# Problem Set 2

Akira Masuda

2020/2/2

```
Course: MACS30100 Perspectives on Computational Modeling (Winter 2020)
Author: Akira Masuda (ID: alakira)
```

---

## The Bayes Classifier

### 1.

#### a. Set seed

```
# Set seed
set.seed(110)
```

#### b. Simulate a dataset

```
x_1 = runif(200, -1, 1)
x_2 = runif(200, -1, 1)
```

#### c. Calculate Y

```
ep = rnorm(200, 0, 0.25)
y = x_1 + x_1^2 + x_2 + x_2^2 + ep
```

#### d. Calculate the probability of success

The log-odds is calculated by the following equation.

$$\log\left(\frac{Pr(success)}{1 - Pr(success)}\right) = X_1 + X_1^2 + X_2 + X_2^2 + \epsilon$$

By transforming the equation, the probability of success is:

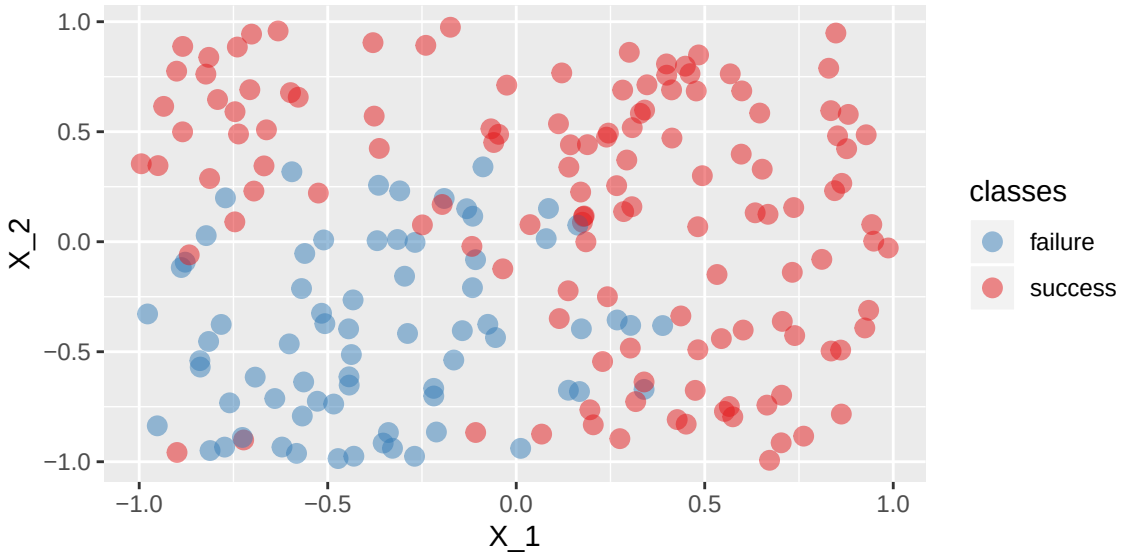
$$Pr(success) = \frac{e^{X_1 + X_1^2 + X_2 + X_2^2 + \epsilon}}{1 + e^{X_1 + X_1^2 + X_2 + X_2^2 + \epsilon}} = \frac{e^Y}{1 + e^Y}$$

```
pr <- exp(y) / (1 + exp(y))
```

#### e. Plot each point from the dataset

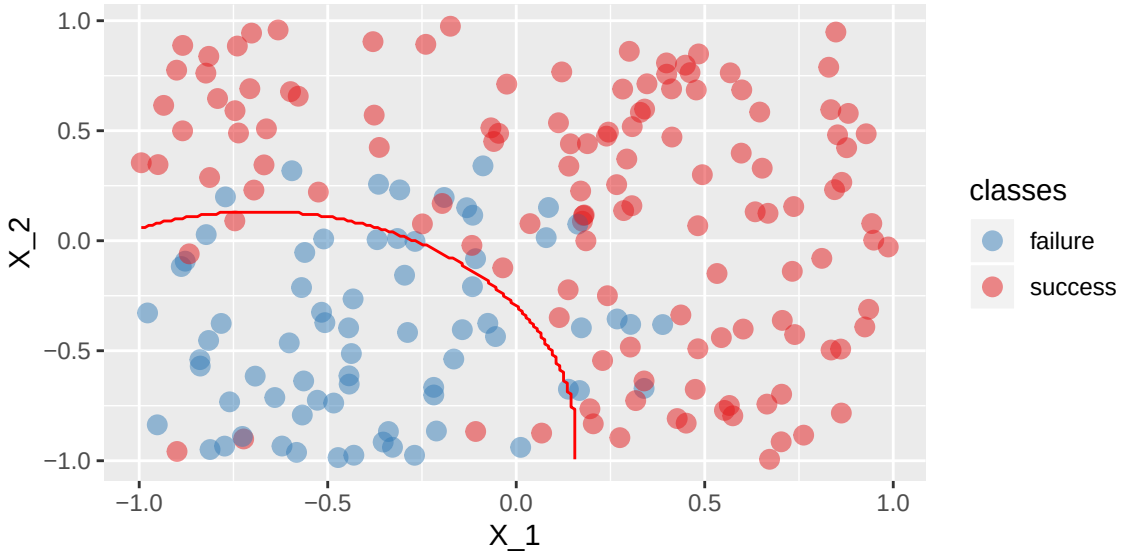
```
# Dataframe
df <- data.frame(X_1 = x_1, X_2 = x_2, Pr = pr)
df <- df %>%
  mutate(cl = case_when(pr > 0.5 ~ 'success',
                        pr <=  0.5 ~ 'failure'),
         cl_n = case_when(pr > 0.5 ~ 1,
```

```
                          pr <= 0.5 ~0))
twoClassColor <- brewer.pal(3,'Set1')[1:2]
names(twoClassColor) <- c('success','failure')

df %>%
  ggplot(aes(x=X_1, y=X_2)) +
  geom_point(aes(color = cl), size = 3, alpha = .5) +
  scale_colour_manual(name = 'classes', values = twoClassColor)
```



