MAST90138 Assignment 3

867492 Haonan Zhong

Question 1a

```r
library(MASS)
suppressMessages(library(pls))
XGtrain <- read.table("XGtrainRain.txt", header = TRUE, sep = ",")
XGtest <- read.table("XGtestRain.txt", header = TRUE, sep = ",")
```

```r
# Fitting the quadratic discriminant model
# qda_model <- qda(G~., data = XGtrain)
# Fitting the logistic regression model
logistic <- glm(G~., data = XGtrain, family = binomial(link = "logit"))
```

```
## Warning: glm.fit: algorithm did not converge
```

```r
# summary(logistic)
```

When attempting to fit the quadratic discriminant model with all the p predictors in the training set, an error was shown that some groups are too small for fitting, and from the summary of the logistic regression fitted with all p predictors, the model is overfitted as it has zero degrees of freedom. And only 149 of the 365 explanatory variables were used to fit the model, mainly because we only have 150 instances in the training set, thus there are insufficient degrees of freedom to fit all p predictors. Therefore, it is not recommended to use these two classifiers on the test set.

Question 1b

```r
# Obtain the projection matrix for PCA
Xtrain <- scale(XGtrain[, -c(366)], scale=FALSE)
PCX <- prcomp(Xtrain, retx = T)
gamma <- PCX$rotation
# Manually re-compute the PC components
Y <- (Xtrain - matrix(rep(1, nrow(Xtrain)), nrow=nrow(Xtrain)) %*% colMeans(Xtrain)) %*% gamma
all(PCX$x == Y)
```

```
## [1] TRUE
```

```r
# Obtain the projection matrix for PLS
PLS <- plsr(G~., data = XGtrain)
phi <- PLS$projection
t <- Xtrain %*% phi
all(PLS$scores == t)
```

```
## [1] TRUE
```

As we can see from the output above, all the manually computed components are the same as the one outputed by the R function.

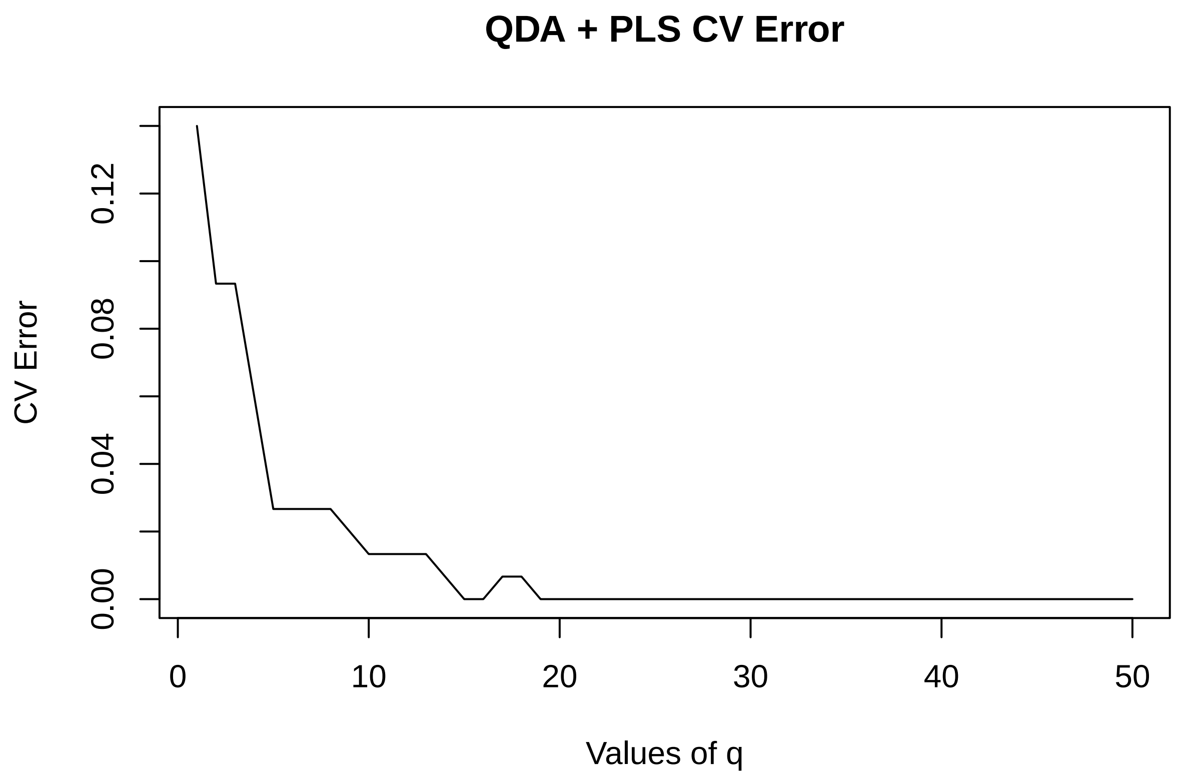
Question 1c

```r
# Train QDA with PLS components
CV_error <- rep(0, 50)
Gtrain <- XGtrain[, 366]

for (q in 1:50) {
  prediction <- c()
  for (i in 1:dim(XGtrain)[1]) {
    GDATACV <- Gtrain[-i]
    YDATACV <- as.data.frame(PLS$scores[-i, 1:q])
    QDA <- qda(GDATACV~., data = YDATACV)

    new_data <- as.data.frame(t(PLS$scores[i, 1:q]))
    colnames(new_data) <- colnames(YDATACV)
    prediction[i] <- as.numeric(predict(QDA, newdata = new_data)$class) - 1
  }
  CV_error[q] <- sum(prediction != Gtrain)/dim(XGtrain)[1]
}

# Plotting the cross validation error
plot(c(1:50), CV_error, type = "l", xlab = "Values of q", ylab = "CV Error", main = "QDA + PLS CV Error
```



```
which(CV_error == min(CV_error))
```

```
##  [1] 15 16 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
## [26] 42 43 44 45 46 47 48 49 50
```

```
min(CV_error)
```

```
## [1] 0
```

