# Linear Discriminant Analysis (LDA)

Tarek El-Hajjaoui

2023-05-27

## Problem 1

### Problem 1: Setting Up Data

Find an example that has three classes and choose two quantitative variables/features to analyze. Do NOT use any data set we have analyzed in class. Randomly select 10 observations for testing and the rest for training.

```
data(Credit, package = "ISLR2")
Credit <- na.omit(Credit)
dim(Credit)
```

```
## [1] 400  11
```

Creating three classes based on income {low, medium, high}

```
# create a new variable "income_category" based on income values
Credit$income_category <- cut(Credit$Income, breaks = c(0, 50000/1000, 100000/1000, Inf),
                              labels = c("low", "middle", "high"),
                              include.lowest = TRUE, right = FALSE)
head(Credit[, c("Age", "Rating", "income_category")], 3)
```

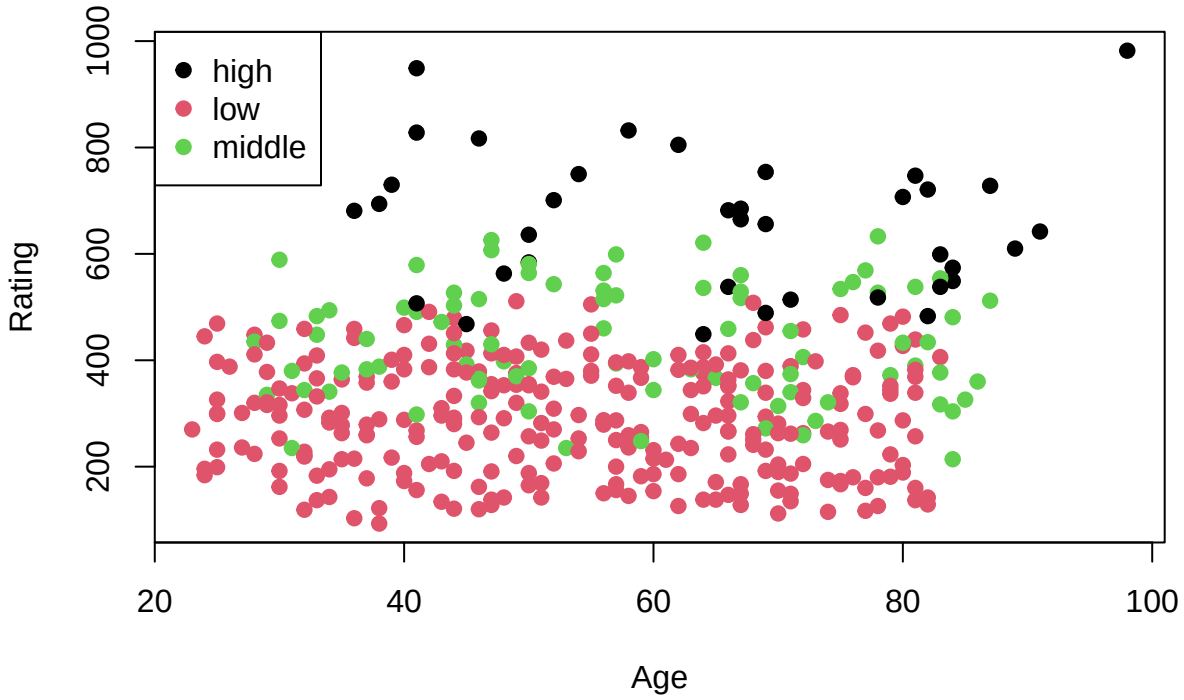
```
##   Age Rating income_category
## 1  34    283             low
## 2  82    483            high
## 3  71    514            high
```

```
cols <- c("Age", "Rating")
income_category <- Credit[, 12]
X <- Credit[, cols]
income_category <- as.factor(as.character(income_category))
head(X, 3)
```

```
##   Age Rating
## 1  34    283
## 2  82    483
## 3  71    514
```

Visualzing Dataset

```
plot(X, col = income_category, pch=19)
legend("topleft", legend = levels(income_category),
       col = 1:3, pch = 19)
```




```
n <- table(Credit$income_category)
N <- sum(n)
cat("There are", N, "total samples.\n")
```

```
## There are 400 total samples.
```

```
cat("The breakdown of income by category is the following:")
```

```
## The breakdown of income by category is the following:
```

```
n
```

```
##
##    low middle   high
##    276     87     37
```

Randomly select 10 observations for testing and the rest for training.