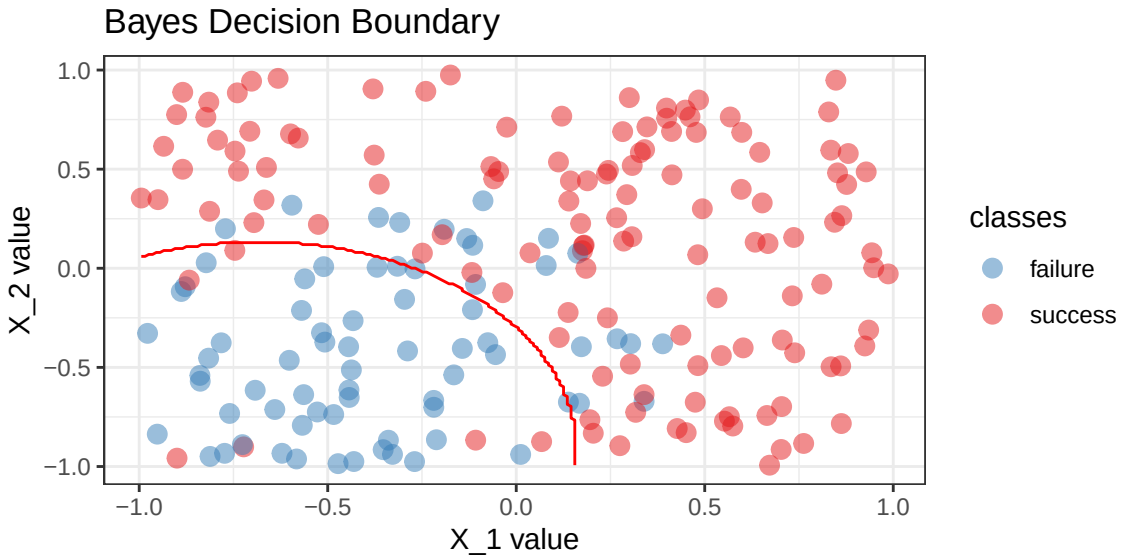

**f. Overlay with Bayes decision boundary**

```
# Refering to  http://www.cmap.polytechnique.fr/~lepennec/R/Learning/Learning.html
V <- 10
T <- 4
TrControl <- trainControl(method = "repeatedcv",
                          number = V,
                          repeats = T)
df_model <- df %>%
  dplyr::select(X_1, X_2, cl)
nbp = 200
Pred1 <- seq(min(df_model$X_1), max(df_model$X_1), length = nbp)
Pred2 <- seq(min(df_model$X_2), max(df_model$X_2), length = nbp)
Grid <- expand.grid(X_1 = Pred1, X_2 = Pred2)

Model <- train(data=df_model, cl ~ ., method = "nb", trControl = TrControl,
               tuneGrid = data.frame(usekernel = c(FALSE), fL = c(0), adjust = c(1)))
Pred <- predict(Model, newdata = Grid)

df_model %>%
  ggplot(aes(x=X_1, y=X_2)) +
  geom_point(aes(color = cl), size = 3, alpha = .5) +
  geom_contour(data = cbind(Grid, classes = Pred),
               aes(z = as.numeric(classes)),
               color = "red", breaks = c(1.5)) +
scale_colour_manual(name = 'classes', values = twoClassColor)
```




**g. Title and axis labels & h. Colored Background**

```
df_model %>%
  ggplot(aes(x=X_1, y=X_2)) +
  geom_point(aes(color = cl), size = 3, alpha = .5) +
  geom_contour(data = cbind(Grid, classes = Pred),
               aes(z = as.numeric(classes)),
               color = "red", breaks = c(1.5)) +
  scale_colour_manual(name = 'classes', values = twoClassColor) +
  ggtitle("Bayes Decision Boundary") +
  xlab('X_1 value') +
  ylab('X_2 value') +
  theme_bw()
```



## Exploring Simulated Differences between LDA and QDA

### 2. In case Bayes decision boundary is linear

#### a. Repeat simulation 1,000 times

```
train_lda2 = c()
test_lda2 = c()
train_qda2 = c()
test_qda2 = c()

for (i in 1:1000) {
  # i data generation
  x_1 = runif(1000, -1, 1)
  x_2 = runif(1000, -1, 1)
  ep = rnorm(1000, 0, 0.25)
  y = case_when(x_1 + x_2 + ep >= 0 ~ TRUE,
                x_1 + x_2 + ep < 0 ~ FALSE)
  df2 <- data.frame(x_1, x_2, y)

  # ii split
  split <- initial_split(df2, prop = .7)
  train <- training(split)
  test <- testing(split)

  # iii training
  lda2 <- MASS::lda(y ~ x_1 + x_2, data = train)
  qda2 <- MASS::qda(y ~ x_1 + x_2, data = train)

  # iv error
  train_lda2 = c(train_lda2, sum(predict(lda2, train)$class != train$y) / 700)
  test_lda2 = c(test_lda2, sum(predict(lda2, test)$class != test$y) / 300)
  train_qda2 = c(train_qda2, sum(predict(qda2, train)$class != train$y) / 700)
  test_qda2 = c(test_qda2, sum(predict(qda2, test)$class != test$y) / 300)
}
```

