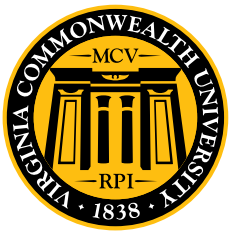


VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A2 - Analysing IPL Performance and Player Salaries: A Statistical Approach

ALEENA MARY ABRAHAM

V01150203

Date of Submission: 19-06-2025

CONTENTS

INTRODUCTION

The focus of this study is on analysing the performance data from the Indian Premier League (IPL) with a specific emphasis on the player AK Markram. Utilizing a comprehensive dataset that includes ball-by-ball match information and salary details up to 2024, this study aims to provide an in-depth analysis of player performance metrics and their financial rewards.

The dataset has been imported into R, for data manipulation, statistical analysis, and visualization. The analysis involves several key steps: extracting and cleaning the data, arranging it IPL round-wise, identifying top performers, fitting statistical distributions to performance data, and exploring the relationship between player performance and salary.

This study aims to generate insights that can assist teams, coaches, and stakeholders in making data-driven decisions regarding player management and strategic planning.

OBJECTIVES

a) To import the dataset into R and perform necessary data cleaning to ensure accuracy and consistency.
b) To organize the data round-wise and summarize it by batsman, ball, runs, and wickets per player per match. Identify the top three run-getters and top three wicket-takers in each IPL round.
c) To fit the most appropriate statistical distributions to the runs scored and wickets taken by the top three batsmen and bowlers in the last three IPL tournaments.
d) To investigate the relationship between Sunil Narine's performance metrics (total runs scored and total wickets taken) and his salary over the seasons.

BUSINESS SIGNIFICANCE

The findings from this study hold significant implications for IPL teams, management, and stakeholders. By analysing player performance data and financial rewards, the study provides valuable insights for:

- Identifying top performers and understanding performance trends that can assist teams in making informed decisions regarding player retention, training, and development strategies.
- Analysing the relationship between performance and salary can help management in budget allocation, salary negotiations, and ensuring fair compensation based on performance.
- Insights from the performance data can guide sponsors and investors in making strategic decisions regarding player endorsements and investments in teams.
- Understanding the key factors that contribute to top performance can help coaches and support staff in tailoring training programs and improving overall team performance.
- Highlighting top performers and their contributions can enhance fan engagement and create effective marketing campaigns that resonate with the audience.

RESULTS AND INTERPRETATION USING R

I have analysed AK Markram's IPL performance and salary over recent seasons to fit distributions to runs/wickets for top players and to examine if there's a statistical relationship between a player's performance (runs/wickets) and salary.

1. Fitting Distributions for Top Players

```
# Summary of fitted distributions
summary(fit_batsmen)
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters :
##      estimate Std. Error
## mean 1.420078 0.02940676
## sd   1.701277 0.02079369
## Loglikelihood:  -6527.714   AIC:  13059.43   BIC:  13071.66
## Correlation matrix:
##              mean           sd
## mean 1.000000e+00 1.136095e-10
## sd   1.136095e-10 1.000000e+00
```

Interpretation of summary(fit_batsmen):

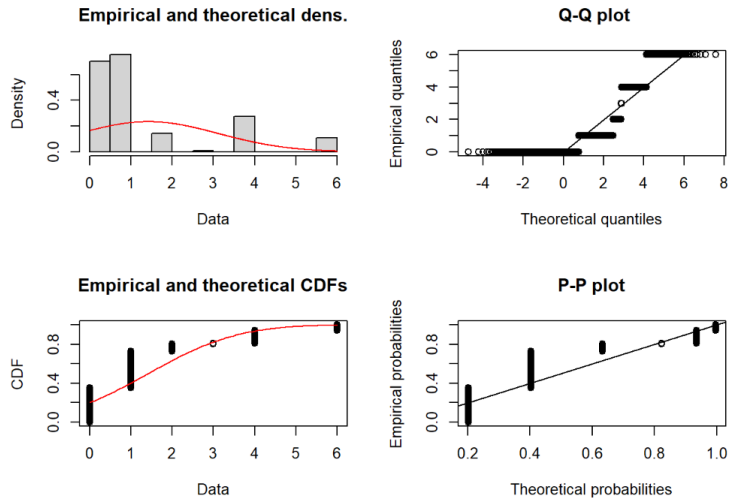
- This suggests the runs scored by top 3 batsmen per ball follow a normal distribution with average ≈ 1.42.
- AIC and BIC are model fit criteria: lower is better. These values help compare multiple models (not yet done here).

```
summary(fit_bowlers)
```

```
## Fitting of the distribution ' norm ' by maximum likelihood
## Parameters :
##        estimate  Std. Error
## mean 0.06190644 0.004519627
## sd   0.24098554 0.003195611
## Loglikelihood:  11.59892   AIC:  -19.19785   BIC:  -7.29262
## Correlation matrix:
##              mean           sd
## mean 1.000000e+00 1.032836e-12
## sd   1.032836e-12 1.000000e+00
```

