STAT-627 Final Project-NBA

Te-Jou(Carol) Hou, Sean Hsu, Chih-Chen Wang (Gilbert), Fang-Yi Wu(Eva)

2024-12-02

```r
library(MASS)
library(caret)
library(dplyr)
library(nnet)
library(class)
library(tree)
library(e1071)
library(tidyverse)
```

I. Importing and Organizing Data

```r
# Read the dataset into NBA_df
path <- file.path(
  "/Users/houderou/Library/Mobile Documents/com~apple~CloudDocs",
  "AMU documents /MS in DS 1/STAT-627-002_Statistical Machine Learning",
  "Final project/nba_salaries_all.csv"
)
NBA_df <- read.csv(path)

# Calculate the quartiles (25th, 50th, 75th percentiles) of the Salary column
Q1 <- quantile(NBA_df$Salary, 0.25)
Q2 <- quantile(NBA_df$Salary, 0.5)
Q3 <- quantile(NBA_df$Salary, 0.75)

# Process the NBA_df dataset
NBA_df |>
  select(Salary, AST,  STL,  BLK , TOV, PF, PTS) |>
  rename(
    Assists = AST,
    Steals = STL,
    Blocks = BLK,
    Turnovers = TOV,
    Personal_Fouls = PF,
    Points = PTS
  ) |>
  mutate(Salary_Group = case_when(
    Salary <= Q1 ~ "Budget Tier",
    Salary > Q1 & Salary <= Q2 ~ "Mid-Tier",
    Salary > Q2 & Salary <= Q3 ~ "Upper Mid-Tier",
```

```r
    Salary > Q3 ~ "Premium Tier")
  ) |>
  mutate(Salary_Group = as.factor(Salary_Group)) |>
  select(-Salary)-> NBA_df
```

II. Model Building and Evaluation

1. Logistic Regression

```r
set.seed(123)

train_index <- createDataPartition(NBA_df$Salary_Group, p = 0.7, list = FALSE)
train_data <- NBA_df[train_index, ]
test_data <- NBA_df[-train_index, ]

train_control <- trainControl(method = "cv", number = 10)

logistic_cv_model <- train(
  Salary_Group ~Assists + Steals + Blocks + Turnovers + Personal_Fouls + Points,
  data = train_data,
  method = "multinom",
  trControl = train_control,
  trace = FALSE )

test_predictions <- predict(logistic_cv_model, newdata = test_data)
conf_matrix <- confusionMatrix(test_predictions, test_data$Salary_Group)
conf_matrix$overall["Accuracy"]
```

```
##  Accuracy
## 0.4130435
```

```r
category_accuracies <- conf_matrix$byClass[, "Balanced Accuracy"]
category_accuracies
```

```
##    Class: Budget Tier       Class: Mid-Tier   Class: Premium Tier
##             0.6463245             0.5132919             0.7514563
## Class: Upper Mid-Tier
##             0.5214932
```

2. LDA and QDA

```r
#LDA
set.seed(123)
train_index <- createDataPartition(NBA_df$Salary_Group, p = 0.7, list = FALSE)
train_2 <- NBA_df[train_index, ]
test_2<- NBA_df[-train_index, ]
```

```r
lda_model <- lda(Salary_Group ~ ., data = train_2)
lda_predictions <- predict(lda_model, test_2)

conf_matrix_lda <- confusionMatrix(lda_predictions$class, test_2$Salary_Group)

overall_accuracy <- conf_matrix_lda$overall["Accuracy"]
overall_accuracy
```

```
##  Accuracy
## 0.4565217
```

```r
balanced_accuracy <- conf_matrix_lda$byClass[, "Balanced Accuracy"]
balanced_accuracy
```

```
##    Class: Budget Tier       Class: Mid-Tier   Class: Premium Tier
##             0.7077670             0.4977376             0.7563107
## Class: Upper Mid-Tier
##             0.5851244
```