#### b. Summarize the results

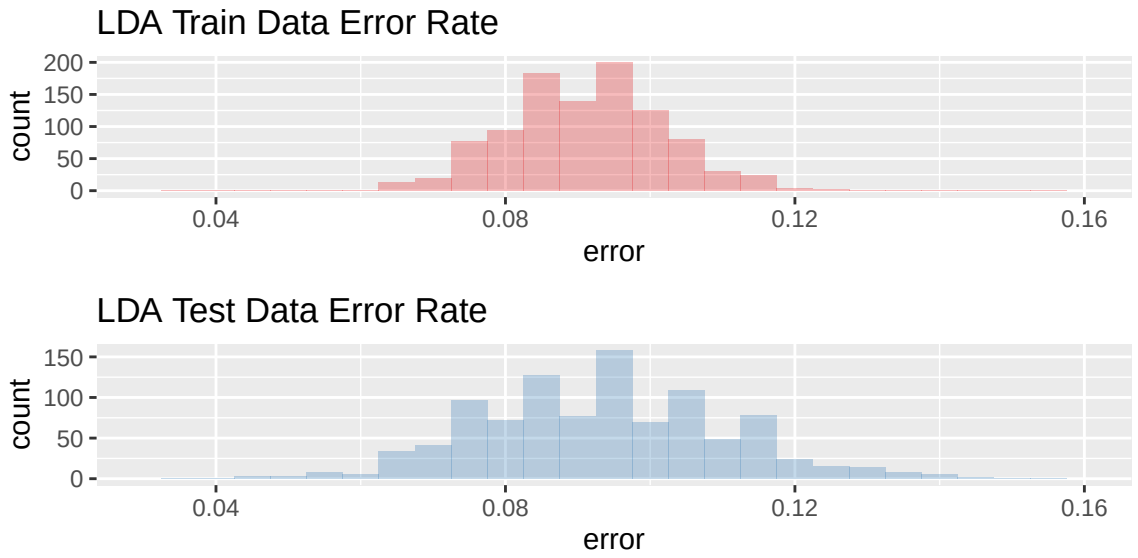
```
df2_result <- data.frame(train_lda2, test_lda2, train_qda2, test_qda2)
df2_result <- df2_result %>%
  gather(dset, error)
df2_result %>%
  group_by(dset) %>%
  summarise(mean = mean(error), sd = sd(error))
```

| dset | mean | sd |
|---|---|---|
| test_lda2 | 0.0934167 | 0.0168609 |
| test_qda2 | 0.0938367 | 0.0169069 |
| train_lda2 | 0.0911400 | 0.0109330 |
| train_qda2 | 0.0909057 | 0.0110357 |

```
lda_p1 <- df2_result %>%
  ggplot(aes(x=error)) +
  scale_colour_manual(name = 'classes', values = twoClassColor) +
```

```
  geom_histogram(data=subset(df2_result, dset == 'train_lda2'),
                 fill = brewer.pal(3,'Set1')[1],
                 alpha = 0.3, binwidth = 0.005) +
  scale_x_continuous(limits = c(0.03,0.16)) +
  ggtitle("LDA Train Data Error Rate")

lda_p2 <- df2_result %>%
  ggplot(aes(x=error)) +
  scale_colour_manual(name = 'classes', values = twoClassColor) +
  geom_histogram(data=subset(df2_result, dset == 'test_lda2'),
                 fill = brewer.pal(3,'Set1')[2],
                 alpha = 0.3, binwidth = 0.005) +
  scale_x_continuous(limits = c(0.03,0.16)) +
  ggtitle("LDA Test Data Error Rate")

grid.arrange(lda_p1, lda_p2, nrow = 2)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).

## Warning: Removed 2 rows containing missing values (geom_bar).
```

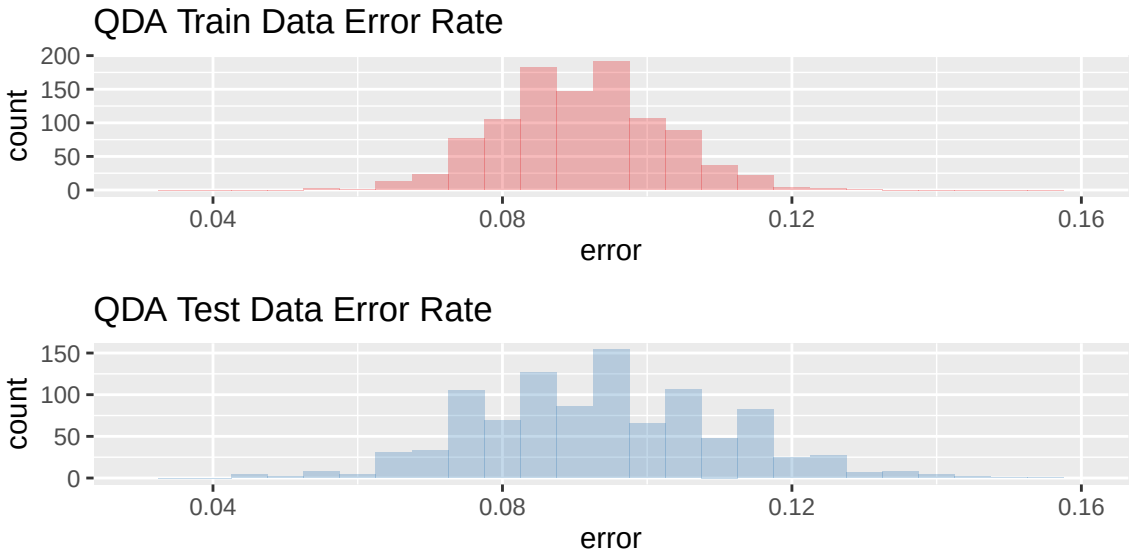



```
qda_p1 <- df2_result %>%
  ggplot(aes(x=error)) +
  scale_colour_manual(name = 'classes', values = twoClassColor) +
  geom_histogram(data=subset(df2_result, dset == 'train_qda2'),
                 fill = brewer.pal(3,'Set1')[1],
                 alpha = 0.3, binwidth = 0.005) +
  scale_x_continuous(limits = c(0.03,0.16)) +
  ggtitle("QDA Train Data Error Rate")

qda_p2 <- df2_result %>%
  ggplot(aes(x=error)) +
  scale_colour_manual(name = 'classes', values = twoClassColor) +
  geom_histogram(data=subset(df2_result, dset == 'test_qda2'),
                 fill = brewer.pal(3,'Set1')[2],
```

```
                alpha = 0.3, binwidth = 0.005) +
  scale_x_continuous(limits = c(0.03,0.16)) +
  ggtitle("QDA Test Data Error Rate")

grid.arrange(qda_p1, qda_p2, nrow = 2)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```




**3. In case Bayes decision boundary is non-linear**

