_3)	cor(beta_1, beta_2)	cor(beta_1, beta_3)	cor(beta_2, beta_3)
2.5%	0.0143302	0.2376899	-0.6843828	-0.1422779	-0.1358167	-0.0341244
97.5%	0.0569661	0.2766663	-0.6670688	-0.0981741	-0.0928118	0.0104627

iv

The 95% confidence interval for Beta1 is [0.34691, 0.3944], suggesting that we are 95% confident that the confidence interval can capture the true beta1. Since the null hypothesis is Beta1 = 0, which is not included in this interval, we reject the null hypothesis.

v

- Finding the best model using 10-fold cross valuation, the test error rate for all 3 models is very close, but lm3 has a slightly smaller test error.

```
lm1 <- glm(rating ~ popularity + genre + mood, family=gaussian(link="identity")
           , mov_train)

lm2 <- glm(rating ~ popularity * genre + popularity * mood + genre:
            mood, family = gaussian(link="identity"), mov_train)

lm3 <- glm(rating ~ popularity * genre + popularity * mood +
          genre:mood + I(popularity^2) + I(genre^2), family =
          gaussian(link="identity"), mov_train)


(cv_er1 <- cv.glm(mov_train, lm1, K = 10)$delta[1])
```

```
## [1] 1.13849
```

```
(cv_er2 <- cv.glm(mov_train, lm2, K = 10)$delta[1])
```

```
## [1] 1.1388
```

```
(cv_er3 <- cv.glm(mov_train, lm3, K = 10)$delta[1])
```

```
## [1] 1.138318
```

vi

- Finding the best model using leave-one-out cross-valuation. lm1 has the smallest test error so it is the best model.

```
msehatloocv1 <- mean(((mov_train$rating - lm1$fitted.values) /
                (1 - hatvalues(lm1))) ^ 2)

msehatloocv2 <- mean(((mov_train$rating - lm1$fitted.values) /
                      (1 - hatvalues(lm2))) ^ 2)

msehatloocv3 <- mean(((mov_train$rating - lm1$fitted.values) /
                      (1 - hatvalues(lm3))) ^ 2)

print(c(msehatloocv1, msehatloocv2, msehatloocv3))
```

```
## [1] 1.138217 1.139054 1.139609
```

1.2

i

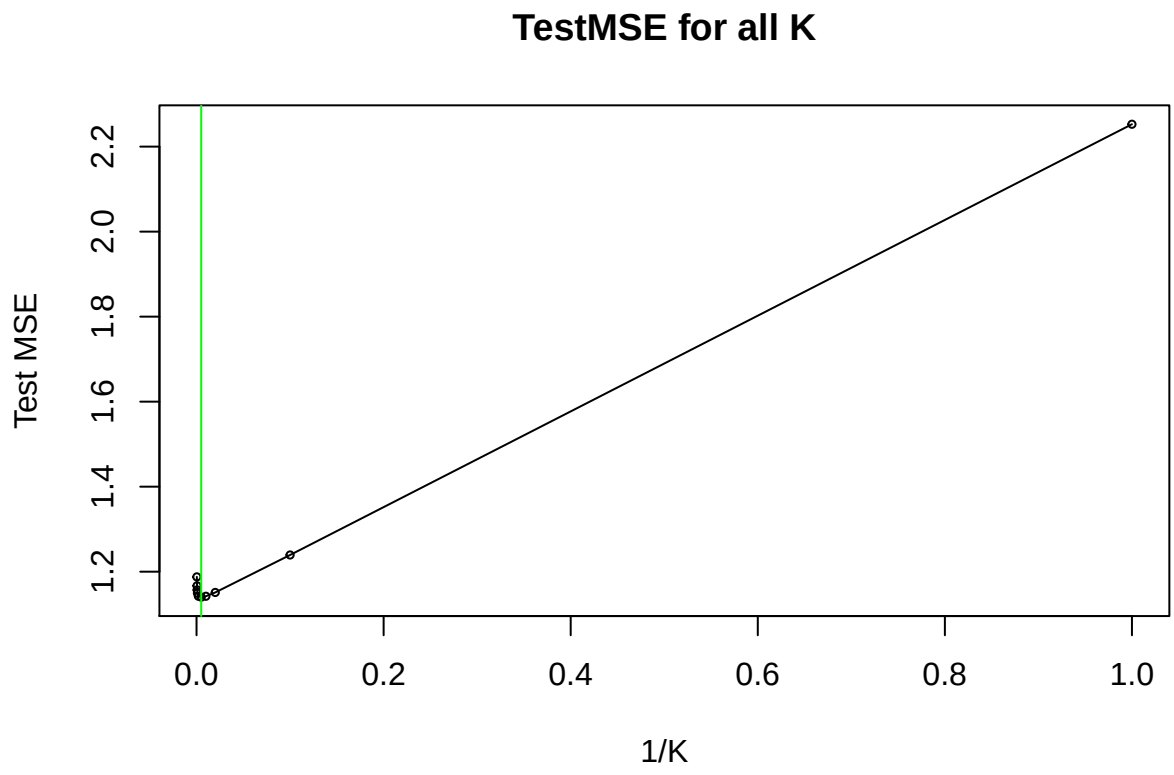
- Constructing the test MSE curve as a function of 1/K using 10-fold cross- validation =.

```r
set.seed(4540)
folds <- createFolds(y = mov_train$rating, k = 10) # folds[[i]] is the i-th fold
Klist = c(1, 10, 50, 100, 200, 500, 1000, 1500, 2000, 3000)

testMSE <- function(K) {
  mse_fold <- rep(NA, 10)
  for (i in 1:10) {
    train_cv <- mov_train[-folds[[i]],]
    test_cv <- mov_train[folds[[i]],]
    kreg <- knn.reg(train = train_cv[, 2:4], test = test_cv[, 2:4],
            y = train_cv$rating, K)
    # obtain prediction of test set immediately.
    mse_fold[i] <- mean((test_cv$rating - kreg$pred)^2)
  }
return(mean(mse_fold))
}

# calculate testmse for all possible K
Testmse <- sapply(Klist, testMSE)
# plot
plot(rev(1/Klist), rev(Testmse), cex = 0.5, xlab = "1/K", ylab =
      "Test MSE", main = "TestMSE for all K" )

lines(rev(1/Klist), rev(Testmse))
abline(v = 1 / Klist[which.min(Testmse)], lwd = 1, col = "green")
```