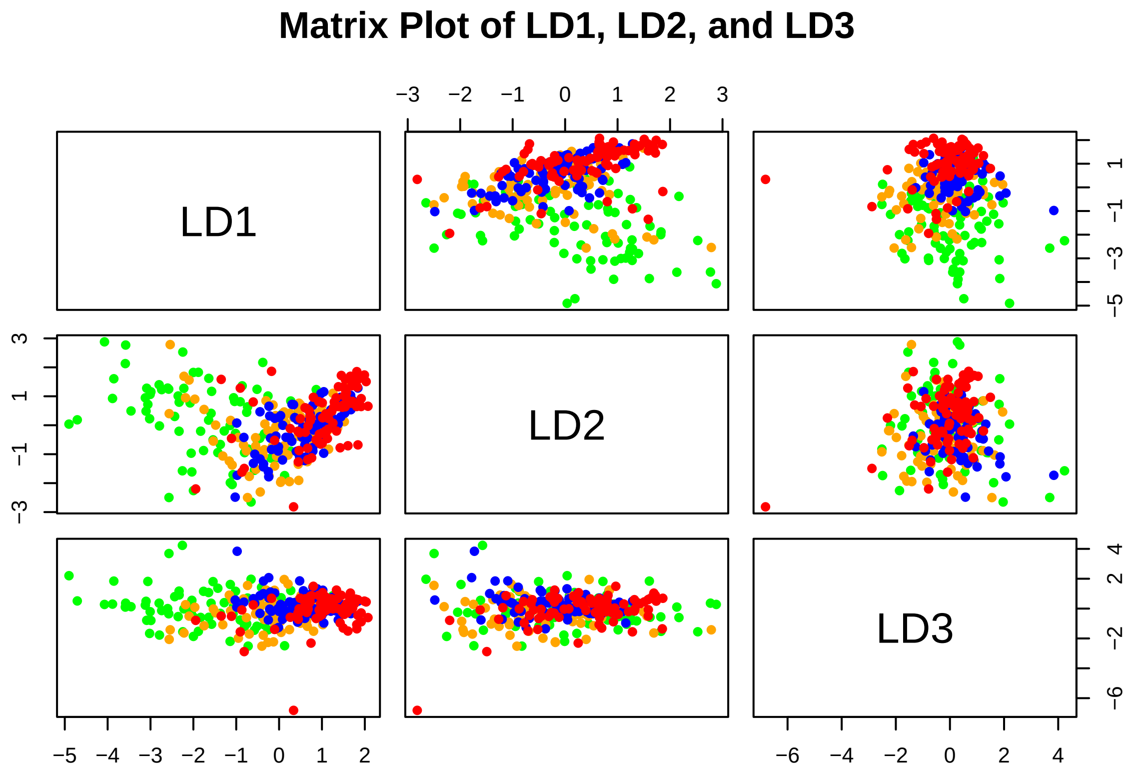
```r
head(lda_predictions$posterior,n=3)
```

```
##    Budget Tier    Mid-Tier Premium Tier Upper Mid-Tier
## 3  0.112581578 0.056934957    0.4861354    0.344348069
## 12 0.000114267 0.001235918    0.9919031    0.006746698
## 13 0.018719907 0.206933175    0.4257466    0.348600329
```

```r
lda_coords <- predict(lda_model, train_2)$x

custom_colors <- c("Budget Tier" = "red",
                   "Mid-Tier" = "blue",
                   "Premium Tier" = "green",
                   "Upper Mid-Tier" = "orange")

pairs(lda_coords,
      col = custom_colors[train_2$Salary_Group],
      pch = 16,
      main = "Matrix Plot of LD1, LD2, and LD3")
```




```r
#QDA
qda_model <- qda(Salary_Group ~ ., data = train_2)
qda_predictions <- predict(qda_model, test_2)

conf_matrix_qda <- confusionMatrix(qda_predictions$class, test_2$Salary_Group)
overall_accuracy <- conf_matrix_qda$overall['Accuracy']
overall_accuracy
```

```
##  Accuracy
## 0.4710145
```

```r
balanced_accuracy <- rowMeans(conf_matrix_qda$byClass[, c('Sensitivity',
                                                          'Specificity')],
                              na.rm = TRUE)
balanced_accuracy
```

```
##    Class: Budget Tier       Class: Mid-Tier   Class: Premium Tier
##             0.6277393             0.6218891             0.7800277
## Class: Upper Mid-Tier
##             0.5596719
```

```r
head(qda_predictions$posterior,n=3)
```

```
##     Budget Tier     Mid-Tier Premium Tier Upper Mid-Tier
```

```
## 3  1.507387e-05 9.384871e-17    0.9931651   0.0068198440
## 12 5.006216e-11 7.756597e-14    0.9995301   0.0004698639
## 13 1.460671e-08 4.723957e-01    0.5177591   0.0098452379
```