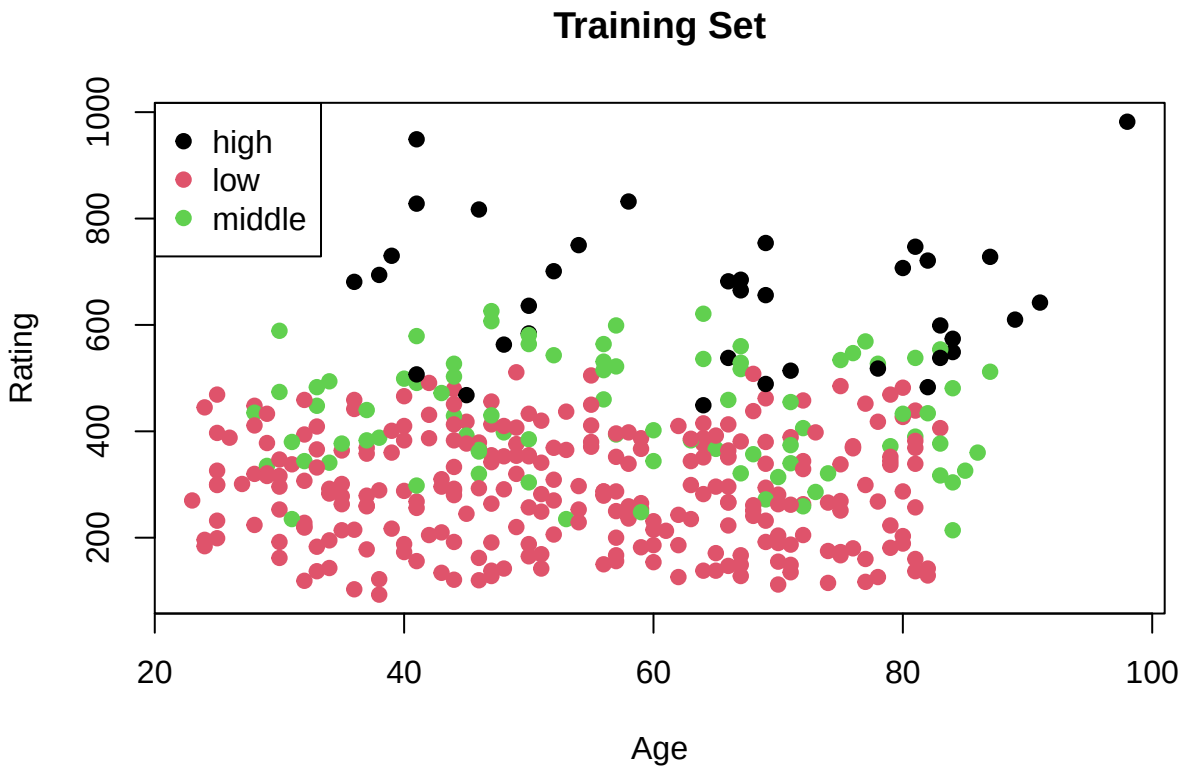
```
set.seed(5)

test.idx <- sample(1:N, 10)

train <- X[-test.idx, ]
income_category.train <- income_category[-test.idx]

income_category.test <- income_category[test.idx]
test <- X[test.idx, ]

cbind(test,income_category.test)
```

```
##     Age Rating income_category.test
## 322  27    236                  low
## 363  62    382                  low
## 185  62    805                 high
## 207  78    180                  low
```

```
## 203  58    145                  low
## 377  46    515               middle
## 297  49    370               middle
## 213  66    323                  low
## 222  78    633               middle
## 71   75    318                  low
```