```
train_lda2 = c()
test_lda2 = c()
train_qda2 = c()
test_qda2 = c()

for (i in 1:1000) {
  # i data generation
  x_1 = runif(1000, -1, 1)
  x_2 = runif(1000, -1, 1)
  ep = rnorm(1000, 0, 0.25)
  y = case_when(x_1 + x_1^2 + x_2 + x_2^2 + ep >= 0 ~ TRUE,
                x_1 + x_1^2 + x_2 + x_2^2 + ep < 0 ~ FALSE)
  df2 <- data.frame(x_1, x_2, y)

  # ii split
  split <- initial_split(df2, prop = .7)
  train <- training(split)
  test <- testing(split)

  # iii training
  lda2 <- MASS::lda(y ~ x_1 + x_2, data = train)
  qda2 <- MASS::qda(y ~ x_1 + x_2, data = train)
```

```
    # iv error
  train_lda2 = c(train_lda2, sum(predict(lda2, train)$class != train$y) / 700)
  test_lda2 = c(test_lda2, sum(predict(lda2, test)$class != test$y) / 300)
  train_qda2 = c(train_qda2, sum(predict(qda2, train)$class != train$y) / 700)
  test_qda2 = c(test_qda2, sum(predict(qda2, test)$class != test$y) / 300)
}
```

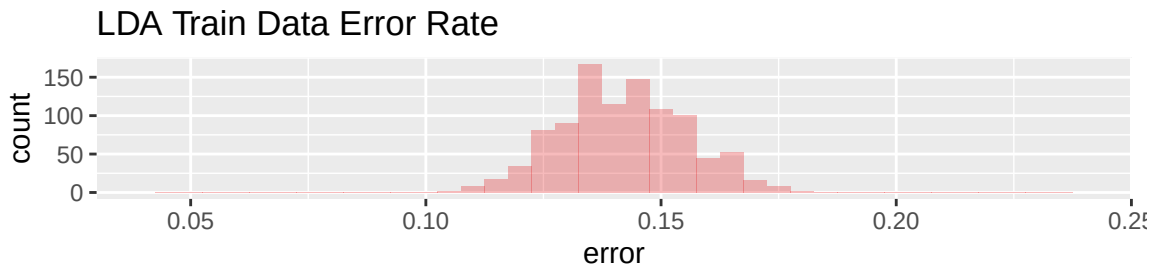
**b. Summarize the results**

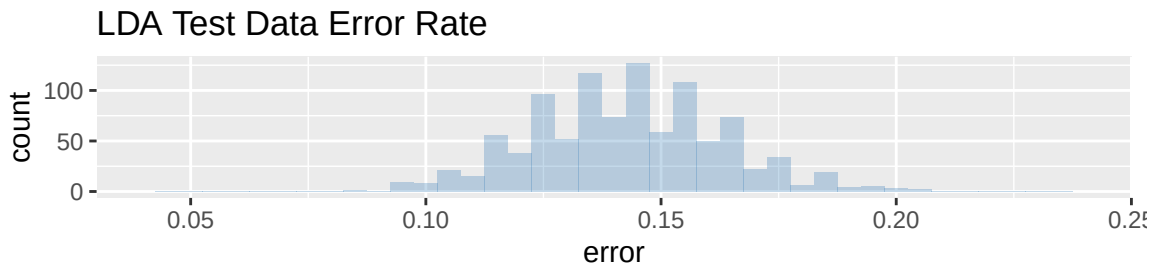
```
df2_result <- data.frame(train_lda2, test_lda2, train_qda2, test_qda2)
df2_result <- df2_result %>%
  gather(dset, error)
df2_result %>%
  group_by(dset) %>%
  summarise(mean = mean(error), sd = sd(error))
```

| dset | mean | sd |
|---|---|---|
| test_lda2 | 0.1427233 | 0.0199135 |
| test_qda2 | 0.1067600 | 0.0171807 |
| train_lda2 | 0.1420243 | 0.0133372 |
| train_qda2 | 0.1051986 | 0.0110349 |