ii

- Find the best K using the test MSE curve

```
names(Testmse) <- c(1, 10, 50, 100, 200, 500, 1000, 1500, 2000, 3000)
Testmse
```

```
##        1       10       50      100      200      500     1000     1500
## 2.252621 1.239277 1.151105 1.142172 1.140094 1.142261 1.149562 1.157004
##     2000     3000
## 1.166753 1.187744
```

```
Klist[which.min(Testmse)]
```

```
## [1] 200
```

iii

- Based on the chosen k-NN model, constructed a 95% confidence interval for sigma^2 using bootstrap.

```
set.seed(4540)
boot.var <- function(data, i){
  knn_reg <- knn.reg(train = data[i, 2:4], test = data[, 2:4], y =
                       data[i, 1], k = 200)
  error <- data[, 1] - knn_reg$pred
  return(var(error))
}

boot.var <- boot(mov_train, boot.var, 200)
(boot.var.ci <- quantile(as.numeric(boot.var$t), prob = c(0.025,
                                                          0.975)))
```

```
##     2.5%    97.5%
## 1.131035 1.137431
```

1.3

- Estimating the test MSEs for the choses linear regression and K-NN models. Between lm3 and KNN model where K = 200, the linear regression model is better because it has the smaller error rate.

```
(mse_lm3 <- mean((mov_test$rating - predict(lm3, mov_test))^2))
```

```
## [1] 1.173243
```

```
knn_reg <- knn.reg(train = mov_train[, 2:4], test = mov_test[, 2:4],
            y = mov_train[, 1], k = 200)
(mse_knn <- mean((mov_test$rating - knn_reg$pred)^2))
```