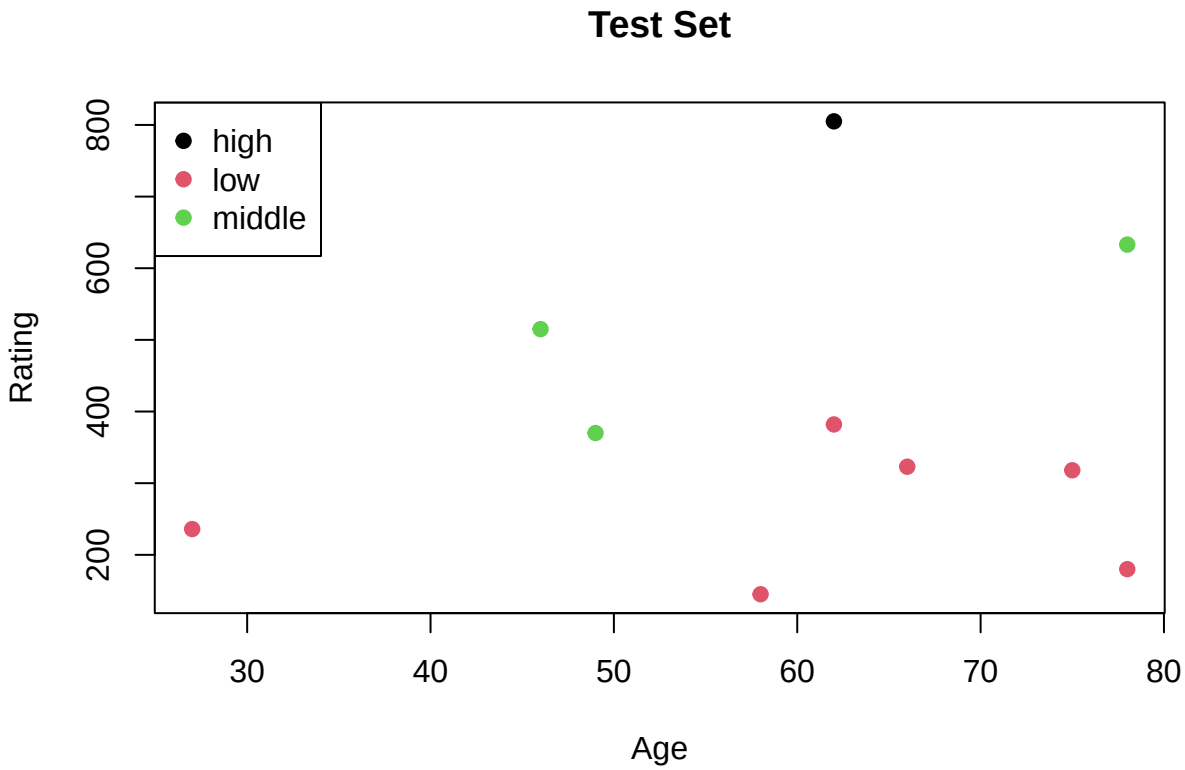
Viewing the observations chosen for the train set.

```
plot(train, col = income_category.train, main = "Training Set", pch=19)
legend("topleft", legend = levels(income_category),
       col = 1:3, pch = 19)
```




Viewing the observations chosen for the test set.

```
plot(test, col = income_category.test, main = "Test Set", pch=19)
legend("topleft", legend = levels(income_category),
       col = 1:3, pch = 19)
```




**Method 1 on next page**

**Problem 1: Method 1**

a. Compute the pooled sample covariance, denoted as $S_{\text{pooled}}$.
b. For each data point in the testing data set, calculate its statistical distance to each of the three groups. Based on this, classify each data point into one of the three groups.

```
# LDA predictor
# minimum Mahalanobis distance approach
predict.type <- function(test, train, income_category.train) {
  n.test <- dim(test)[1]

  n.train <- table(income_category.train)
  N.train <- sum(n.train)

  train1 <- train[income_category.train=="high",]
  train2 <- train[income_category.train=="middle",]
  train3 <- train[income_category.train=="low",]

  n1 <- length(train1)
  n2 <- length(train2)
  n3 <- length(train3)

  means <- cbind(colMeans(train1), colMeans(train2),
                 colMeans(train3))

  colnames(means) <- levels(income_category)

  S1 <- cov(train1)
  S2 <- cov(train2)
  S3 <- cov(train3)

  # Compute the pooled sample covariance, denoted as S.pooled
  S.pooled <- ((n1-1) * S1 + (n2-1) * S2 +
                 (n3-1) * S3) / (N.train-3)

  # For each data point in the testing data set,
  # calculate its statistical distance to each of the three groups.
  group <- rep(0, n.test)
  for (i in 1:n.test) {
    X0 <- test[i, ]
    D <- c(mahalanobis(X0, means[, 1], S.pooled),
           mahalanobis(X0, means[, 2], S.pooled),
           mahalanobis(X0, means[, 3], S.pooled))
    # Classify each data point into one of the three groups.
    group[i] <- which.min(D) # allocate X0 to the group with the minimum Mahalanobis distance
  }
  group
}
```

```
# using predict.type function to make predictions
pred <- predict.type(test, train, income_category.train)
# comparing pred and true values
cbind(income_category.test, pred)
```

```
##       income_category.test pred
##  [1,]                    2    3
```