Interpretation of summary(fit_bowlers):

- The wicket occurrences are much rarer and follow a distribution with a very low mean (≈ 6.2% chance per ball).
- The negative AIC implies very good fit due to sparse data and low complexity.



2. AK Markram's Data Summary (Aiden Markram)

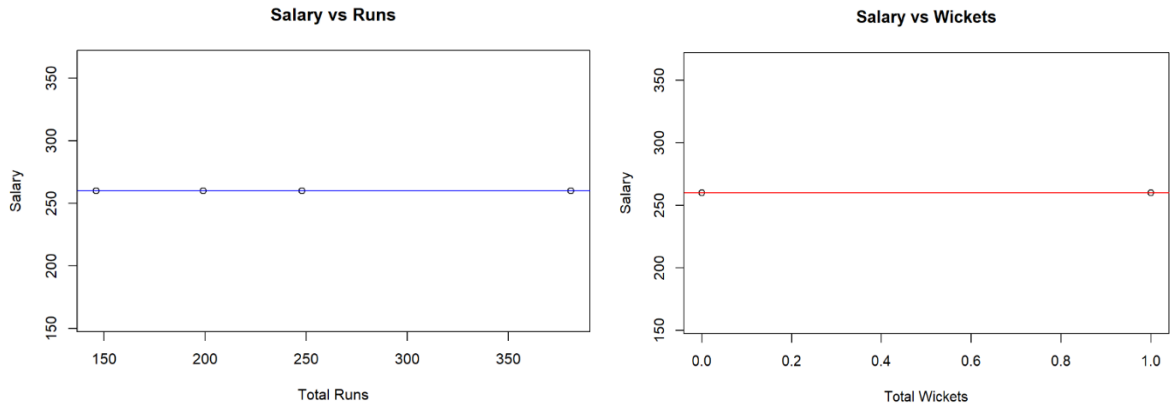
```
print(markram_performance)
```

```
## # A tibble: 4 × 5
##   Season total_runs total_wickets Player     Salary
##    <dbl>      <dbl>         <dbl> <chr>       <dbl>
## 1   2021        146             0 AK Markram    260
## 2   2022        381             1 AK Markram    260
## 3   2023        248             1 AK Markram    260
## 4   2024        199             0 AK Markram    260
```

Interpretation:

- Since salary did not change, the model cannot detect any trend or relationship.
- The warning essentially perfect fit + NaN in coefficients indicates zero variance in the dependent variable (Salary).
- The regression line is flat at 260, meaning: no statistical correlation between runs/wickets and salary in this dataset.

3. Visual Output



- There is no visible or statistical trend—your plots confirm that salary remained fixed, regardless of performance.