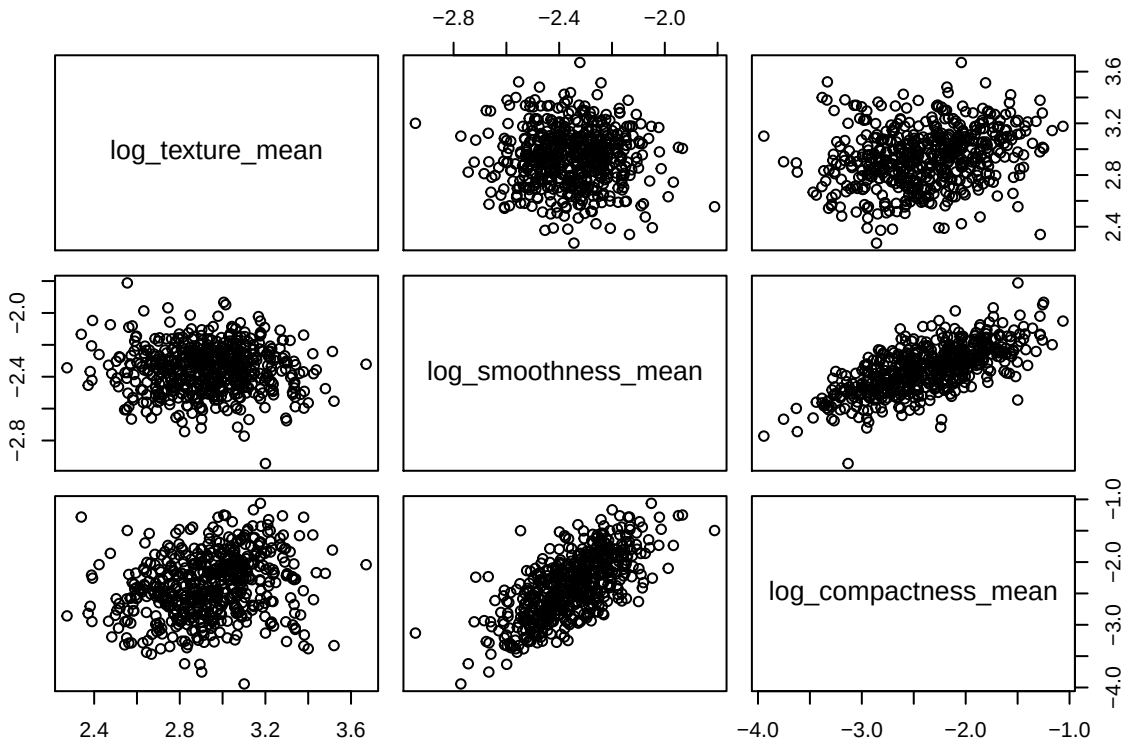
```
## [1] 1.187296
```

2.1

i

- Checking for dependice among the predictors by using pair plots and a correlation matrix. There seems to be an obvious positive linear relationship between log_smoothness_mean and log_compactness_mean. No significant dependence among other pairs.

```
wisc_train$diagnosis <- ifelse(wisc_train$diagnosis == "M", 1, 0) # Y = 1 --M
pairs(wisc_train[, -1])
```




```
cor(wisc_train[, -1])
```

```
##                       log_texture_mean log_smoothness_mean log_compactness_mean
## log_texture_mean           1.00000000         -0.01645797            0.2411181
## log_smoothness_mean       -0.01645797          1.00000000            0.6821049
## log_compactness_mean       0.24111814          0.68210493            1.0000000
```

ii

- Fit a logistic regression model with Y as the response and X1,X2 and X3 as predictors. The 95% confidence interval for Beta1 hat is [4.048455, 6.969725] which means that we are 95% confident that the confidence interval will capture the true beta1.

```
lg1 <- glm(diagnosis ~., family = binomial, wisc_train)
summary(lg1)
```

```
##
## Call:
## glm(formula = diagnosis ~ ., family = binomial, data = wisc_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3316  -0.5432  -0.1922   0.5010   2.4845
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -5.7618     2.7381  -2.104   0.0354 *
## log_texture_mean       5.4550     0.7432   7.339 2.14e-13 ***
## log_smoothness_mean    0.7287     1.2280   0.593   0.5529
## log_compactness_mean   4.0258     0.4517   8.912  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 712.94  on 539  degrees of freedom
## Residual deviance: 394.51  on 536  degrees of freedom
## AIC: 402.51
##
## Number of Fisher Scoring iterations: 6
```

```
confint(lg1)[2,]
```

```
##    2.5 %   97.5 %
## 4.048455 6.969725
```

iii

- The logistic regression model with Y as the response and X1,X2,X3,X1X2,X2X3, X1X3 as predictors.

```
lg2 <- glm(diagnosis ~ log_texture_mean * log_smoothness_mean
      + log_texture_mean * log_compactness_mean +
      log_smoothness_mean : log_compactness_mean, family =
      binomial, wisc_train)

coef(lg2)
```

```
##                           (Intercept)
##                            -85.422867
##                      log_texture_mean
##                             42.984341
##                     log_smoothness_mean
##                            -36.977170
```

```
##                     log_compactness_mean
##                                21.706424
##     log_texture_mean:log_smoothness_mean
##                                17.235624
##    log_texture_mean:log_compactness_mean
##                                -1.317384
## log_smoothness_mean:log_compactness_mean
##                                 5.829145
```