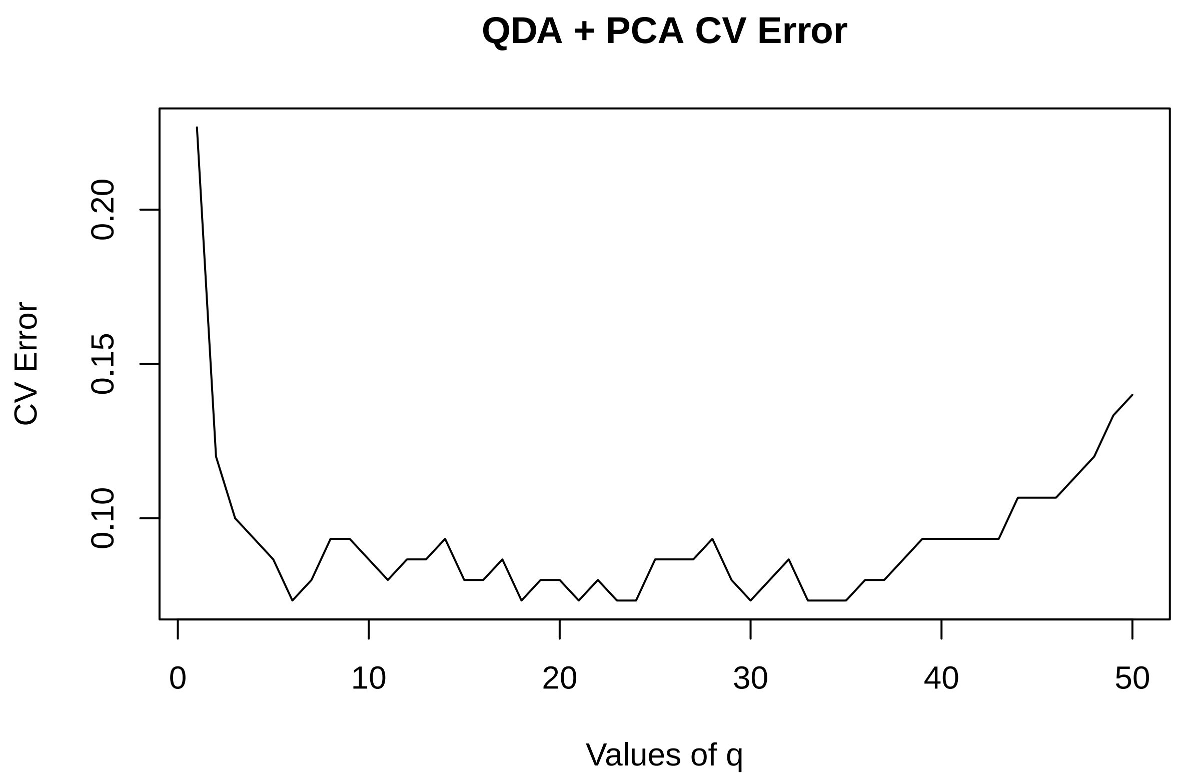
As we can see from the result above, cross validation error are equal to zero and lowest when $q = 15$. Although there are numerous values for q that leads to the lowest error, but in order to keep the model simple, thus, the chosen number of components for PLS should be $q = 15$.

```
# Train QDA with PCA components
CV_error <- rep(0, 50)
for (q in 1:50) {
  prediction <- c()
  for (i in 1:dim(XGtrain)[1]) {
    GDATACV <- Gtrain[-i]
    YDATACV <- as.data.frame(PCX$x[-i, 1:q])
    QDA <- qda(GDATACV~., data = YDATACV)

    new_data <- as.data.frame(t(PCX$x[i, 1:q]))
    colnames(new_data) <- colnames(YDATACV)
```

```r
    prediction[i] <- as.numeric(predict(QDA, newdata = new_data)$class) - 1
  }
  CV_error[q] <- sum(prediction != Gtrain)/dim(XGtrain)[1]
}

# Plotting the cross validation error
plot(c(1:50), CV_error, type = "l", xlab = "Values of q", ylab = "CV Error", main = "QDA + PCA CV Error
```



```r
which(CV_error == min(CV_error))
```

```
## [1]  6 18 21 23 24 30 33 34 35
```

```r
min(CV_error)
```