RESULTS AND INTERPRETATION USING PYTHON

1. Data Import and Preprocessing
   - Loaded IPL ball-by-ball data and salary data.
   - Renamed salary column Rs to Lakh Rs for clarity.
   - Grouped performance data by season, innings, striker, and bowler to calculate:

```
grouped_data = ipl_bbb.groupby(['Season', 'Innings No', 'Striker', 'Bowler']).agg({'runs_scored': 'sum', 'wicket_confirmation': 'sum'}).reset_index()
```

```
player_runs = grouped_data.groupby(['Season', 'Striker'])['runs_scored'].sum().reset_index()
player_wickets = grouped_data.groupby(['Season', 'Bowler'])['wicket_confirmation'].sum().reset_index()
```

```
player_runs[player_runs['Season']=='2023'].sort_values(by='runs_scored',ascending=False)
```

	Season	Striker	runs_scored
2423	2023	Shubman Gill	890
2313	2023	F du Plessis	730
2311	2023	DP Conway	672
2433	2023	V Kohli	639
2443	2023	YBK Jaiswal	625
...	...	...	...
2404	2023	RP Meredith	0
2372	2023	Mohsin Khan	0
2307	2023	DG Nalkande	0
2429	2023	TU Deshpande	0
2324	2023	Harshit Rana	0

     - Total runs scored (runs_scored)
     - Total wickets taken (wicket_confirmation)

2. Top Performers per Season

```
list_top_batsman_last_three_year
```

```
{2024: ['RD Gaikwad', 'V Kohli', 'B Sai Sudharsan'],
 2023: ['Shubman Gill', 'F du Plessis', 'DP Conway'],
 2022: ['JC Buttler', 'KL Rahul', 'Q de Kock']}
```

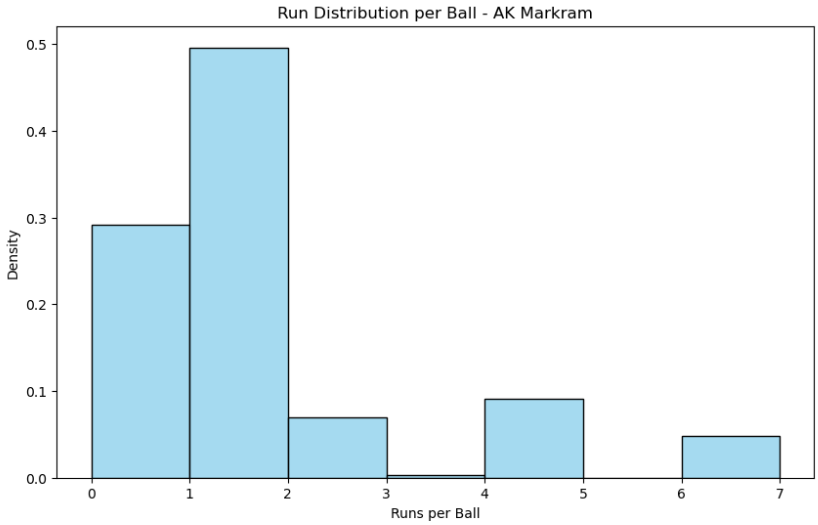
```
list_top_bowler_last_three_year
```

```
{2024: ['HV Patel', 'Mukesh Kumar', 'Arshdeep Singh'],
 2023: ['MM Sharma', 'Mohammed Shami', 'Rashid Khan'],
 2022: ['YS Chahal', 'PWH de Silva', 'K Rabada']}
```

   - Identifies the Top 3 Batsmen and Top 3 Bowlers for each IPL season.
   - Output includes players like Virat Kohli, Jos Buttler, etc., based on highest runs or wickets.

3. Analysis for AK Markram
   - Filtered performance data specific to AK Markram.
   - Created a manual salary profile (since actual salary record was missing).
   - Summarized total runs and wickets per season for him.