```
lda_p1 <- df2_result %>%
  ggplot(aes(x=error)) +
  scale_colour_manual(name = 'classes', values = twoClassColor) +
  geom_histogram(data=subset(df2_result, dset == 'train_lda2'),
                 fill = brewer.pal(3,'Set1')[1],
                 alpha = 0.3, binwidth = 0.005) +
  scale_x_continuous(limits = c(0.04,0.24)) +
  ggtitle("LDA Train Data Error Rate")

lda_p2 <- df2_result %>%
  ggplot(aes(x=error)) +
  scale_colour_manual(name = 'classes', values = twoClassColor) +
  geom_histogram(data=subset(df2_result, dset == 'test_lda2'),
                 fill = brewer.pal(3,'Set1')[2],
                 alpha = 0.3, binwidth = 0.005) +
  scale_x_continuous(limits = c(0.04,0.24)) +
  ggtitle("LDA Test Data Error Rate")

grid.arrange(lda_p1, lda_p2, nrow = 2)
```

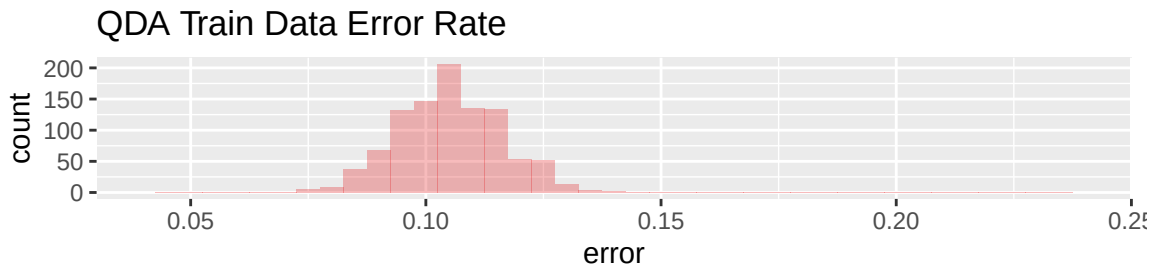
```
## Warning: Removed 2 rows containing missing values (geom_bar).

## Warning: Removed 2 rows containing missing values (geom_bar).
```

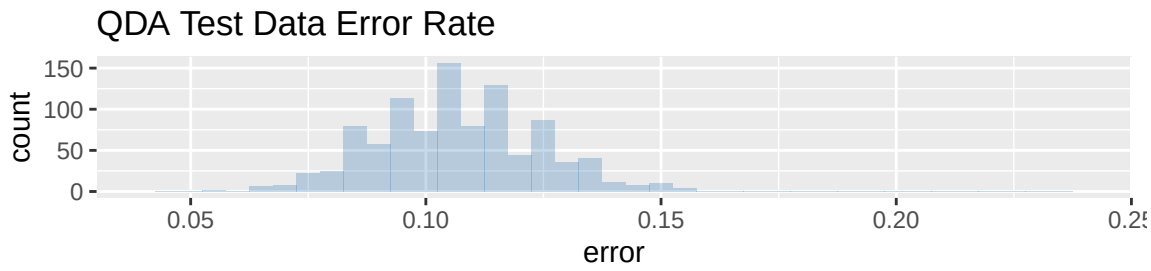



```
qda_p1 <- df2_result %>%
  ggplot(aes(x=error)) +
  scale_colour_manual(name = 'classes', values = twoClassColor) +
  geom_histogram(data=subset(df2_result, dset == 'train_qda2'),
                 fill = brewer.pal(3,'Set1')[1],
                 alpha = 0.3, binwidth = 0.005) +
  scale_x_continuous(limits = c(0.04,0.24)) +
  ggtitle("QDA Train Data Error Rate")

qda_p2 <- df2_result %>%
  ggplot(aes(x=error)) +
  scale_colour_manual(name = 'classes', values = twoClassColor) +
  geom_histogram(data=subset(df2_result, dset == 'test_qda2'),
                 fill = brewer.pal(3,'Set1')[2],
                 alpha = 0.3, binwidth = 0.005) +
  scale_x_continuous(limits = c(0.04,0.24)) +
  ggtitle("QDA Test Data Error Rate")

grid.arrange(qda_p1, qda_p2, nrow = 2)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).

## Warning: Removed 2 rows containing missing values (geom_bar).
```



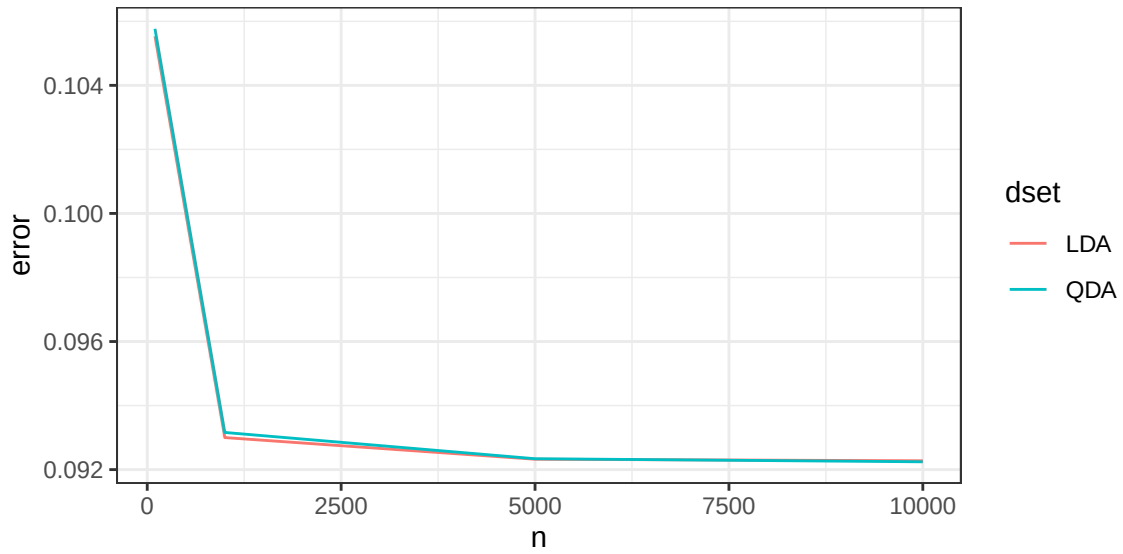

**4. Increasing the sample size**