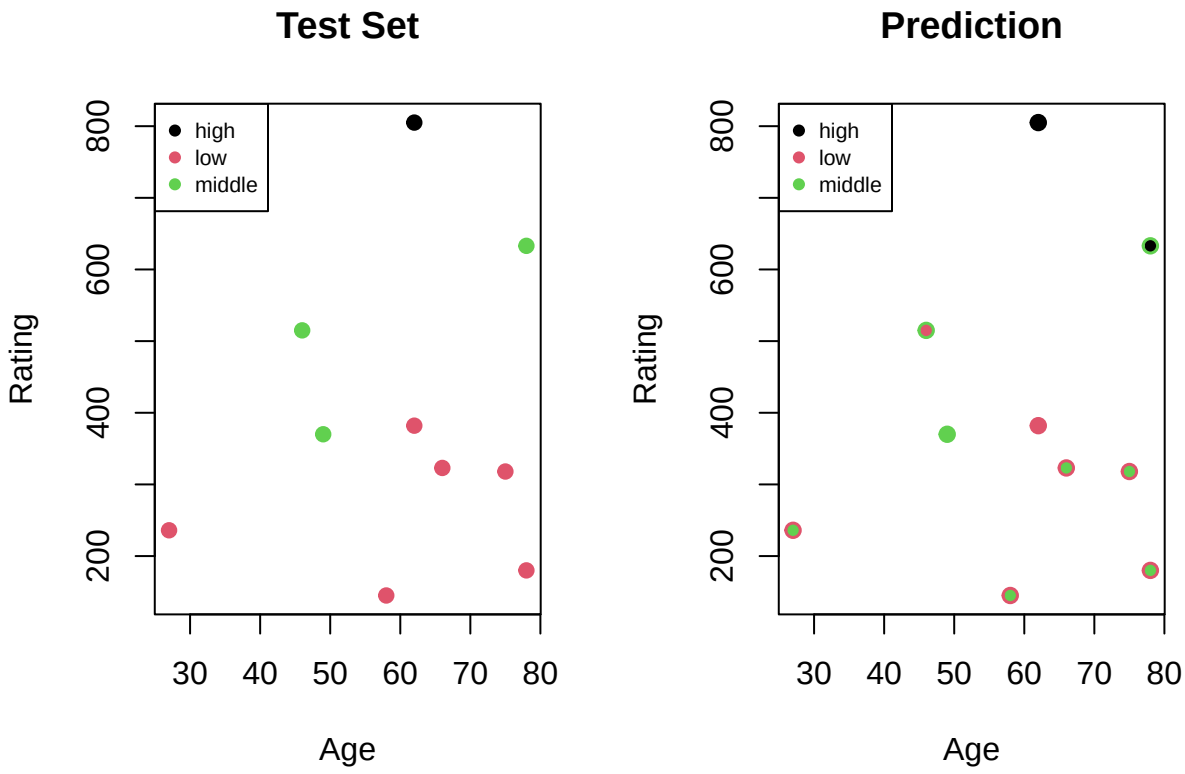
```
##  [2,]                    2    2
##  [3,]                    1    1
##  [4,]                    2    3
##  [5,]                    2    3
##  [6,]                    3    2
##  [7,]                    3    3
##  [8,]                    2    3
##  [9,]                    3    1
## [10,]                    2    3
```

```
correct <- income_category.test==levels(income_category)[pred]
method_1_acc <- sum(correct)/10
cat("Method 1 has a test accuracy of", method_1_acc*100, "%.\n")
```

```
## Method 1 has a test accuracy of 30 %.
```