Comparison Between LDA and QDA:

```
# CV Accuracy:
lda_cv <- lda(Salary_Group ~ ., data = train_2, CV = TRUE)
lda_cv_accuracy <- mean(lda_cv$class == train_2$Salary_Group)
cat("LDA CV Accuracy:", lda_cv_accuracy, "\n")
```

```
## LDA CV Accuracy: 0.4924012
```

```
qda_cv <- qda(Salary_Group ~ ., data = train_2, CV = TRUE)
qda_cv_accuracy <- mean(qda_cv$class == train_2$Salary_Group)
cat("QDA CV Accuracy:", qda_cv_accuracy, "\n")
```

```
## QDA CV Accuracy: 0.4741641
```

3. KNN

```
#Knn
set.seed(123)

train_index <- createDataPartition(NBA_df$Salary_Group, p = 0.7, list = FALSE)
train_3 <- NBA_df[train_index, ]
test_3 <- NBA_df[-train_index, ]

train_control <- trainControl(method = "cv", number = 10)

knn_cv_model <- train(
  Salary_Group ~Assists + Steals + Blocks + Turnovers + Personal_Fouls + Points,
  data = train_3,
  method = "knn",
  trControl = train_control,
  tuneGrid = data.frame(k = 1:20)
)

optimal_k <- knn_cv_model$bestTune$k
optimal_k
```

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

```
test_predictions <- knn(
  train = train_3[, c("Assists", "Steals", "Blocks",
                      "Turnovers", "Personal_Fouls", "Points")],
  test = test_3[, c("Assists", "Steals", "Blocks",
                    "Turnovers", "Personal_Fouls", "Points")],
  cl = train_3$Salary_Group,
```

```r
  k = optimal_k
)
test_accuracy <- mean(test_predictions == test_3$Salary_Group)

conf_matrix <- confusionMatrix(
  factor(test_predictions, levels = levels(test_3$Salary_Group)),
  test_3$Salary_Group)

class_stats <- conf_matrix$byClass

overall_accuracy <- conf_matrix$overall['Accuracy']
overall_accuracy
```

```
##  Accuracy
## 0.4565217
```