```
train_lda4_mean = c()
test_lda4_mean = c()
train_qda4_mean = c()
test_qda4_mean = c()
for (n in c(100, 1000, 5000, 10000)) {
  train_lda4 = c()
  test_lda4 = c()
  train_qda4 = c()
  test_qda4 = c()
  for (i in 1:1000) {
    # i data generation
    x_1 = runif(n, -1, 1)
    x_2 = runif(n, -1, 1)
    ep = rnorm(n, 0, 0.25)
    y = case_when(x_1 + x_2 + ep >= 0 ~ TRUE,
                  x_1 + x_2 + ep < 0 ~ FALSE)
    df <- data.frame(x_1, x_2, y)

    # ii split
    split <- initial_split(df, prop = .7)
    train <- training(split)
    test <- testing(split)

    # iii training
    lda4 <- MASS::lda(y ~ x_1 + x_2, data = train)
    qda4 <- MASS::qda(y ~ x_1 + x_2, data = train)

    # iv error
    train_lda4 = c(train_lda4, sum(predict(lda4, train)$class != train$y)
                   / (0.7*n))
    test_lda4 = c(test_lda4, sum(predict(lda4, test)$class != test$y)
                  / (0.3*n))
```

```
    train_qda4 = c(train_qda4, sum(predict(qda4, train)$class != train$y)
                   / (0.7*n))
    test_qda4 = c(test_qda4, sum(predict(qda4, test)$class != test$y)
                  / (0.3*n))
  }
  train_lda4_mean = c(train_lda4_mean, mean(train_lda4))
  test_lda4_mean = c(test_lda4_mean, mean(test_lda4))
  train_qda4_mean = c(train_qda4_mean, mean(train_qda4))
  test_qda4_mean = c(test_qda4_mean, mean(test_qda4))
}
```

**b. Summarize the results**

```
df4 <- data.frame(n=c(100, 1000, 5000, 10000),
                  LDA=test_lda4_mean, QDA=test_qda4_mean)
df4 <- df4 %>%
  gather(dset, error, -n)
df4 %>%
  ggplot(aes(x=n, y=error, color=dset)) +
  geom_line() +
  theme_bw()
```




## Modeling voter turnout

### 5. Building classifiers

**a. Split data**

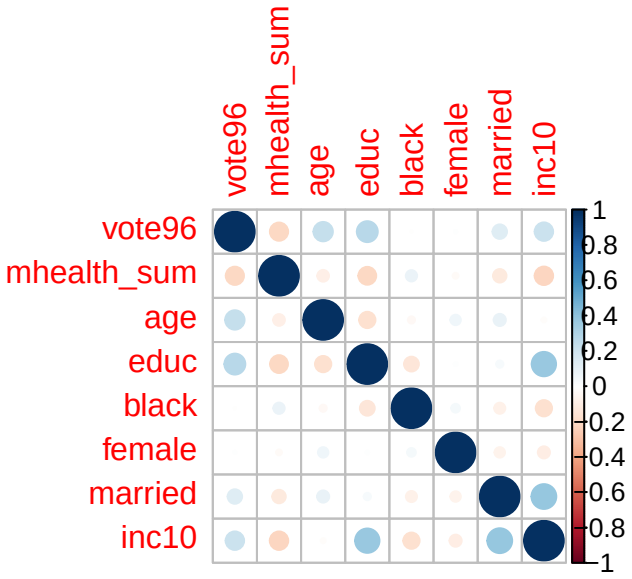
Since there are NAs in the data, I first omit those observations that contains NAs, then split them into training set and test set.

```
df5 <- read.csv('mental_health.csv')
# Drop NAs
df5 <- df5 %>%
  drop_na()
split <- initial_split(df5, prop = .7)
```

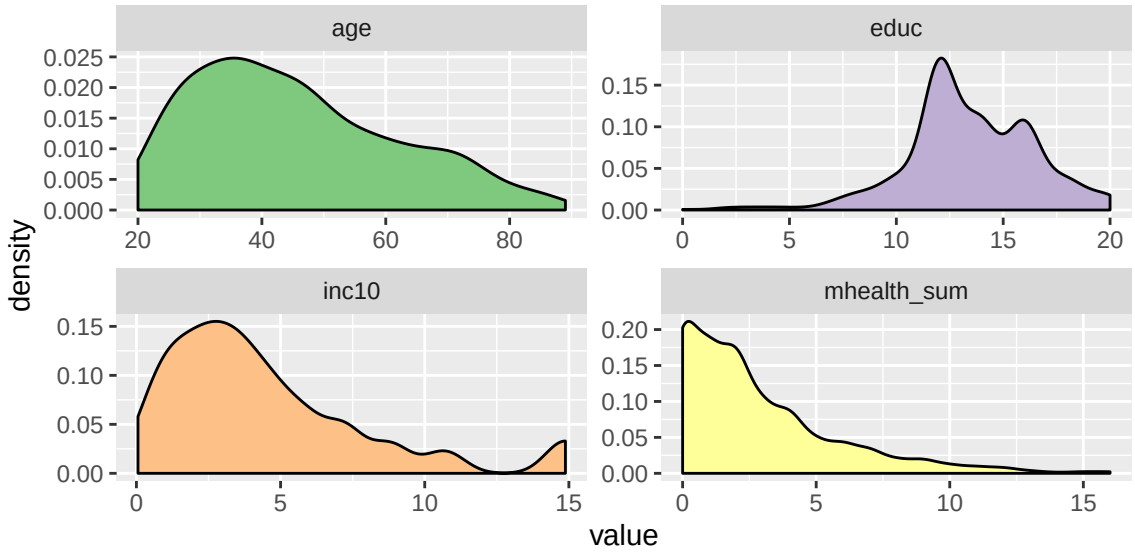
```
train <- training(split)
test <- testing(split)
```

From the Lab notebook, below is some simple EDA.