c. Visualize the classification results in the two-dimensional space of the two chosen features, with the x-axis representing the first feature and the y-axis representing the second feature.

```
par(mfrow = c(1, 2))
plot(test, col = income_category.test, main = "Test Set", pch=19)
legend("topleft", legend = levels(income_category),
       col = 1:3, pch = 19, cex = 0.7)
plot(test, col = income_category.test, main = "Prediction", pch=21, bg=pred, lwd=1.5)
legend("topleft", legend = levels(income_category),
       col = 1:3, pch = 19, cex = 0.7)
```




**Method 2 on next page**

**Problem 1: Method 2**

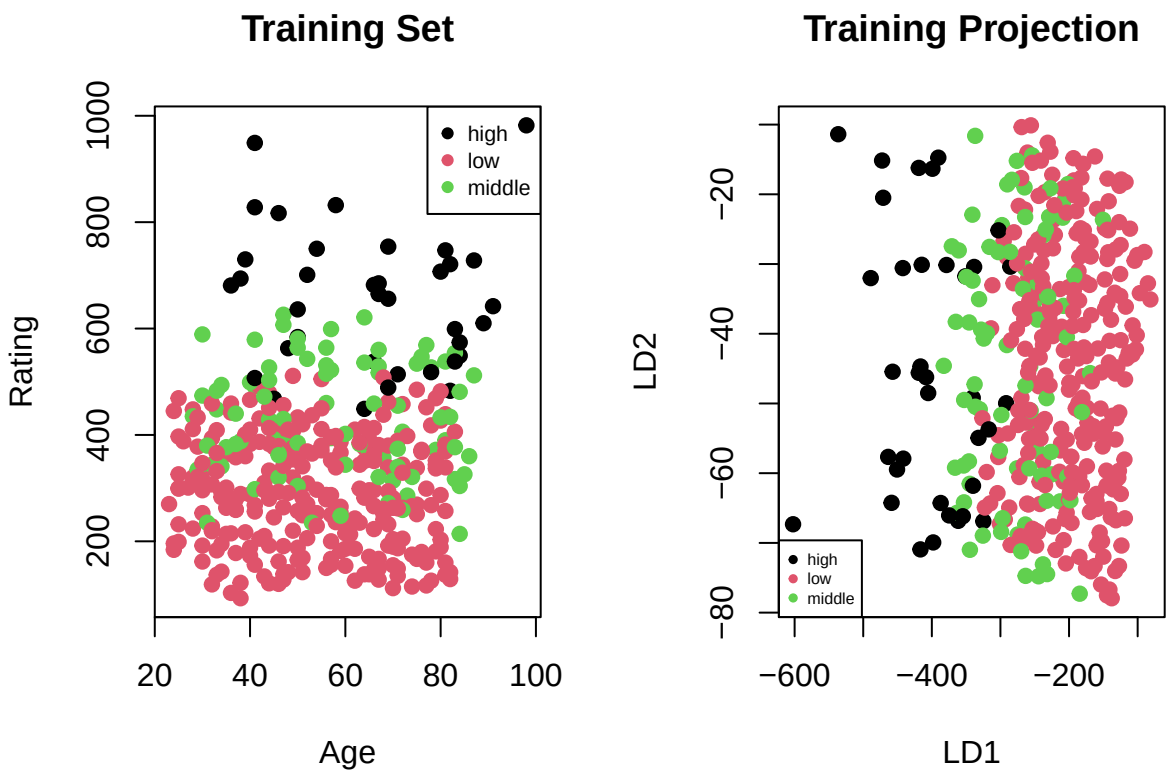
d. Compute the between-group covariance matrix, denoted as $\mathbf{B}$, and the within-group covariance matrix, denoted as $\mathbf{W}$.