iv

- Added a 3rd model and used 10-fold cross-validation to choose the best model based on the overall error rate in correctly classifying the malignant tumors.

```
set.seed(4540)
folds <- createFolds(y = wisc_train$diagnosis, k = 10) # folds[[i]] is
                                                       # the i-th fold

aler_corm <- function(c) {
    overall_error_fold <- matrix(NA, 3, 10)
    correct_M_fold <- matrix(NA, 3, 10)

    for (i in 1:10) {
      train_cv <- wisc_train[-folds[[i]], ]
      test_cv <- wisc_train[folds[[i]], ]


      lg1 <- glm(diagnosis ~., family = binomial, train_cv)
      prob_lg1 <- predict(lg1, test_cv, type = "response")
      pred_lg1 <- ifelse(prob_lg1 > c, 1, 0)
      CM <- table(pred_lg1, test_cv$diagnosis)
      correct_M <- CM[2,2] / sum(CM[, 2])
      overall_error <- mean(pred_lg1 != test_cv$diagnosis)
      overall_error_fold[1, i] <- overall_error
      correct_M_fold[1, i] <- correct_M

      lg2 <- glm(diagnosis ~ log_texture_mean * log_smoothness_mean +
                 log_texture_mean * log_compactness_mean +
                 log_smoothness_mean : log_compactness_mean, family
                 = binomial, train_cv)

      prob_lg2 <- predict(lg2, test_cv, type = "response")
      pred_lg2 <- ifelse(prob_lg2 > c, 1, 0)
      CM <- table(pred_lg2, test_cv$diagnosis)
      correct_M <- CM[2,2] / sum(CM[, 2])
      overall_error <- mean(pred_lg2 != test_cv$diagnosis)
      overall_error_fold[2, i] <- overall_error
      correct_M_fold[2, i] <- correct_M


      lg3 <- glm(diagnosis ~ log_texture_mean * log_smoothness_mean +
                 log_texture_mean * log_compactness_mean +
```

```
                log_smoothness_mean : log_compactness_mean +
                I(log_texture_mean^2) + I(log_smoothness_mean^2) +
                I(log_compactness_mean^2), family = binomial,
                train_cv)

      prob_lg3 <- predict(lg3, test_cv, type = "response")
      pred_lg3 <- ifelse(prob_lg3 > c, 1, 0)
      CM <- table(pred_lg3, test_cv$diagnosis)
      correct_M <- CM[2,2] / sum(CM[, 2])
      overall_error <- mean(pred_lg3 != test_cv$diagnosis)
      overall_error_fold[3, i] <- overall_error
      correct_M_fold[3, i] <- correct_M

    }
    all_error <- rowMeans(overall_error_fold)
    correct_M <- rowMeans(correct_M_fold)
    return(c(all_error, correct_M))
}

c <- c(0.25, 0.35, 0.45, 0.55, 0.65)
error_corretM <- sapply(c, aler_corm)
colnames(error_corretM) = c(0.25, 0.35, 0.45, 0.55, 0.65)

overall_error_rate <- error_corretM[1:3, ]
rownames(overall_error_rate) = c("lg_a", "lg_b", "lg_c")
# overall_error_rate

accuracy_correctly_M <- error_corretM[4:6, ]
rownames(accuracy_correctly_M) = c("lg_a", "lg_b", "lg_c")
# accuracy_correctly_M

kable(overall_error_rate, caption = "Overall error rate for logistic
      model a, b, and c")
```

Table 2: Overall error rate for logistic model a, b, and c

	0.25	0.35	0.45	0.55	0.65
lg_a	0.2240741	0.1907407	0.1796296	0.1592593	0.1740741
lg_b	0.2037037	0.1888889	0.1722222	0.1611111	0.1777778
lg_c	0.1944444	0.1814815	0.1703704	0.1574074	0.1629630

```
kable(accuracy_correctly_M, caption = "Accuracy in correctly
        classifying the malignant tumors for logistic model a, b, and
        c")
```

Table 3: Accuracy in correctly classifying the malignant tumors for logistic model a, b, and c

	0.25	0.35	0.45	0.55	0.65
lg_a	0.8674605	0.8199468	0.7853268	0.7176578	0.6308340

	0.25	0.35	0.45	0.55	0.65
lg_b	0.8674605	0.8164985	0.7768786	0.7147613	0.6139929
lg_c	0.8863186	0.8326473	0.7953579	0.7530457	0.6614719

v

Considering the overall error rate, the logistic model (c) has the smallest overall error rate among these three models. Generally speaking, the overall error rate decreases as threshold increases until threshold = 0.55, then it goes up a little bit when threshold = 0.65. Therefore, the best logistic model is (c) with threshold = 0.55 based on the overall error rate.

Considering the accuracy in correctly classifying the malignant tumors, the logistic model (c) has the largest accuracy in correctly classifying M among these three models. Generally speaking, accuracy decreases as threshold increases. Therefore, the best logistic model is (c) with threshold = 0.25 based on the accuracy in correctly classifying the malignant tumors.