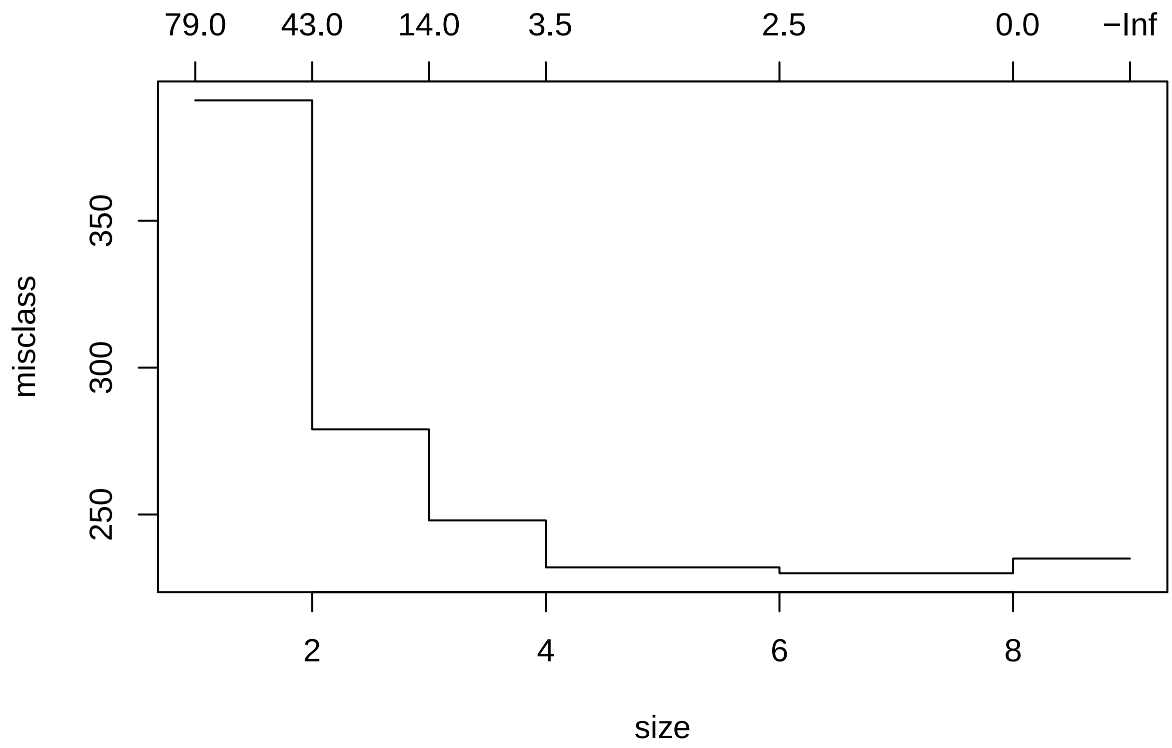
```r
balanced_accuracy <- rowMeans(class_stats[, c('Sensitivity', 'Specificity')],
                              na.rm = TRUE)
balanced_accuracy
```

```
##    Class: Budget Tier       Class: Mid-Tier   Class: Premium Tier
##             0.6552011             0.5608032             0.7414702
## Class: Upper Mid-Tier
##             0.5899321
```

4. Decision Trees

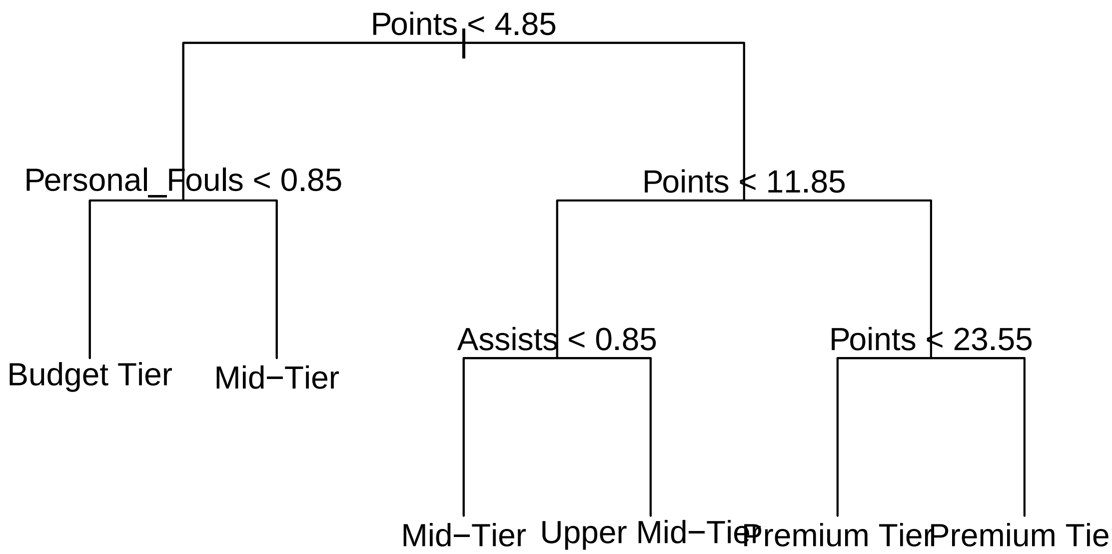
```r
# Build a decision tree model
set.seed(123)
TREE = tree(Salary_Group ~ ., data = NBA_df)

# Apply cross-validation techniques to find the optimal tuning parameters
set.seed(123)
CV.TREE = cv.tree(TREE, FUN = prune.misclass)
#CV.TREE
plot(CV.TREE)
```




```r
# The best decision tree model
TREE.BEST = prune.tree(TREE, best = 6)

plot(TREE.BEST, type="uniform")
text(TREE.BEST)
```




```r
TREE.BEST
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 467 1295.00 Budget Tier ( 0.252677 0.248394 0.250535 0.248394 )
##    2) Points < 4.85 146  283.00 Budget Tier ( 0.554795 0.328767 0.006849 0.109589 )
##      4) Personal_Fouls < 0.85 57   72.17 Budget Tier ( 0.789474 0.157895 0.000000 0.052632 ) *
##      5) Personal_Fouls > 0.85 89  188.50 Mid-Tier ( 0.404494 0.438202 0.011236 0.146067 ) *
##    3) Points > 4.85 321  840.30 Premium Tier ( 0.115265 0.211838 0.361371 0.311526 )
##      6) Points < 11.85 205  538.10 Upper Mid-Tier ( 0.146341 0.302439 0.170732 0.380488 )
##       12) Assists < 0.85 33   75.21 Mid-Tier ( 0.303030 0.515152 0.090909 0.090909 ) *
##       13) Assists > 0.85 172  438.90 Upper Mid-Tier ( 0.116279 0.261628 0.186047 0.436047 ) *
##      7) Points > 11.85 116  206.20 Premium Tier ( 0.060345 0.051724 0.698276 0.189655 )
##       14) Points < 23.55 89  183.40 Premium Tier ( 0.078652 0.067416 0.606742 0.247191 ) *
##       15) Points > 23.55 27    0.00 Premium Tier ( 0.000000 0.000000 1.000000 0.000000 ) *
```

```r
summary(TREE.BEST)
```

```
##
## Classification tree:
## snip.tree(tree = TREE, nodes = c(5L, 13L))
## Variables actually used in tree construction:
## [1] "Points"         "Personal_Fouls" "Assists"
## Number of terminal nodes:  6
```

```
## Residual mean deviance:  2.078 = 958.2 / 461
## Misclassification error rate: 0.4497 = 210 / 467
```