e. Find the two linear discriminants by using the eigenvectors of $\mathbf{W}^{-1}\mathbf{B}$. Visualize the directions of the two linear discriminants.

```
n.train <- table(income_category.train)
N.train <- sum(n.train)
train1 <- train[income_category.train=="high",]
train2 <- train[income_category.train=="middle",]
train3 <- train[income_category.train=="low",]
# Total Variance
Tot <- (3*N.train-1)*cov(train)
# Compute the the within-group covariance matrix, W.
W <- (n.train[1]-1)*cov(train1) + (n.train[2]-1)*cov(train2) +(n.train[3]-1)*cov(train3)
# Compute the between-group covariance matrix, B.
B <- Tot-W
# Find the two linear discriminants by using the eigenvectors of W^(-1) * B
LDA <- eigen(solve(W)%*%B)$vector
```

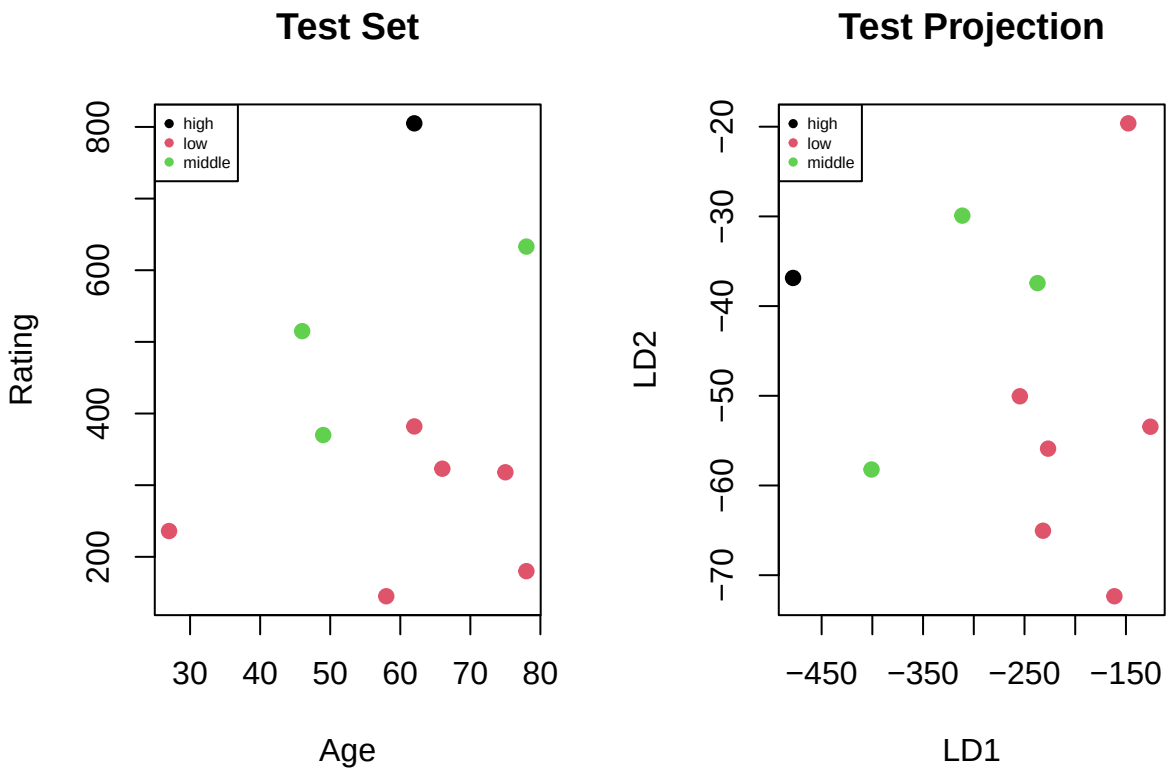
f. Project the training data onto the two linear discriminants and plot the projected data.

```
proj.train <- as.matrix(train)%*%LDA
colnames(proj.train) <- c("LD1", "LD2")

par(mfrow=c(1, 2))
plot(train, col = income_category.train, main = "Training Set", pch=19)
legend("topright", legend = levels(income_category),
       col = 1:3, pch = 19, cex = 0.7)
plot(proj.train, col = income_category.train, pch=19, main = "Training Projection")
legend("bottomleft", legend = levels(income_category),
       col = 1:3, pch = 19, cex = 0.5)
```