2.2

i

ii

iii

- Fitting LDA and QDA models and identifying the prior probability and the mean parameter estimates. Since the we use the same dataset to train LDA and QDA model, prior probabilities and mean parameter estimates are the same in LDA and QDA.

```
library(MASS)

lda <- lda(diagnosis ~., wisc_train)
# lda$prior
# lda$means

qda <- qda(diagnosis ~., wisc_train)
# qda$prior
# qda$means

kable(lda$prior, caption = "prior probability of LDA")
```

Table 4: prior probability of LDA

	x
0	0.6277778
1	0.3722222

```
kable(qda$prior, caption = "prior probability of QDA")
```

Table 5: prior probability of QDA

	x
0	0.6277778
1	0.3722222

```
kable(lda$means, caption = "mean parameter estimates of LDA")
```

Table 6: mean parameter estimates of LDA

	log_texture_mean	log_smoothness_mean	log_compactness_mean
0	2.865239	-2.391700	-2.610107
1	3.055710	-2.282046	-1.988306

```
kable(qda$means, caption = "mean parameter estimates of QDA")
```

Table 7: mean parameter estimates of QDA

	log_texture_mean	log_smoothness_mean	log_compactness_mean
0	2.865239	-2.391700	-2.610107
1	3.055710	-2.282046	-1.988306

iv

- Using 10-fold cross-validation to determine which threshold should be used in LDA for predicting the response depending on the two classification performance metrics.

```
# use the same folds split in 1(iv)
LDAcv <- function(c) {
  overall_error_fold <- matrix(NA, 3, 10)
  correct_M_fold <- matrix(NA, 3, 10)
  for (i in 1:10) {
    train_cv <- wisc_train[-folds[[i]], ]
    test_cv <- wisc_train[folds[[i]], ]

    lda1 <- lda(diagnosis ~., train_cv)
    prob <- predict(lda1, test_cv)$posterior[, 2] #post prob of falling in y = 1
    pred <- ifelse(prob > c, 1, 0)
    CM <- table(pred, test_cv$diagnosis)
    correct_M <- CM[2,2] / sum(CM[, 2])
    overall_error <- mean(pred != test_cv$diagnosis)
    overall_error_fold[1, i] <- overall_error
    correct_M_fold[1, i] <- correct_M
```

```
    lda2 <- lda(diagnosis ~ log_texture_mean * log_smoothness_mean +
        log_texture_mean * log_compactness_mean +
        log_smoothness_mean : log_compactness_mean,
        train_cv)
    prob <- predict(lda2, test_cv)$posterior[, 2] #post prob of falling in y = 1
    pred <- ifelse(prob > c, 1, 0)
    CM <- table(pred, test_cv$diagnosis)
    correct_M <- CM[2,2] / sum(CM[, 2])
    overall_error <- mean(pred != test_cv$diagnosis)
    overall_error_fold[2, i] <- overall_error
    correct_M_fold[2, i] <- correct_M

    lda3 <- lda(diagnosis ~ log_texture_mean * log_smoothness_mean +
          log_texture_mean * log_compactness_mean +
          log_smoothness_mean : log_compactness_mean +
          I(log_texture_mean^2) + I(log_smoothness_mean^2) +
          I(log_compactness_mean^2), train_cv)
    prob <- predict(lda3, test_cv)$posterior[, 2] #post prob of falling in y = 1
    pred <- ifelse(prob > c, 1, 0)
    CM <- table(pred, test_cv$diagnosis)
    correct_M <- CM[2,2] / sum(CM[, 2])
    overall_error <- mean(pred != test_cv$diagnosis)
    overall_error_fold[3, i] <- overall_error
    correct_M_fold[3, i] <- correct_M
  }
  all_error <- rowMeans(overall_error_fold)
  correct_M <- rowMeans(correct_M_fold)

  return(c(all_error, correct_M))
}
c <- c(0.25, 0.35, 0.45, 0.55, 0.65)
error_corretM <- sapply(c, LDAcv)
colnames(error_corretM) = c(0.25, 0.35, 0.45, 0.55, 0.65)
overall_error_rate <- error_corretM[1:3, ]
rownames(overall_error_rate) = c("LDA_a", "LDA_b", "LDA_c")
# overall_error_rate

accuracy_correctly_M <- error_corretM[4:6, ]
rownames(accuracy_correctly_M) = c("LDA_a", "LDA_b", "LDA_c")
# accuracy_correctly_M

kable(overall_error_rate, caption = "Overall error rate for LDA model a,
                                                    b, and c")
```

Table 8: Overall error rate for LDA model a, b, and c

	0.25	0.35	0.45	0.55	0.65
LDA_a	0.2240741	0.1925926	0.1833333	0.1703704	0.1777778
LDA_b	0.2000000	0.1814815	0.1592593	0.1740741	0.1851852
LDA_c	0.1888889	0.1629630	0.1629630	0.1555556	0.1814815

```
kable(accuracy_correctly_M, caption = "Accuracy in correctly classifying the
                            malignant tumors for LDA model a, b, and c")
```

Table 9: Accuracy in correctly classifying the malignant tumors for LDA model a, b, and c

