VIRGINIA COMMONWEALTH UNIVERSITY

Statistical Analysis and Modeling (SCMA 632)

A1b: Analysing IPL data using PYTHON and R

by

IDAMAKANTI SREENIDHI

V01107252

Date of Submission: 18-06-2024

CONTENTS

Sl. No.	Title	Page No.
1.	Introduction	3-4
2.	Analysis, Result, and Conclusion Using R	4-8
3.	Analysis, Result, and Conclusion Using Python	8-11

Analysing IPL data using ‘R’ and ‘python’

Introduction:

The Indian Premier League (IPL) stands as one of the most exciting and popular T20 cricket tournaments globally, showcasing top cricketing talents from around the world. In this assignment, we delve into analyzing IPL data spanning multiple seasons. The objectives encompass extracting, organizing, and analyzing player performance data to derive insights into batting and bowling performances, as well as exploring the relationship between player performance and their salaries.

Assignment Overview:

1. Data Extraction and Organization:
  - Utilize R or Python to extract and load IPL data files.
  - Organize the data round-wise, focusing on key metrics such as runs scored, wickets taken, and other relevant statistics per player per match.

2. Top Performers Analysis:
  - Identify the top three run-getters and top three wicket-takers for each IPL season considered.
  - Analyze the distribution of runs scored and wickets taken by these top players using appropriate statistical methods, aiming to fit the most suitable distribution.

3. Statistical Modeling:

  - Fit probability distributions (such as Poisson, Negative Binomial, etc.) to the runs scored and wickets taken data of the top performers.
  - Interpret the fitted distributions to understand the typical performance patterns of these elite players.

4. Performance vs. Salary Relationship:
  - Investigate the relationship between player performance metrics (runs, wickets, etc.) and their corresponding salaries.
  - Utilize statistical techniques (regression analysis, correlation, etc.) to quantify and interpret this relationship.

5. Tools and Methodology:

- Programming Languages: R and Python will be employed for data extraction, manipulation, visualization, and statistical analysis.

- Data Handling: CSV or structured data formats will be used for storing and processing IPL statistics.

- Statistical Analysis: Descriptive statistics, probability distributions, regression analysis, and correlation techniques will be applied to explore relationships and derive meaningful insights.

Assignment Outcomes:

This assignment aims to provide a comprehensive analysis of IPL player performances across recent seasons, utilizing statistical tools and techniques to uncover patterns in batting, bowling, and player salary dynamics. By the end of this analysis, we seek to gain deeper insights into what makes top IPL performers stand out and how their performance metrics relate to their financial compensation.

Analysis using R

1) Extracting Data in R

```
 # Set the working directory and verify it
setwd('D:\\CHRIST\\Boot camp\\DATA')
getwd()

# Function to install and load libraries
install_and_load <- function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package, dependencies = TRUE)
    library(package, character.only = TRUE)
  }
}

# Load required libraries
libraries <- c("dplyr", "readr", "readxl", "tidyr", "ggplot2", "BSDA", "glue", "fitdistrplus",
"stringdist")
```

```
lapply(libraries, install_and_load)


# Reading the files into R
ipl_bbb <- read_csv('IPL_ball_by_ball_updated till 2024.csv')
ipl_salary <- read_excel('IPL SALARIES 2024.xlsx')
```

2) Arranging the data:

```
# Summarising player runs and wickets
player_runs <- grouped_data %>%
  group_by(Season, Striker) %>%
  summarise(runs_scored = sum(runs_scored, na.rm = TRUE)) %>%
  ungroup()

player_wickets <- grouped_data %>%
  group_by(Season, Bowler) %>%
  summarise(wicket_confirmation = sum(wicket_confirmation, na.rm = TRUE)) %>%
  ungroup()

# Get top 3 run-getters and bottom 3 wicket-takers per season
top_run_getters <- player_runs %>%
  group_by(Season) %>%
  top_n(3, runs_scored) %>%
  ungroup()

bottom_wicket_takers <- player_wickets %>%
  group_by(Season) %>%
  top_n(3, wicket_confirmation) %>%
  ungroup()
```

Explanation:

Calculates total runs (player_runs) and wickets (player_wickets) per season for each player (striker and bowler).
Identifies top 3 run-getters (top_run_getters) and bottom 3 wicket-takers (bottom_wicket_takers) for each season based on aggregated statistics.