```
# checking correlation across several features
train %>%
  select_if(is.numeric) %>%
  cor() %>%
  corrplot::corrplot()
```




```
# checking normality assumption for continuous features
train %>%
  dplyr::select(age, educ, inc10, mhealth_sum) %>%
  gather(metric, value) %>%
  ggplot(aes(value, fill = metric)) +
  geom_density(show.legend = FALSE) +
  scale_fill_brewer(type = "qual") +
  facet_wrap(~ metric, scales = "free")
```




**b. Estimation**

For the following training and predicting, I use caret package.
The arguments of the function can be found in https://topepo.github.io/caret/available-models.html

```
# create response and feature data
features <- setdiff(names(train), "vote96")
x <- train[, features]
y <- as.factor(train$vote96)

# Logistic regression
m5.lr <- train(x = x, y = y, method = "glm")

# LDA
m5.lda <- train(x = x, y = y, method = "lda")

# QDA
m5.qda <- train(x = x, y = y, method = "qda")

# Naive Bayes
m5.nb <- train(x = x, y = y, method = "nb")

# K-NN with Euclidean distance metrics
m5.knn <- train(x = x, y = y, method = "knn",
                tuneGrid = expand.grid(k = 1:10))
```

**c. Performance metrics**

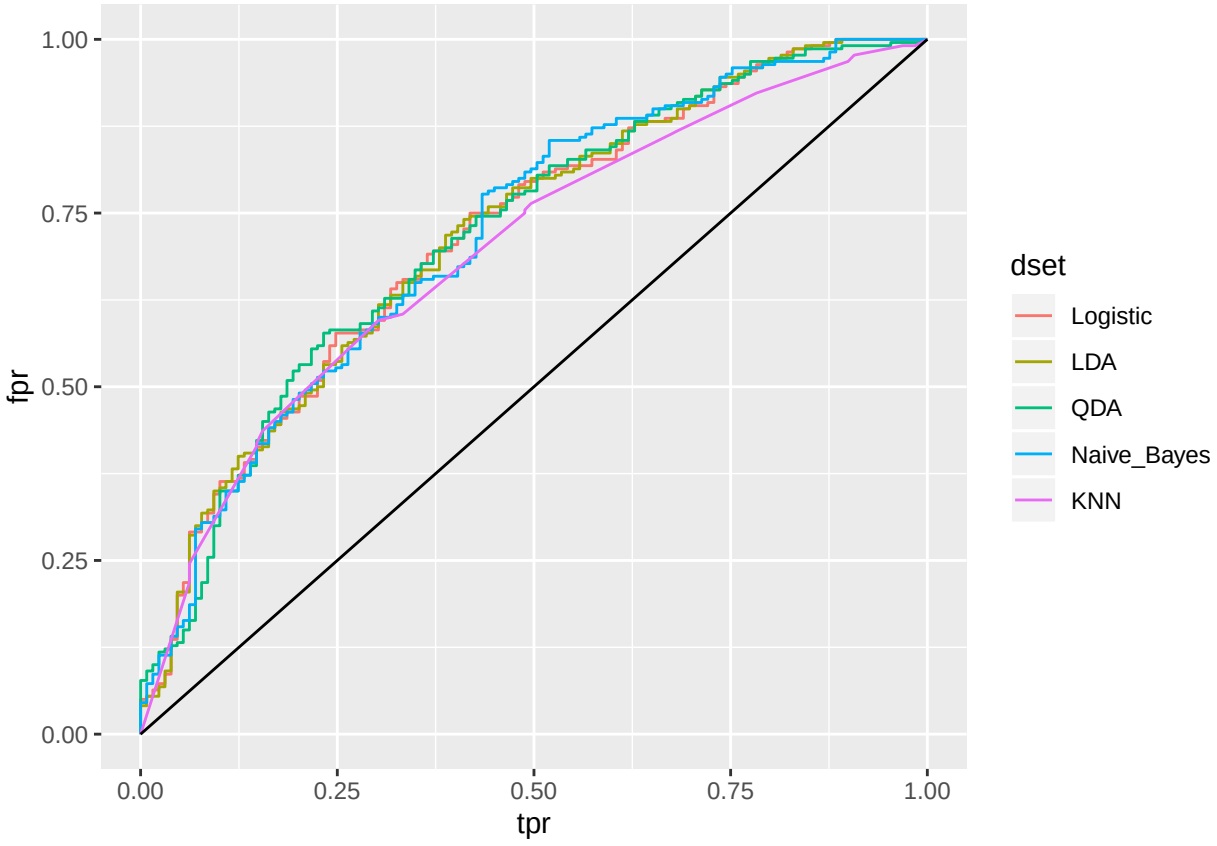
```
# Logistic regression
pred.c <- predict(m5.lr, newdata = test)
pred.p <- predict(m5.lr, newdata = test, type = 'prob')
pred.roc <- prediction(pred.p['1'], as.factor(test$vote96))
auc.tmp <- performance(pred.roc, 'auc')
perf <- performance(pred.roc, 'tpr', 'fpr')
roc.lr <- data.frame(x = slot(perf, 'x.values'),
                     y = slot(perf, 'y.values'),
```

```
                      dset = 'Logistic')
colnames(roc.lr) <- c('tpr', 'fpr', 'dset')

lr.perf <- data.frame(Logistic = c(
  confusionMatrix(pred.c, as.factor(test$vote96))[3]$overall['Accuracy'],
  confusionMatrix(pred.c, as.factor(test$vote96))[4]$byClass,
  AUC = as.numeric(auc.tmp@y.values)))