	0.25	0.35	0.45	0.55	0.65
LDA_a	0.8833653	0.8320896	0.7853268	0.7247984	0.6272319
LDA_b	0.8348914	0.7914024	0.7379582	0.6597630	0.5891540
LDA_c	0.8551917	0.8126524	0.7781922	0.7093634	0.6044411

v

- Repeated for QDA

```
QDAcv <- function(c) {
  overall_error_fold <- matrix(NA, 3, 10)
  correct_M_fold <- matrix(NA, 3, 10)
  for (i in 1:10) {
    train_cv <- wisc_train[-folds[[i]], ]
    test_cv <- wisc_train[folds[[i]], ]

    qda1 <- qda(diagnosis ~., train_cv)
    prob <- predict(qda1, test_cv)$posterior[, 2] #post prob of falling in y = 1
    pred <- ifelse(prob > c, 1, 0)
    CM <- table(pred, test_cv$diagnosis)
    correct_M <- CM[2,2] / sum(CM[, 2])
    overall_error <- mean(pred != test_cv$diagnosis)
    overall_error_fold[1, i] <- overall_error
    correct_M_fold[1, i] <- correct_M

    qda2 <- qda(diagnosis ~ log_texture_mean * log_smoothness_mean +
            log_texture_mean * log_compactness_mean +
            log_smoothness_mean : log_compactness_mean,
            train_cv)
    prob <- predict(qda2, test_cv)$posterior[, 2] #post prob of falling in y = 1
    pred <- ifelse(prob > c, 1, 0)
    CM <- table(pred, test_cv$diagnosis)
    correct_M <- CM[2,2] / sum(CM[, 2])
    overall_error <- mean(pred != test_cv$diagnosis)
    overall_error_fold[2, i] <- overall_error
    correct_M_fold[2, i] <- correct_M

    qda3 <- qda(diagnosis ~ log_texture_mean * log_smoothness_mean +
            log_texture_mean * log_compactness_mean +
            log_smoothness_mean : log_compactness_mean +
            I(log_texture_mean^2) + I(log_smoothness_mean^2) +
            I(log_compactness_mean^2), train_cv)
    prob <- predict(qda3, test_cv)$posterior[, 2] #post prob of falling in y = 1
    pred <- ifelse(prob > c, 1, 0)
```

```
    CM <- table(pred, test_cv$diagnosis)
    correct_M <- CM[2,2] / sum(CM[, 2])
    overall_error <- mean(pred != test_cv$diagnosis)
    overall_error_fold[3, i] <- overall_error
    correct_M_fold[3, i] <- correct_M
  }
  all_error <- rowMeans(overall_error_fold)
  correct_M <- rowMeans(correct_M_fold)
  return(c(all_error, correct_M))
}

c <- c(0.25, 0.35, 0.45, 0.55, 0.65)
error_corretM <- sapply(c, QDAcv)
colnames(error_corretM) = c(0.25, 0.35, 0.45, 0.55, 0.65)
overall_error_rate <- error_corretM[1:3, ]
rownames(overall_error_rate) = c("QDA_a", "QDA_b", "QDA_c")
# overall_error_rate
accuracy_correctly_M <- error_corretM[4:6, ]
rownames(accuracy_correctly_M) = c("QDA_a", "QDA_b", "QDA_c")
# accuracy_correctly_M
kable(overall_error_rate, caption = "Overall error rate for QDA model a, b
                                                                    , and c")
```

Table 10: Overall error rate for QDA model a, b , and c

	0.25	0.35	0.45	0.55	0.65
QDA_a	0.2074074	0.1759259	0.1703704	0.1648148	0.1611111
QDA_b	0.1962963	0.1944444	0.1814815	0.1851852	0.1759259
QDA_c	0.2018519	0.1962963	0.1907407	0.1759259	0.1722222

```
kable(accuracy_correctly_M, caption = "Accuracy in correctly classifying the
      malignant tumors for QDA model a, b, and c")
```

Table 11: Accuracy in correctly classifying the malignant tumors for QDA model a, b, and c

	0.25	0.35	0.45	0.55	0.65
QDA_a	0.8766203	0.8438973	0.7992041	0.7554037	0.6644344
QDA_b	0.8793067	0.8606084	0.8494655	0.8137279	0.8137279
QDA_c	0.8734138	0.8699655	0.8659655	0.8509536	0.8471075

vi

Logistic model (a) is closest to LDA as they both have linear decision boundaries;

Logistic model (c) is closest to QDA as they both have quadratic decision boundaries.

Logistic model (a) is closest to LDA (a) with both linear decision boundaries;

Logistic model(c), LDA(c), and QDA (a) are very close as they have similar quadratic decision boundaries.