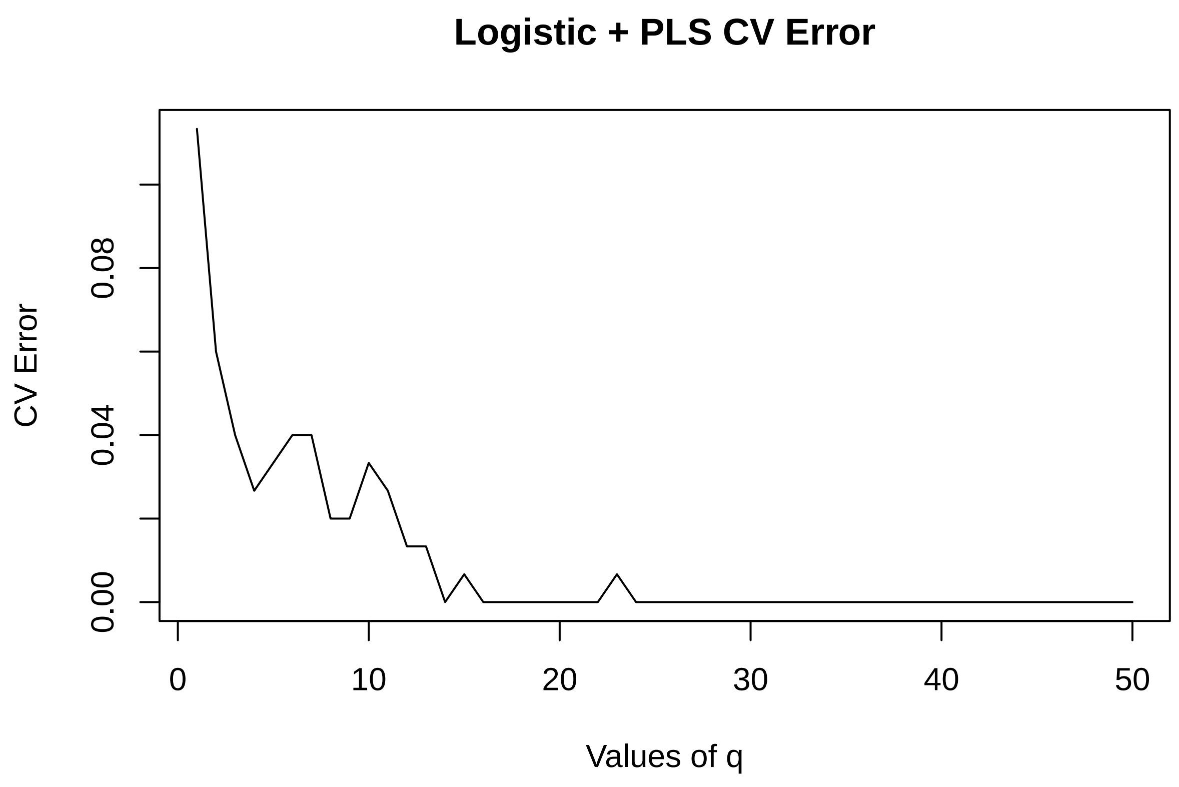
```
## [1] 0.07333333
```

As we can see from the output above, cross validation error are equal to 0.073 and lowest when $q = 6$. Although there are numerous values for q that leads to the lowest error, but in order to keep the model simple, thus, the chosen number of components for PCA should be $q = 6$.

```r
# Train logistic regression model with PLS
CV_error <- rep(0, 50)
for (q in 1:50) {
  prediction <- c()
  for (i in 1:dim(XGtrain)[1]) {
    GDATACV <- Gtrain[-i]
    YDATACV <- as.data.frame(PLS$scores[-i, 1:q])
    suppressWarnings(logistic <- glm(GDATACV~., data = YDATACV, family = binomial(link = "logit")))

    new_data <- as.data.frame(t(PLS$scores[i, 1:q]))
    colnames(new_data) <- colnames(YDATACV)
    prediction[i] <- ifelse(predict(logistic, newdata = new_data) > 0, 1, 0)
  }
  CV_error[q] <- sum(prediction != Gtrain)/dim(XGtrain)[1]
}

# Plotting the cross validation error
plot(c(1:50), CV_error, type = "l", xlab = "Values of q", ylab = "CV Error", main = "Logistic + PLS CV I
```



```
which(CV_error == min(CV_error))
```

```
##  [1] 14 16 17 18 19 20 21 22 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
## [26] 41 42 43 44 45 46 47 48 49 50
```

```
min(CV_error)
```

