VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A1b: Preliminary preparation and analysis of data- Descriptive statistics

PUSHPAK DEVAKI

V01108254

Date of Submission: 18-06-2024

CONTENTS

INTRODUCTION

The data procured from IPL organisers is analysed, in order to determine the top and bottom three run getters and wicket takers. To get the data we need for analysis, we will clean and alter the dataset. The dataset has been loaded into R/Python, a potent statistical programming language renowned for its effectiveness in handling and analysing big datasets.

Our goals include analysing the relationship between players on field performance and their salaries and the correlation between same, over last three IPL tournaments. The analysis also includes players remuneration and their cricket contributions in the matches.

OBJECTIVES

a) Record detailed statistics per match, including batsman, ball, runs, and wickets per player.

b) Analyze and establish the relationship between a player's on-field performance and their salary.

c) Provide insights into how a player's performance correlates with their remuneration in the IPL.

RESULTS & INTERPRETATION

a) Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match. Indicate the top three run-getters and top three wicket-takers in each IPL round. (From R)

Code:

```
> # Summarise 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()


> # Sort player runs for season 2023
> player_runs_2023 <- player_runs %>%
+   filter(Season == '2023') %>%
+   arrange(desc(runs_scored))
>
> # 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()
```

Result:

```
> print(top_run_getters)
# A tibble: 51 × 3
   Season  Striker        runs_scored
   <chr>   <chr>                <dbl>
 1 2007/08 G Gambhir              534
 2 2007/08 SE Marsh               616
 3 2007/08 ST Jayasuriya          514
 4 2009    AB de Villiers         465
 5 2009    AC Gilchrist           495
 6 2009    ML Hayden              572
 7 2009/10 JH Kallis              572
 8 2009/10 SK Raina               528
 9 2009/10 SR Tendulkar           618
10 2011    CH Gayle               608
```

```
> print(bottom_wicket_takers)
# A tibble: 58 × 3
   Season  Bowler          wicket_confirmation
   <chr>   <chr>                         <dbl>
 1 2007/08 IK Pathan                        20
 2 2007/08 JA Morkel                        20
 3 2007/08 SK Warne                         20
 4 2007/08 SR Watson                        20
 5 2007/08 Sohail Tanvir                    24
 6 2009    A Kumble                         22
 7 2009    A Nehra                          22
 8 2009    RP Singh                         26
 9 2009/10 A Mishra                         20
10 2009/10 Harbhajan Singh                  20
```

Interpretation:

From the above dataset, we can see that the top three run getters are g Gambhir, SE Marsh, ST Jayasuriya with the runs as 534, 616 and 514 respectively. On the other hand, the top three wicket takers are IK Pathan, JA Morkel and SK Warne by taking 20 wickets each respectively.

B) Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments.

(Code from R)

```
> # Define a function to get the best distribution
> get_best_distribution <- function(data) {
+   dist_names <- c('norm', 'lnorm', 'gamma', 'weibull', 'exponential', 'l
ogis', '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
+     cat("p value for", dist_name, "=", p_value, "\n")
+     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]]))
+ }
```

```
> # Function to fit the best distribution
> get_best_distribution <- function(data) {
+   # Fit different distributions
+   fit_norm <- fitdist(data, "norm")
+   fit_pois <- fitdist(data, "pois")
+   fit_exp <- fitdist(data, "exp")
+
+   # Compare the distributions
+   gof_stat <- gofstat(list(fit_norm, fit_pois, fit_exp), fitnames = c("N
ormal", "Poisson", "Exponential"))
+
+   # Print the goodness-of-fit statistics
+   print(gof_stat)
+
+   # Return the best fit distribution
+   best_fit <- names(which.min(gof_stat$aic))
+   return(best_fit)
+ }
>
> # Fit the distribution to Q de Kock's runs scored and get the best distr
ibution
> best_distribution <- get_best_distribution(Q_de_Kock_runs)
```

<u>Result:</u>

```
Goodness-of-fit statistics
                                Normal   Poisson Exponential
Kolmogorov-Smirnov statistic 0.1280142 0.4254026   0.0805889
Cramer-von Mises statistic   0.4175224 6.0350887   0.1594708
Anderson-Darling statistic   2.6398461       Inf         Inf

Goodness-of-fit criteria
                                 Normal  Poisson Exponential
Akaike's Information Criterion 989.2156 2914.264    925.9846
Bayesian Information Criterion 994.5235 2916.918    928.6386
```

Interpretation:

The low values of KS and CVM which is 0.08 and 0.15 respectively, suggests that exponential distribution provides the best fit.

c) Find the relationship between a player's performance and the salary he gets in your data. (Code from Python)

```
# Create a new column in df_salary with matched names from df_runs

df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x,
df_runs['Striker'].tolist()))

# Merge the DataFrames on the matched names

df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Striker')

df_merged.info()

# Calculate the correlation

correlation = df_merged['Rs'].corr(df_merged['runs_scored'])

print("Correlation between Salary and Runs:", correlation)
```

Result:

```
Correlation between Salary and Runs: 0.30612483765821674
```

Interpretation:

As we can see the correlation coefficient is 0.3061, this indicates a positive relationship between salary and runs. So, the relation suggests if the salary increases the run score is also likely to improve.

RECOMMENDATIONS

- As we have figured out the correlation between the salary and runs made, it would be very helpful for the franchise owners to evaluate the players performance and choose them accordingly.
- This will also help them in deciding the budget allocation for the players and affordability for each player.
- ROI can be effectively calculated and the players can be chosen wisely according to these predictions.