2.3

- Used 10-fold cross-validation, construct curves for the (test) overall error rate and (test) accuracy in correctly classifying the malignant tumors as a function of 1/K.

i

```
KNNcv <- function(k) {
  overall_error_fold <- rep(NA, 10)
  correct_M_fold <- rep(NA, 10)
  for (i in 1:10) {
    train_cv <- wisc_train[-folds[[i]], ]
    test_cv <- wisc_train[folds[[i]], ]
    pred <- knn(train_cv[, 2:4], test_cv[, 2:4], train_cv$diagnosis,
    k)
    CM <- table(pred, test_cv$diagnosis)
    correct_M <- CM[2,2] / sum(CM[, 2])
    overall_error <- mean(pred != test_cv$diagnosis)
    overall_error_fold[i] <- overall_error
    correct_M_fold[i] <- correct_M
  }
  all_error <- mean(overall_error_fold)
  correct_M <- mean(correct_M_fold)
  return(c(all_error, correct_M))
}

K <- c(1, 2, 3, 5, 10, 20, 50, 100)
error_corretM <- sapply(K, KNNcv)
colnames(error_corretM) = c(1, 2, 3, 5, 10, 20, 50, 100)
rownames(error_corretM) = c("overall error rate", "accuracy in
correctly classifying the malignant tumors")

round_M <- round(error_corretM, 4)
kable(round_M, caption = "Overall error rate and accuracy in
correctly classifying M for KNN model with different K")
```

Table 12: Overall error rate and accuracy in correctly classifying M for KNN model with different K

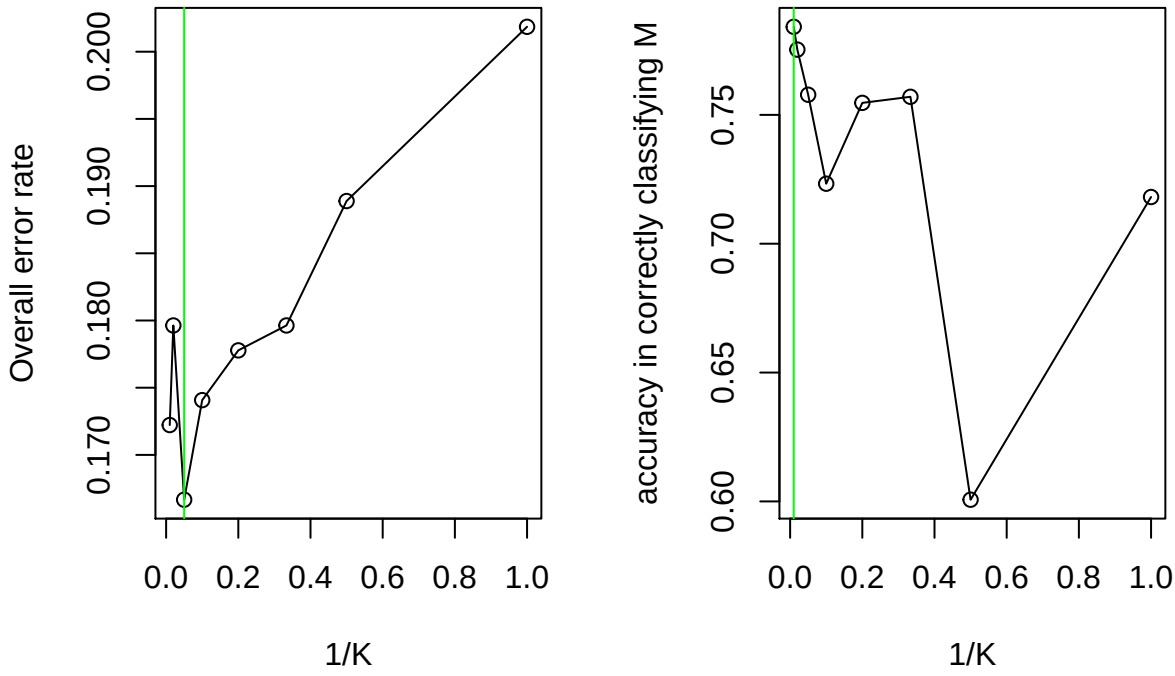
	1	2	3	5	10	20	50	100
overall error rate	0.2019	0.1889	0.1796	0.1778	0.1741	0.1667	0.1796	0.1722
accuracy in correctly classifying the malignant tumors	0.7181	0.6007	0.7570	0.7547	0.7233	0.7578	0.7753	0.7842

ii

- Use the previous curves to find the optimal Ks with the minimum (test) error rates and (test) accuracy in correctly classifying the malignant tumors.

```
par(mfrow = c(1,2))
kopt_er = K[which.min(error_corretM[1, ])]
plot(rev(1/K), rev(error_corretM[1, ]), xlab = "1/K", ylab =
      "Overall error rate")
lines(rev(1/K), rev(error_corretM[1, ]))
abline(v = 1/kopt_er, col = "green")

kopt_ac = K[which.max(error_corretM[2, ])]
plot(rev(1/K), rev(error_corretM[2, ]), xlab = "1/K", ylab =
      "accuracy in correctly classifying M")
lines(rev(1/K), rev(error_corretM[2, ]))
abline(v = 1/kopt_ac, col = "green")
```




```
kopt_er
```

```
## [1] 20
```

```
kopt_ac
```

```
## [1] 100
```

