stats 101c final project

Sizhuo Tian

2024-12-04

```
library(dplyr)
library(randomForest)
library(readxl)
library(caTools)
library(tidyr)

data <- read_excel("Dataset.xlsx", na = "-")  # Treat '-' as NA
```

Task 1: predict the winning team based on higher average total scores

```
# Step 1: Load and Clean Data
data <- read_excel("Dataset.xlsx", na = "-") %>%
  mutate(
    `Game Date` = as.Date(`Game Date`, format = "%m/%d/%Y"),  # Convert to Date
    PTS = as.numeric(PTS)  # Ensure PTS is numeric
  )

# Step 2: Calculate Team Average Scores
average_scores <- data %>%
  group_by(Team) %>%
  summarize(Avg_PTS = mean(PTS, na.rm = TRUE))

# Step 3: Add Opponent Information and Score Difference
data <- data %>%
  left_join(average_scores, by = "Team") %>%
  mutate(
    Opponent = ifelse(grepl("@", `Match Up`),
                      sub(".* @ ", "", `Match Up`),
                      sub(".* vs\\. ?", "", `Match Up`)),
    Opponent_Avg_PTS = average_scores$Avg_PTS[match(Opponent, average_scores$Team)],
    Avg_Score_Diff = Avg_PTS - Opponent_Avg_PTS  # Calculate score difference
  )

# Step 4: Train-Test Split
cutoff_date <- as.Date("2024-02-27")
train_data <- data %>% filter(`Game Date` < cutoff_date)
test_data <- data %>% filter(`Game Date` >= cutoff_date)

# Step 5: Train Random Forest Model
rf_model <- randomForest(
```

```
  factor(`W/L`) ~ Avg_Score_Diff,  # Predict Win/Loss using score difference
  data = train_data,
  ntree = 500,
  importance = TRUE
)

# Step 6: Evaluate the Model
predictions <- predict(rf_model, test_data)
confusion_matrix <- table(Predicted = predictions, Actual = test_data$`W/L`)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

# Print Results
print(confusion_matrix)
```

```
##           Actual
## Predicted   L   W
##         L 234 179
##         W 136 191
```

```
print(paste("Accuracy: ", round(accuracy * 100, 2), "%"))
```

```
## [1] "Accuracy:  57.43 %"
```

Task 2: predict the winning team based on "giving more weight to data closer to the game date"

```
data <- read_excel("Dataset.xlsx", na = "-") %>%
  mutate(`Game Date` = as.Date(`Game Date`, format = "%m/%d/%Y"))

team_avg_scores <- data %>%
  group_by(Team) %>%
  summarize(Team_Avg_PTS = mean(PTS, na.rm = TRUE))


calculate_weighted_average <- function(team, target_date, data, team_avg_scores) {
  # Filter past games for the team
  past_games <- data %>%
    filter(Team == team & `Game Date` < target_date)

  # If there are no past games, return the team's average score as the default
  if (nrow(past_games) == 0) {
    default_score <- team_avg_scores$Team_Avg_PTS[team_avg_scores$Team == team]
    return(default_score)
  }

  # Calculate weights: 1 / (days difference + 1)
  past_games <- past_games %>%
    mutate(weight = 1 / (as.numeric(target_date - `Game Date`) + 1))

  # Compute the weighted average score
```

```
  weighted_avg_pts <- sum(past_games$PTS * past_games$weight, na.rm = TRUE) /
                      sum(past_games$weight, na.rm = TRUE)

  weighted_avg_pts
}


data <- data %>%
  rowwise() %>%
  mutate(Weighted_Avg_PTS = calculate_weighted_average(Team, `Game Date`, data, team_avg_scores)) %>%
  ungroup()


cutoff_date <- as.Date("2024-02-27")
train_data <- data %>% filter(`Game Date` < cutoff_date)
test_data <- data %>% filter(`Game Date` >= cutoff_date)

# Convert W/L to a factor
train_data <- train_data %>%
  mutate(`W/L` = as.factor(`W/L`))

# Train the model
rf_model <- randomForest(
  factor(`W/L`) ~ Weighted_Avg_PTS,  # Use weighted average as the predictor
  data = train_data,
  ntree = 500,
  importance = TRUE
)

# Predict outcomes on the test set
predictions <- predict(rf_model, test_data)

# Generate a confusion matrix
confusion_matrix <- table(Predicted = predictions, Actual = test_data$`W/L`)

# Calculate accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

# Print results
print(confusion_matrix)
```

```
##          Actual
## Predicted   L   W
##         L 215 194
##         W 155 176
```

```
print(paste("Accuracy: ", round(accuracy * 100, 2), "%"))
```

```
## [1] "Accuracy:  52.84 %"
```