3) Working with Dataframes and Data Manipulation

```
# Create a dataframe for match id and year
ipl_year_id <- data.frame(
  id = ipl_bbb$`Match id`,
  year = format(as.Date(ipl_bbb$Date, format = "%d/%m/%Y"), "%Y")
)
```

4) Create a copy of ipl_bbb dataframe and add a year column

```
ipl_bbbc <- ipl_bbb %>%
  mutate(year = format(as.Date(Date, format = "%d/%m/%Y"), "%Y"))
```

Explanation:

Creates a dataframe ipl_year_id mapping match IDs to years extracted from ipl_bbb. Creates a copy ipl_bbbc of ipl_bbb with an additional year column formatted from the Date column.
 Statistical Modeling with Distributions

5) Define a function to get the best distribution

```
get_best_distribution <- function(data) {
  dist_names <- c('norm', 'lnorm', 'gamma', 'weibull', 'exponential', 'logis', 'cauchy')
  dist_results <- list()
  params <- list()
  for (dist_name in dist_names) {
    fit <- fitdist(data, dist_name)
    ks_test <- ks.test(data, dist_name, fit$estimate)
    p_value <- ks_test$p.value
    dist_results[[dist_name]] <- p_value
    params[[dist_name]] <- fit$estimate
  }
  best_dist <- names(which.max(unlist(dist_results)))
  best_p <- max(unlist(dist_results))
  cat("\nBest fitting distribution:", best_dist, "\n")
  cat("Best p value:", best_p, "\n")
  cat("Parameters for the best fit:", params[[best_dist]], "\n")
  return(list(best_dist, best_p, params[[best_dist]]))
}
```

Explanation:
Defines get_best_distribution function to fit various probability distributions ('norm', 'lnorm', 'gamma', 'weibull', 'exponential', 'logis', 'cauchy') to data and selects the best fit based on Kolmogorov-Smirnov test p-values.

6) Data Aggregation and Analysis

```
# Total runs each year
total_run_each_year <- ipl_bbbc %>%
  group_by(year, Striker) %>%
  summarise(runs_scored = sum(runs_scored, na.rm = TRUE)) %>%
  ungroup() %>%
  arrange(year, desc(runs_scored))

list_top_batsman_last_three_year <- list()
for (i in unique(total_run_each_year$year)[1:3]) {
  list_top_batsman_last_three_year[[as.character(i)]] <- total_run_each_year %>%
    filter(year == i) %>%
    top_n(3, runs_scored) %>%
    pull(Striker)
}
```

Explanation:

Aggregates total runs per year for each batsman (total_run_each_year).
Creates a list of top 3 batsmen (list_top_batsman_last_three_year) for the last three years based on total runs scored.

7) Statistical Distribution Analysis

```
# Get best distribution for top batsmen in the last three years
for (key in names(list_top_batsman_last_three_year)) {
  for (Striker in list_top_batsman_last_three_year[[key]]) {
    cat("***********************\n")
    cat("Year:", key, " Batsman:", Striker, "\n")
    get_best_distribution(runs %>% filter(Striker == Striker) %>% pull(runs_scored))
    cat("\n\n")
  }
}
```

Explanation:

Applies get_best_distribution function to top batsmen's runs data over the last three years to determine the best fitting distribution for each batsman.
6) Data Integration and Salary Analysis
# Create a new column in ipl_salary with matched names from R2024
ipl_salary$Matched_Player <- sapply(ipl_salary$Player, function(x) match_names(x, R2024$Striker))

8) Merge the dataframes on the matched names
df_merged <- merge(ipl_salary, R2024, by.x = "Matched_Player", by.y = "Striker")

# Calculate the correlation between Salary and Runs
correlation <- cor(df_merged$Rs, df_merged$runs_scored, use = "complete.obs")

Explanation:

correlation <- cor(df_merged$Rs, df_merged$runs_scored, use = "complete.obs")
This line calculates the Pearson correlation coefficient between the salary (Rs) and runs scored (runs_scored) for players in df_merged dataframe.
The use = "complete.obs" argument ensures that only complete pairs of observations are used in the calculation.