5. SVM

```r
# Split data into training and testing sets
set.seed(123)
train_indices <- sample(1:nrow(NBA_df), size = 0.7 * nrow(NBA_df))
train_data <- NBA_df[train_indices, ]
test_data <- NBA_df[-train_indices, ]

# Build the SVM model
svm_model <- svm(Salary_Group ~ ., data = train_data, kernel = "radial",
                 cost = 1)

# Model evaluation
predicted <- predict(svm_model, test_data)
conf_matrix <- table(Predicted = predicted, Actual = test_data$Salary_Group)
#print(conf_matrix)

# Calculate accuracy
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
print(paste("Accuracy:", round(accuracy, 2)))
```

```
## [1] "Accuracy: 0.46"
```

```r
# Define the kernels to evaluate
kernels <- c("linear", "polynomial", "radial", "sigmoid")

# Initialize a list to store tuning results for each kernel
results <- list()

# Loop through each kernel and perform tuning
for (kernel_type in kernels) {
  set.seed(123)
  tuned_svm <- tune(
    svm,
    Salary_Group ~ .,
    data = train_data,
    kernel = kernel_type,
    ranges = list(cost = c(0.1, 1, 10, 100), gamma = c(0.01, 0.1, 1))
  )

  # Store the result for the current kernel
  results[[kernel_type]] <- tuned_svm
}

# Extract the best model across all kernels
best_result <- NULL
best_performance <- Inf
```

```r
for (kernel_type in kernels) {
  if (results[[kernel_type]]$best.performance < best_performance) {
    best_performance <- results[[kernel_type]]$best.performance
    best_result <- results[[kernel_type]]
  }
}

# Print the best model and its parameters
print(best_result$best.model)
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = Salary_Group ~ ., data = train_data,
##     ranges = list(cost = c(0.1, 1, 10, 100), gamma = c(0.01, 0.1,
##         1)), kernel = kernel_type)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  299
```

```r
# Summary of the best tuning result
#summary(best_result)
# Calculate best accuracy
best_accuracy <- 1 - best_result$best.performance
print(paste("Best Accuracy:", round(best_accuracy, 4)))
```

```
## [1] "Best Accuracy: 0.5067"
```