iii

- Used bootstrap to construct the 95% confidence intervals for the sensitivity and specificity of the model with the best overall error rate.

```
set.seed(4540)
  fct_ses_spe <- function(data, i) {
  pred <- knn(data[i, 2:4], data[, 2:4], data[i, 1], k = 20)
  CM <- table(pred, data[, 1])
  sensi <- CM[2,2] / sum(CM[, 2])
  speci <- 1 - (CM[2,1] / sum(CM[, 1]))
  return(c(sensi, speci))
  }

(boot_rep <- boot(wisc_train, fct_ses_spe, 1e2))
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = wisc_train, statistic = fct_ses_spe, R = 100)
##
##
## Bootstrap Statistics :
##      original        bias    std. error
## t1* 0.7661692 -0.0005472637  0.02901742
## t2* 0.8879056 -0.0064896755  0.02195490
```

```
(CI_sensi <- quantile(boot_rep$t[, 1], c(0.025, 0.975)))
```

```
##      2.5%     97.5%
## 0.7111940 0.8261194
```

```
(CI_speci <- quantile(boot_rep$t[, 2], c(0.025, 0.975)))
```

```
##      2.5%     97.5%
## 0.8362094 0.9160029
```

4

- Constructed the confusion matrices for the new test data using the logistic regression, LDA, QDA, and K-NN models with the best error rates. Based on the models

```
wisc_test <- read.csv("wisc_eval.csv")
wisc_test$diagnosis <- ifelse(wisc_test$diagnosis == "M", 1, 0)

lg3 <- glm(diagnosis ~ log_texture_mean * log_smoothness_mean +
      log_texture_mean * log_compactness_mean +
```

```
      log_smoothness_mean : log_compactness_mean +
      I(log_texture_mean^2) + I(log_smoothness_mean^2) +
      I(log_compactness_mean^2), family = binomial,
      wisc_train)
prob_lg3 <- predict(lg3, wisc_test, type = "response")
pred_lg3 <- ifelse(prob_lg3 > 0.55, 1, 0)
(CM <- table(pred_lg3, wisc_test$diagnosis))
```

```
##
## pred_lg3  0  1
##        0 15  6
##        1  3  5
```

```
sensi <- CM[2,2] / sum(CM[, 2])
names(sensi) <- c("sensitivity")
sensi
```

```
## sensitivity
##   0.4545455
```

```
speci <- 1 - CM[2,1] / sum(CM[, 1])
names(speci) <- c("specificity")
speci
```

```
## specificity
##   0.8333333
```

```
lda <- lda(diagnosis ~ log_texture_mean * log_smoothness_mean +
        log_texture_mean * log_compactness_mean +
        log_smoothness_mean : log_compactness_mean +
        I(log_texture_mean^2) + I(log_smoothness_mean^2) +
        I(log_compactness_mean^2), wisc_train)

prob <- predict(lda, wisc_test)$posterior[, 2]
pred_lda <- ifelse(prob > 0.55, 1, 0)
(CM <- table(pred_lda, wisc_test$diagnosis))
```

```
##
## pred_lda  0  1
##        0 15  6
##        1  3  5
```

```
sensi <- CM[2,2] / sum(CM[, 2])
names(sensi) <- c("sensitivity")
sensi
```

```
## sensitivity
##   0.4545455
```

```
speci <- 1 - CM[2,1] / sum(CM[, 1])
names(speci) <- c("specificity")
speci
```

```
## specificity
##   0.8333333
```

```
qda <- qda(diagnosis ~., wisc_train)
prob <- predict(qda, wisc_test)$posterior[, 2]
pred_qda <- ifelse(prob > 0.65, 1, 0)
(CM <- table(pred_qda, wisc_test$diagnosis))
```

```
##
## pred_qda  0  1
##        0 17  6
##        1  1  5
```

```
sensi <- CM[2,2] / sum(CM[, 2])
names(sensi) <- c("sensitivity")
sensi
```

```
## sensitivity
##   0.4545455
```

```
speci <- 1 - CM[2,1] / sum(CM[, 1])
names(speci) <- c("specificity")
speci
```

```
## specificity
##   0.9444444
```

```
pred_knn <- knn(wisc_train[, 2:4], wisc_test[, 2:4],
      wisc_train$diagnosis, 20)
(CM_knn <- table(pred_knn, wisc_test$diagnosis))
```

```
##
## pred_knn  0  1
##        0 15  6
##        1  3  5
```

```
sensi <- CM_knn[2,2] / sum(CM_knn[, 2])
names(sensi) <- c("sensitivity")
sensi
```

```
## sensitivity
##   0.4545455
```

```r
speci <- 1 - CM_knn[2,1] / sum(CM_knn[, 1])
names(speci) <- c("specificity")
speci
```

```
## specificity
##   0.8333333
```