# LDA
pred.c <- predict(m5.lda, newdata = test)
pred.p <- predict(m5.lda, newdata = test, type = 'prob')
pred.roc <- prediction(pred.p['1'], as.factor(test$vote96))
auc.tmp <- performance(pred.roc, 'auc')
perf <- performance(pred.roc, 'tpr', 'fpr')
roc.lda <- data.frame(x = slot(perf, 'x.values'),
                      y = slot(perf, 'y.values'),
                      dset = 'LDA')
colnames(roc.lda) <- c('tpr', 'fpr', 'dset')

lda.perf <- data.frame(LDA = c(
  confusionMatrix(pred.c, as.factor(test$vote96))[3]$overall['Accuracy'],
  confusionMatrix(pred.c, as.factor(test$vote96))[4]$byClass,
  AUC = as.numeric(auc.tmp@y.values)))

# QDA
pred.c <- predict(m5.qda, newdata = test)
pred.p <- predict(m5.qda, newdata = test, type = 'prob')
pred.roc <- prediction(pred.p['1'], as.factor(test$vote96))
auc.tmp <- performance(pred.roc, 'auc')
perf <- performance(pred.roc, 'tpr', 'fpr')
roc.qda <- data.frame(x = slot(perf, 'x.values'),
                      y = slot(perf, 'y.values'),
                      dset = 'QDA')
colnames(roc.qda) <- c('tpr', 'fpr', 'dset')

qda.perf <- data.frame(QDA = c(
  confusionMatrix(pred.c, as.factor(test$vote96))[3]$overall['Accuracy'],
  confusionMatrix(pred.c, as.factor(test$vote96))[4]$byClass,
  AUC = as.numeric(auc.tmp@y.values)))

# Naive Bayes
pred.c <- predict(m5.nb, newdata = test)
pred.p <- predict(m5.nb, newdata = test, type = 'prob')
pred.roc <- prediction(pred.p['1'], as.factor(test$vote96))
auc.tmp <- performance(pred.roc, 'auc')
perf <- performance(pred.roc, 'tpr', 'fpr')
roc.nb <- data.frame(x = slot(perf, 'x.values'),
                     y = slot(perf, 'y.values'),
                     dset = 'Naive_Bayes')
colnames(roc.nb) <- c('tpr', 'fpr', 'dset')

nb.perf <- data.frame(Naive_Bayes = c(
  confusionMatrix(pred.c, as.factor(test$vote96))[3]$overall['Accuracy'],
```

```
  confusionMatrix(pred.c, as.factor(test$vote96))[4]$byClass,
  AUC = as.numeric(auc.tmp@y.values)))

# K-NN with Euclidean distance metrics
pred.c <- predict(m5.knn, newdata = test)
pred.p <- predict(m5.knn, newdata = test, type = 'prob')
pred.roc <- prediction(pred.p['1'], as.factor(test$vote96))
auc.tmp <- performance(pred.roc, 'auc')
perf <- performance(pred.roc, 'tpr', 'fpr')
roc.knn <- data.frame(x = slot(perf, 'x.values'),
                      y = slot(perf, 'y.values'),
                      dset = 'KNN')
colnames(roc.knn) <- c('tpr', 'fpr', 'dset')

knn.perf <- data.frame(KNN = c(
  confusionMatrix(pred.c, as.factor(test$vote96))[3]$overall['Accuracy'],
  confusionMatrix(pred.c, as.factor(test$vote96))[4]$byClass,
  AUC = as.numeric(auc.tmp@y.values)))

d <- data.frame(x= c(0, 1), y= c(0, 1))

cbind(lr.perf, lda.perf, qda.perf, nb.perf, knn.perf)
```

| | Logistic | LDA | QDA | Naive_Bayes | KNN |
|---|---|---|---|---|---|
| Accuracy | 0.6762178 | 0.6790831 | 0.6848138 | 0.6905444 | 0.6676218 |
| Sensitivity | 0.3178295 | 0.3255814 | 0.4418605 | 0.3333333 | 0.3255814 |
| Specificity | 0.8863636 | 0.8863636 | 0.8272727 | 0.9000000 | 0.8681818 |
| Pos Pred Value | 0.6212121 | 0.6268657 | 0.6000000 | 0.6615385 | 0.5915493 |
| Neg Pred Value | 0.6890459 | 0.6914894 | 0.7165354 | 0.6971831 | 0.6870504 |
| Precision | 0.6212121 | 0.6268657 | 0.6000000 | 0.6615385 | 0.5915493 |
| Recall | 0.3178295 | 0.3255814 | 0.4418605 | 0.3333333 | 0.3255814 |
| F1 | 0.4205128 | 0.4285714 | 0.5089286 | 0.4432990 | 0.4200000 |
| Prevalence | 0.3696275 | 0.3696275 | 0.3696275 | 0.3696275 | 0.3696275 |
| Detection Rate | 0.1174785 | 0.1203438 | 0.1633238 | 0.1232092 | 0.1203438 |
| Detection Prevalence | 0.1891117 | 0.1919771 | 0.2722063 | 0.1862464 | 0.2034384 |
| Balanced Accuracy | 0.6020965 | 0.6059725 | 0.6345666 | 0.6166667 | 0.5968816 |
| AUC | 0.7171952 | 0.7181818 | 0.7186399 | 0.7171600 | 0.6949789 |

```
rbind(roc.lr, roc.lda, roc.qda, roc.nb, roc.knn) %>%
  ggplot(aes(x=tpr, y=fpr)) +
  geom_line(aes(color=dset)) +
  geom_line(data=d, aes(x=x, y=y))
```




**d. Results**

The results above show that the best method is using QDA since its balanced accuracy is the highest (0.6346), and its AUC is the largest (0.7186). Each model has its strength and weakness