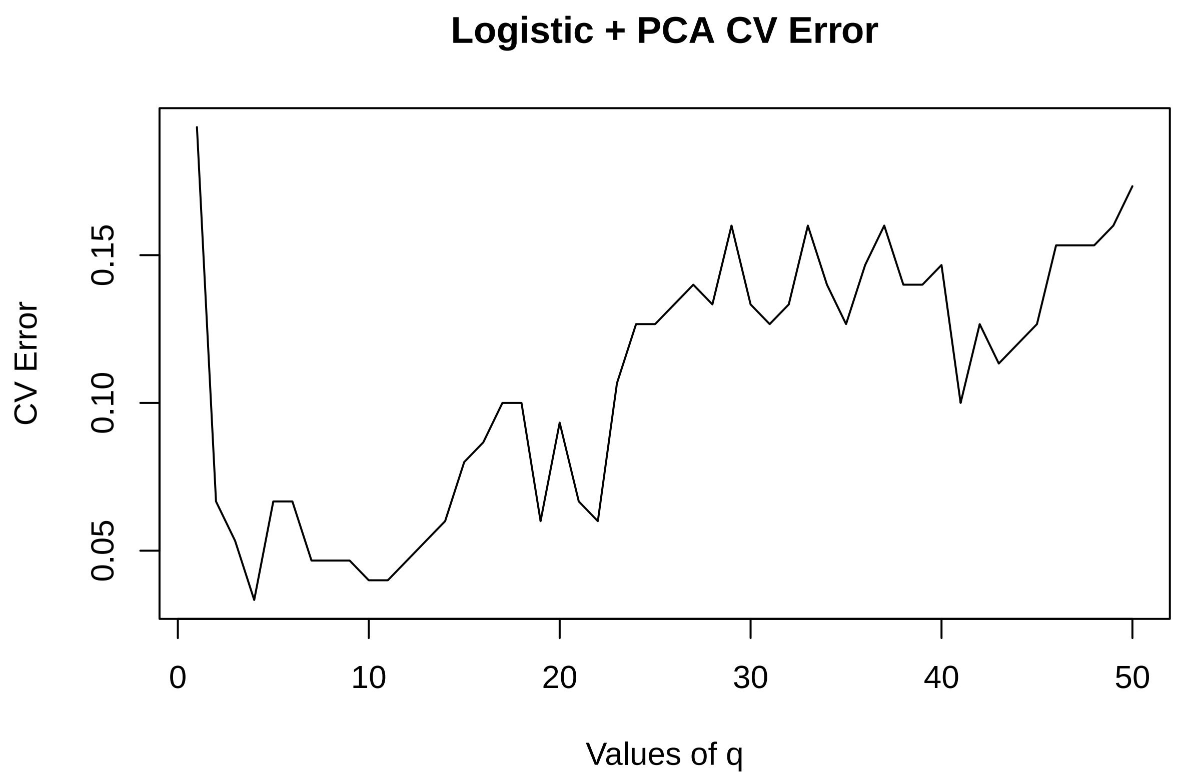
```
## [1] 0
```

As we can see from the result above, cross validation error are equal to zero and lowest when $q = 14$. Although there are numerous values for q that leads to the lowest error, but in order to keep the model simple, thus, the chosen number of components for PLS should be $q = 14$.

```r
# Train logistic regression model with PCA components
CV_error <- rep(0, 50)
for (q in 1:50) {
  prediction <- c()
  for (i in 1:dim(XGtrain)[1]) {
    GDATACV <- Gtrain[-i]
    YDATACV <- as.data.frame(PCX$x[-i, 1:q])
    suppressWarnings(logistic <- glm(GDATACV~., data = YDATACV, family = binomial(link = "logit")))

    new_data <- as.data.frame(t(PCX$x[i, 1:q]))
    colnames(new_data) <- colnames(YDATACV)
    prediction[i] <- ifelse(predict(logistic, newdata = new_data) > 0, 1, 0)
  }
  CV_error[q] <- sum(prediction != Gtrain)/dim(XGtrain)[1]
}

# Plotting the cross validation error
plot(c(1:50), CV_error, type = "l", xlab = "Values of q", ylab = "CV Error", main = "Logistic + PCA CV 
```



```r
which(CV_error == min(CV_error))
```

```
## [1] 4
```

```
min(CV_error)
```

```
## [1] 0.03333333
```

As we can see from the result above, cross validation error are equal to 0.033 and lowest when $q = 4$. Thus, the chosen number of components for PCA should be $q = 4$.

Question 1d

Based on the results of question 1c, we can see that for both quadratic discriminant model and logistic regression model, with the optimal q, PLS tend to give the lowest cross validation error equals to 0, which is better than each of the classifiers with PCA. However, with considerations of computational cost and issue of overfitting, PCA version is preferred.