Analysis using Python:

1) Extracting the necessary information from the provided files in Python:

import pandas as pd

# Load the required files
ipl_bbb = pd.read_csv('IPL_ball_by_ball_updated till 2024.csv', low_memory=False)
ipl_salary = pd.read_excel('IPL SALARIES 2024.xlsx')

2)Organize the data round-wise and calculate batsman, balls, runs, and wickets per player per match. Identify the top three run-getters and wicket-takers in each IPL round:

In this step, we organize the data round-wise and identify the top three run-getters and wicket-takers in each IPL round.
# Group data by season, innings, striker, and bowler
grouped_data = ipl_bbb.groupby(['Season', 'Innings No', 'Striker', 'Bowler']).agg({'runs_scored': sum, 'wicket_confirmation': sum}).reset_index()

# Calculate total runs scored by each player in a season
player_runs = grouped_data.groupby(['Season', 'Striker'])['runs_scored'].sum().reset_index()

# Calculate total wickets taken by each player in a season
player_wickets = grouped_data.groupby(['Season', 'Bowler'])['wicket_confirmation'].sum().reset_index()

```
# Identify top three run-getters and wicket-takers in each IPL round
top_run_getters = player_runs.groupby('Season').apply(lambda x: x.nlargest(3, 'runs_scored')).reset_index(drop=True)
top_wicket_takers = player_wickets.groupby('Season').apply(lambda x: x.nlargest(3, 'wicket_confirmation')).reset_index(drop=True)
```

Interpretation:

By grouping the data by season, innings, striker, and bowler, we can analyze the performance metrics (runs scored and wickets taken) of each player in the IPL matches. Identifying the top three run-getters and wicket-takers in each IPL round helps in recognizing the most impactful players based on their performance statistics.

3)Fit the most suitable distribution for runs scored and wickets taken by the top three batsmen and bowlers in the last three IPL tournaments:

Here, we fit the most suitable distribution for runs scored and wickets taken by the top three batsmen and bowlers in the last three IPL tournaments.

```
# Loop through top players and fit distributions for their runs scored and wickets taken
for player in top_players:
    runs_data = get_runs_data(player)  # Function to get runs data for the player
    wickets_data = get_wickets_data(player)  # Function to get wickets data for the player

    best_dist_runs, _, _ = get_best_distribution(runs_data)  # Fit distribution for runs scored
    best_dist_wickets, _, _ = get_best_distribution(wickets_data)  # Fit distribution for wickets taken
```

Interpretation:

Fitting distributions for runs scored and wickets taken by the top players allows us to understand the underlying patterns in their performance data.

By determining the best-fitting distribution for each player, we can potentially model their future performance and make predictions based on historical data.

4) Analyze the relationship between a player's performance and the salary he receives using the available data:

In this step, we analyze the relationship between a player's performance and the salary he receives.

```
from fuzzywuzzy import process

# Function to match player names
def match_names(name, names_list):
    match, score = process.extractOne(name, names_list)
    return match if score >= 80 else None

# Match player names and merge the DataFrames
df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x, df_runs['Striker'].tolist()))
df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Striker')

# Calculate the correlation between salary and runs scored
correlation = df_merged['Rs'].corr(df_merged['runs_scored'])
print("Correlation between Salary and Runs:", correlation)
```

Interpretation:

Matching player names between salary and performance datasets enables us to analyze the relationship between player performance and the salary they receive.

The correlation between a player's performance (runs scored) and their salary indicates the degree of association between these two variables. A positive correlation suggests that higher performance is associated with higher salaries.

Conclusion:

Explanation of correlation output:

- Magnitude:

The correlation coefficient (r) ranges from -1 to 1.

A value of 0.306 indicates a positive, but relatively weak, linear relationship between the salary of IPL players and the runs they scored in the 2024 season.

- Direction:

Since the correlation coefficient is positive (0.306), it suggests that as the salary of a player increases, the runs scored by the player tend to increase as well, though the relationship is not very strong.

- Strength:

Correlation coefficients are categorized into different levels of strength:

0.0 to 0.1: No or negligible correlation
0.1 to 0.3: Weak correlation
0.3 to 0.5: Moderate correlation
0.5 to 0.7: Strong correlation
0.7 to 1.0: Very strong correlation
With a correlation coefficient of 0.306, the relationship between salary and runs scored is on the weaker side of moderate.

- Implications:

While there is some positive relationship between higher salaries and more runs scored, the weak to moderate strength suggests that other factors also play significant roles in determining the number of runs a player scores. Salary alone does not fully explain the performance of a player in terms of runs.

- Practical Considerations:

In practical terms, this weak to moderate correlation implies that teams and management should consider multiple factors (e.g., player form, fitness, experience, match conditions) along with salary when assessing a player's potential contribution to the team.