```r
# ================================
# COM692 Coursework 2: Football Data Analysis
# Author: Rahmatul Ektidar Aziz
# ================================

# ======================
# 0. Load Required Libraries
# ======================
library(ggplot2)
library(dplyr)
library(corrplot)
library(pROC)
library(glmnet)

# ======================
# 1. Load and Preprocess Data
# ======================

# Load datasets
team_stats <- read.csv("team_stats.csv")
standings <- read.csv("standings.csv")
match_summary <- read.csv("match_summary.csv")

# Remove duplicates
match_summary <- match_summary[!duplicated(match_summary), ]

# Format dates
match_summary$date <- as.POSIXct(match_summary$date)

# Remove missing values
match_summary <- na.omit(match_summary)

# Merge stats
merged_stats <- merge(team_stats, standings, by = "team") %>% distinct()
merged_stats$goal_difference <- merged_stats$goals.x - merged_stats$conceded

# ======================
# 2. Exploratory Analysis
# ======================

# Goals vs Points
ggplot(merged_stats, aes(x = goals.x, y = points)) +
  geom_point(color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Goals vs. Points", x = "Goals Scored", y = "League Points")

# Possession vs Rank
ggplot(merged_stats, aes(x = possession, y = rank)) +
  geom_point(color = "darkgreen") +
  geom_smooth(method = "lm", se = FALSE, color = "orange") +
  scale_y_reverse() +
  labs(title = "Possession vs. Rank", x = "Possession (%)", y = "League Rank")

# Expected Goals vs Rank
ggplot(merged_stats, aes(x = expected_goals, y = rank)) +
  geom_point(color = "purple") +
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  scale_y_reverse() +
  labs(title = "Expected Goals vs. Rank", x = "Expected Goals (xG)", y = "League Rank")

# Correlation matrix
numeric_data <- merged_stats[sapply(merged_stats, is.numeric)]
cor_matrix <- cor(numeric_data, use = "complete.obs")
corrplot(cor_matrix, method = "color", type = "upper", tl.col = "black")
```

```
# ============================
# 3. Match Outcome Preparation
# ============================

# Define outcomes
match_summary$outcome <- ifelse(match_summary$home_goals > match_summary$away_goals, "HomeWin",
                                ifelse(match_summary$home_goals < match_summary$away_goals,
"AwayWin", "Draw"))
match_summary$binary_outcome <- ifelse(match_summary$outcome == "HomeWin", 1, 0)

# Train/test split
set.seed(123)
sample_index <- sample(1:nrow(match_summary), 0.8 * nrow(match_summary))
train_data <- match_summary[sample_index, ]
test_data <- match_summary[-sample_index, ]

# ================================
# 4. Regularized Logistic Modeling
# ================================

# Prepare input for glmnet
X_train <- as.matrix(train_data[, c("home_goals", "away_goals", "attendance")])
y_train <- as.numeric(train_data$binary_outcome)

# Train Lasso model with cross-validation
cv_lasso <- cv.glmnet(X_train, y_train, family = "binomial", alpha = 1, nfolds = 5)
plot(cv_lasso)
best_lambda <- cv_lasso$lambda.min
print(paste("Best Lambda:", best_lambda))  # Example output: "Best Lambda: 9.588e-05"

# Predict on test set
X_test <- as.matrix(test_data[, c("home_goals", "away_goals", "attendance")])
y_test <- as.numeric(test_data$binary_outcome)

# Predict probabilities and classes
predicted_prob <- predict(cv_lasso, newx = X_test, s = best_lambda, type = "response")
predicted_class <- ifelse(predicted_prob > 0.5, 1, 0)

# Accuracy
accuracy <- mean(predicted_class == y_test)
print(paste("Lasso Model Accuracy:", round(accuracy * 100, 2), "%"))  # Example: "Lasso Model
Accuracy: 100%"

# ==================
# 5. Model Evaluation
# ==================

# ROC Curve
roc_obj <- roc(y_test, as.numeric(predicted_prob))
plot(roc_obj, col = "blue", main = "ROC Curve - Lasso Logistic Model")
auc(roc_obj)  # Example output: Area under the curve: 1
```