Question 1e

```
# Retrain all the classifiers with the suggested value of q and the full training set
QDA_PLS <- qda(Gtrain~., data = as.data.frame(PLS$scores[, 1:15]))
QDA_PCA <- qda(Gtrain~., data = as.data.frame(PCX$x[, 1:6]))

logistic_PLS <- glm(Gtrain~., data = as.data.frame(PLS$scores[, 1:14]), family = binomial(link = "logit
logistic_PCA <- glm(Gtrain~., data = as.data.frame(PCX$x[, 1:4]), family = binomial(link = "logit"))
```

```
# Prepare the test set
Xtest <- XGtest[,-c(366)]
Gtest <- XGtest[, 366]
repbarX <- matrix(rep(colMeans(XGtrain), dim(Xtest)[1]), nrow = dim(Xtest)[1], byrow = T)
Y_new <- as.matrix(Xtest - repbarX) %*% gamma

# Report test error for quadratic discriminant with PCA components
PCA_newdata <- as.data.frame(Y_new[, 1:6])
prediction <- predict(QDA_PCA, newdata = PCA_newdata)$class
QDA_PCA_ERROR <- sum(prediction != Gtest)/dim(Xtest)[1]
paste("Test error for quadratic discriminant with PCA components is", QDA_PCA_ERROR)
```

```
## [1] "Test error for quadratic discriminant with PCA components is 0.0975609756097561"
```

```
# Report test error for logistic regression with PCA components
PCA_newdata <- as.data.frame(Y_new[, 1:4])
prediction <- ifelse(predict(logistic_PCA, newdata = PCA_newdata) > 0, 1, 0)
LR_PCA_ERROR <- sum(prediction != Gtest)/dim(Xtest)[1]
paste("Test error for logistic regression with PCA components is", LR_PCA_ERROR)
```

```