g. Transform the testing data to the same two-dimensional space using the linear discriminants. Calculate the distance between each data point and each of the three groups. Classify each data point into one of the three groups based on the minimum distance (Euclidean distance).

```
proj.test <- as.matrix(test)%*%LDA
colnames(proj.test) <- c("LD1", "LD2")

par(mfrow=c(1, 2))
plot(test, col = income_category.test, main = "Test Set", pch=19)
legend("topleft", legend = levels(income_category),
       col = 1:3, pch = 19, cex = 0.5)
plot(proj.test, col = income_category.test, pch=19, main = "Test Projection")
legend("topleft", legend = levels(income_category),
       col = 1:3, pch = 19, cex = 0.5)
```




```
# minimum Euclidean distance approach
proj.train1 <- proj.train[income_category.train=="high",]
proj.train2 <- proj.train[income_category.train=="middle",]
proj.train3 <- proj.train[income_category.train=="low",]

proj.means <- cbind(colMeans(proj.train1), colMeans(proj.train2),
                    colMeans(proj.train3))

colnames(proj.means) <- levels(income_category)

pred2 <- rep(0, 10)

for (i in 1:10){
  X0 <- matrix(rep(proj.test[i, ], 3), 2, 3)
  D <- diag(t(X0-proj.means)%*%(X0-proj.means)) # Euclidean distance
  pred2[i] <- which.min(D)
}
# comparing pred and true values
cbind(income_category.test, pred2)
```

```
##       income_category.test pred2
##  [1,]                    2     3
##  [2,]                    2     2
##  [3,]                    1     1
##  [4,]                    2     3
##  [5,]                    2     3
##  [6,]                    3     2
##  [7,]                    3     3
##  [8,]                    2     3
##  [9,]                    3     1
```

```
## [10,]                    2     3
```

```
correct <- income_category.test==levels(income_category)[pred2]
method_2_acc <- sum(correct)/10
cat("Method 2 has a test accuracy of", method_2_acc*100, "%.\n")
```

```
## Method 2 has a test accuracy of 30 %.
```

### Compare Method 1 and Method 2

h. Do the two methods give you the same predictions?

```
cbind(income_category.test, pred, pred2)
```

```
##       income_category.test pred pred2
##  [1,]                    2    3     3
##  [2,]                    2    2     2
##  [3,]                    1    1     1
##  [4,]                    2    3     3
##  [5,]                    2    3     3
##  [6,]                    3    2     2
##  [7,]                    3    3     3
##  [8,]                    2    3     3
##  [9,]                    3    1     1
## [10,]                    2    3     3
```

- The predictions of both methods are exactly the same as shown above, which is why method 1 has a 30% accuracy and method 2 has a 30% accuracy.

**Problem 2 on next page.**

## Problem 2

Read the following article. Then summarize what you learned from it by writing a paragraph with 100-300 words. Please comment on what extension did the paper make, and why it is useful extension.

"Penalized classification using Fisher's linear discriminant". Link of pdf: https://rss.onlinelibrary.wiley.com/doi/epdf/10.1111/j.1467-9868.2011.00783.x

**Summary**

The article discusses the limitations of linear discriminant analysis (LDA) in the context of high-dimensional data. LDA is a classical method for supervised classification problems where data consists of multiple features measured on observations belonging to different classes. However, when the number of features (p) is much larger than the number of observations (n), LDA becomes inappropriate for two main reasons.

1. In the high-dimensional settings, the standard estimate for the within-class covariance matrix becomes singular, making it impossible to apply the usual discriminant rule.
2. The interpretation of the classification rule obtained from LDA becomes challenging since it involves all p features.

To address these issues, the authors propose a method called penalized LDA, which is evaluated through simulation studies on gene expression datasets.

- The penalized LDA method addresses the issue of singularity by introducing a penalty term in the optimization objective. This penalty term is applied to the discriminant vectors, which are the coefficients that determine the linear combination of features used for classification. Essentially the penalty term acts as a regularizer by shrinking or setting the coefficients to zero of less informative features. As a result, the penalized LDA method reduces the dimensionality of the feature space, thereby avoiding the singularity problem.
- The penalization method not only addresses the singularity issue but also provides a more interpretable classification rule. Reducing the number of features results in a more concise and understandable set of features that contribute to the classification decision, enhancing the interpretability of the results.