## [1] "Test error for logistic regression with PCA components is 0.024390243902439"
```

```
# Report test error for quadratic discriminant with PLS components
t_new <- as.matrix(Xtest - repbarX) %*% phi
PLS_newdata <- as.data.frame(t_new[, 1:15])
prediction <- predict(QDA_PLS, newdata = PLS_newdata)$class
QDA_PLS_ERROR <- sum(prediction != Gtest)/dim(Xtest)[1]
paste("Test error for quadratic discriminant with PLS components is", QDA_PLS_ERROR)
```

```
## [1] "Test error for quadratic discriminant with PLS components is 0.0975609756097561"
```

```r
# Report test error for logistic regression with PLS components
PLS_newdata <- as.data.frame(t_new[, 1:14])
prediction <- ifelse(predict(logistic_PLS, newdata = PLS_newdata) > 0, 1, 0)
LR_PLS_ERROR <- sum(prediction != Gtest)/dim(Xtest)[1]
paste("Test error for logistic regression with PLS components is", LR_PLS_ERROR)
```

```
## [1] "Test error for logistic regression with PLS components is 0.121951219512195"
```

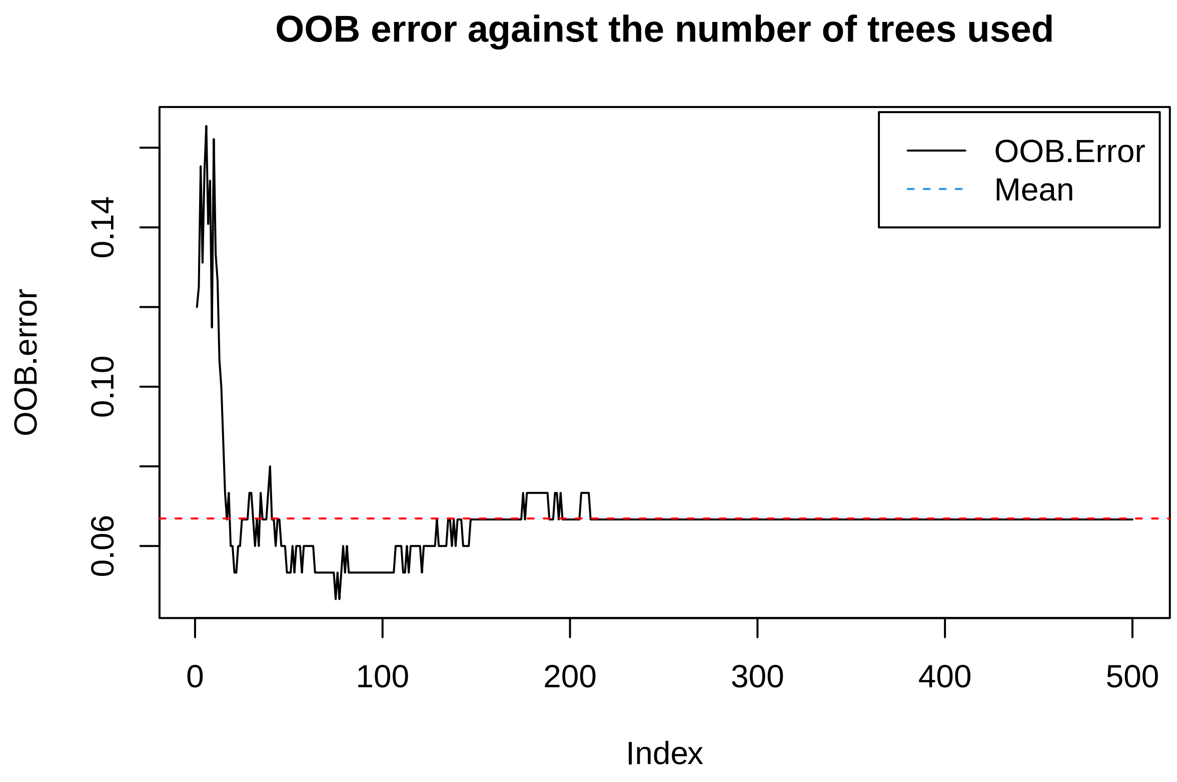
As we can see from the test errors for the four classifiers, quadratic discriminant produces similar test error with both PCA and PLS component. Whilst logistic regression with PCA component produces a test error that's much smaller than logistic regression with PLS. And it is worth mentioning that for both quadratic discriminant and logistic regression, they tend to use less number of PCA components, while producing similar or better test error than PLS components.

Question 2a

```r
suppressMessages(library(randomForest))

set.seed(90139)

rainfall.rf <- randomForest(factor(G) ~., data = XGtrain, importance = TRUE)
OOB.error <- rainfall.rf$err.rate[,1]
plot(OOB.error, type = "l")
abline(h = mean(OOB.error), col = 'red', xlab = "number of trees used", lty = 2)
title("OOB error against the number of trees used")
legend("topright", c('OOB.Error','Mean'), col=c('black','4'), lty=c(1,2), inset=.01)
```



```
which(OOB.error == OOB.error[500])
```

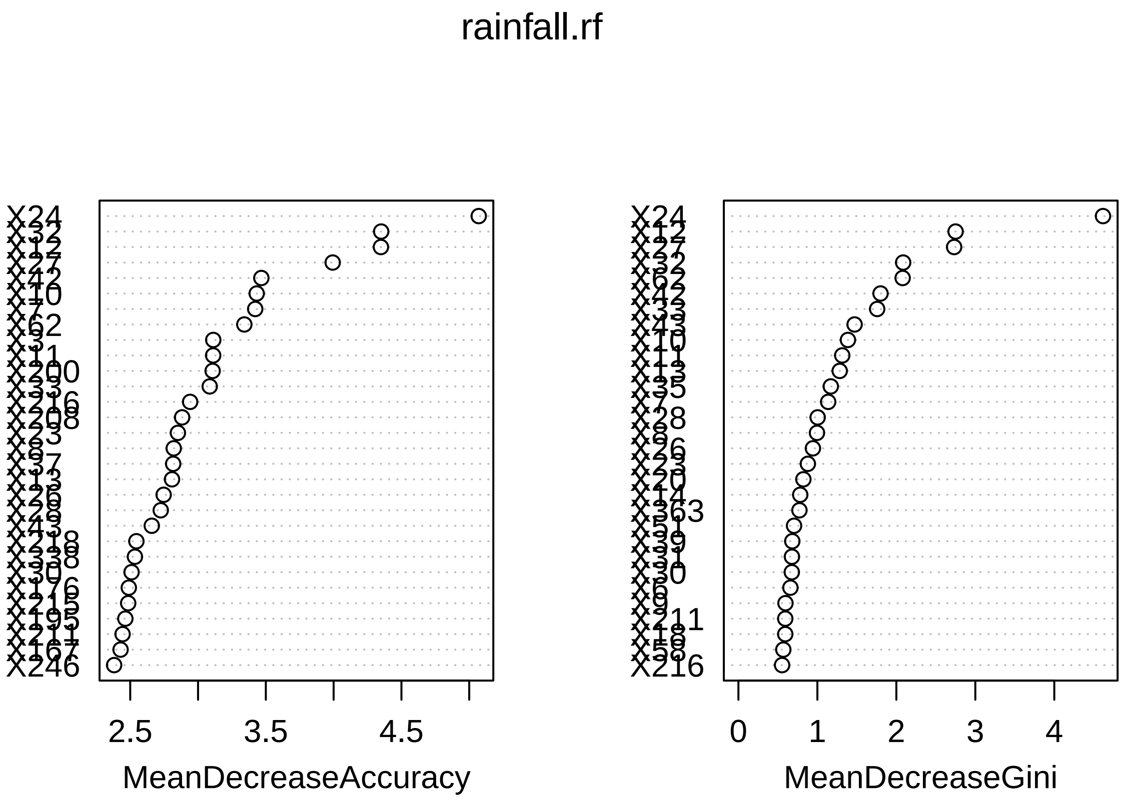
```
##   [1]  17  25  26  27  28  31  33  36  37  38  41  42  44  45 129 135 136 138
##  [19] 140 141 142 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
##  [37] 162 163 164 165 166 167 168 169 170 171 172 173 174 176 189 190 191 194
##  [55] 196 197 198 199 200 201 202 203 204 205 211 212 213 214 215 216 217 218
##  [73] 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236
##  [91] 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254
## [109] 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272
## [127] 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290
## [145] 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308
## [163] 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326
## [181] 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344
## [199] 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362
## [217] 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
## [235] 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398
## [253] 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416
## [271] 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434
## [289] 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452
## [307] 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470
## [325] 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488
## [343] 489 490 491 492 493 494 495 496 497 498 499 500
```

As we can see, as the number of trees increases, the random forest becomes more stable. The out-of-bag error begins to stabilize when number of trees equals to 211. Therefore, the choice of number of tree here will be 211.

```r
# Refit the random forest classifier using the chosen number of tree
rainfall.rf <- randomForest(factor(G)~., data = XGtrain, importance = TRUE, ntree = 211)
```

Question 2b

```r
varImpPlot(rainfall.rf)
```



As we can see from the output plots above, most of the variables/days are primarily summer days in Australia, in which may reflect the fact of the large difference of rainfall during summer in the north and south of the country.

Question 2c

```r
test_prediction <- predict(rainfall.rf, XGtest[,1:365])
RF_ERROR <- sum(test_prediction != XGtest$G) / length(XGtest$G)
paste("Classification error:", RF_ERROR)
```

```
## [1] "Classification error: 0.048780487804878"
```

```r
# Try refitting and running the tree 10 times
for (i in 1:10) {
  rainfall.rf <- randomForest(factor(G)~., data = XGtrain, importance = TRUE, ntree = 211)
  test_prediction <- predict(rainfall.rf, XGtest[,1:365])
  RF_ERROR_1 <- sum(test_prediction != XGtest$G) / length(XGtest$G)
  print(paste("Classification error:", RF_ERROR_1))
}
```

```
## [1] "Classification error: 0.0731707317073171"
## [1] "Classification error: 0.0975609756097561"
## [1] "Classification error: 0.0975609756097561"
## [1] "Classification error: 0.0731707317073171"
## [1] "Classification error: 0.0975609756097561"
## [1] "Classification error: 0.048780487804878"
## [1] "Classification error: 0.024390243902439"
## [1] "Classification error: 0.0975609756097561"
## [1] "Classification error: 0.048780487804878"
## [1] "Classification error: 0.0975609756097561"
```

As we can see from the output above, we don't get the same classification error every time, which is primarily due to the process of constructing the trees. Randomness was introduced each time when constructing a tree, where bootstrap samples were randomly drawn from the training set. Furthermore, m features were randomly selected from p features as split candidates, which may result in different trees, thus, reduces the correlation between the trees in a forest.

To make random forest for stable, one may increases the number of split candidates when constructing a tree in which reduces the randomness and increases the chance of picking the same split feature. Moreover, one may select large number of trees when constructing a random forest to reduce the randomness of bootstrap sampling.

Question 3

```
rbind(LR_PCA_ERROR, LR_PLS_ERROR, QDA_PCA_ERROR, QDA_PLS_ERROR, RF_ERROR)
```

```
##                     [,1]
## LR_PCA_ERROR  0.02439024
## LR_PLS_ERROR  0.12195122
## QDA_PCA_ERROR 0.09756098
## QDA_PLS_ERROR 0.09756098
## RF_ERROR      0.04878049
```

As we can see from the percentage of misclassification for each of the five classifier. We can see that logistic regression with PCA produces the smallest error 2.44%, followed by random forest at 4.88%, and both QDA model shares the same error rate, while logistic regression with PLS produces the highest error rate among all at 12.20%. For logistic regression, PCA perform better than PLS might be due the samples from two class became linearly separable after applying PCA allowing logistic regression to yield better performance. Moreover, the reason that QDA did not perform well could be due to it assumes the samples comes from a normal distribution while the data is not. Lastly, the error rate of random forest tends to fluctuate a lot, which is primarily due to the randomness when training a forest.