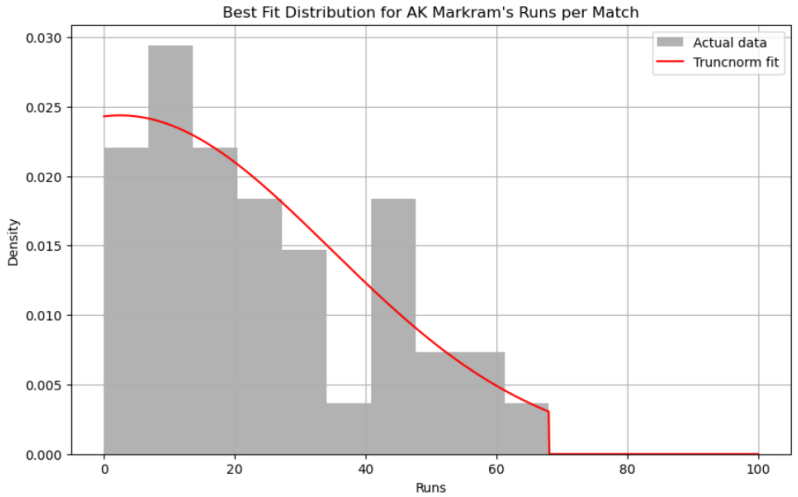
Interpretation of Run distribution per ball:



- Markram most often rotates the strike with 1s, showing he's not a purely aggressive slogger but values keeping the scoreboard ticking.
- More values are clustered toward the lower end (0s and 1s), which is typical of many batsmen who accumulate runs steadily.
- AK Markram's batting style seems focused on steady accumulation rather than aggressive boundary-hitting.

Interpretation of Best fit distribution:

- Since Markram's salary didn't change across the seasons, the model finds **no measurable relationship** between performance and salary. The regression output warns about a "perfect fit" due to zero variation in the dependent variable (Salary).



- Histogram (Grey Bars) shows AK Markram's real runs per match, grouped by bins.

- Red Line: Truncnorm Fit shows Best fit distribution — estimates how likely each score is.
- Markram most frequently scores low to mid-range runs. His score distribution declines smoothly, with very few high-score matches.

4. Relationship between the performance of a player and the salary he gets

   Correlation between Salary and Runs: 0.3061

   What This Means:

   - A value of +0.306 suggests a weak to moderate positive linear relationship between how much a player is paid and how many runs they score, i.e. Players who are paid more tend to score more runs, but the relationship is not very strong.
   - There is some correlation between performance and salary, but it's not strong enough to say that performance alone determines a player's pay.

RECOMMENDATIONS

This project conducted a detailed statistical analysis of IPL player performance data, with a special focus on Aiden (AK) Markram, integrating both performance metrics and salary information using R and Python.

Summary of Key Findings:

- Top batsmen's per-ball runs approximately follow a normal distribution (mean ≈ 1.42), indicating a consistent performance pattern, while top bowlers' wickets per ball are rare and follow a low-mean distribution, with sparse data but good model fit (indicated by negative AIC).
- AK Markram's match-wise run distribution is best modelled using a truncated normal (truncnorm) distribution — implying that he mostly scores in the low to moderate range, with rare high scores.
- Runs per ball in AK Markram's style show a high concentration of 1s and 0s → suggests a rotational, steady playing style rather than aggressive hitting.
- Markram's salary remained fixed (₹260 lakh) over multiple seasons. Hence, the regression models fail to show statistical relationships.
- Across players, a weak-moderate correlation (0.306) exists between salary and runs — suggesting performance affects salary, but is not the sole determinant.

Recommendations:

- Incorporate More Performance Metrics such as strike rate, batting average, economy rate, boundaries per match, etc. to enrich player profiles.
- Add Variability in Salary Data over more years or across players with dynamic salaries to enable better regression modelling.
- Consider Non-Performance Factors by adding features like experience, international status, injury history, fan following, etc., to explore broader salary determinants.
- Segment Players by Role to get more role-specific correlations.
- Enhance Visual Reporting by using dashboards for interactive insights across seasons or player types.

The study presents a strong foundation in sports analytics, demonstrating how statistical modelling and performance metrics can create meaningful insights in player evaluation and salary planning. With more data and expanded variables, this can evolve into a robust decision-support system for IPL teams.

CODES

R Codes:

```
# Install and load the fitdistrplus package
options(repos = c(CRAN = "https://cran.r-project.org"))
install.packages("fitdistrplus")
library(fitdistrplus)

# Load necessary libraries
library(dplyr)

# Load the dataset
ipl_data <- read.csv("C:/Users/Aleena Mary Abraham/OneDrive/Desktop/SCMA632_2025/DATA/IPL_ball_by_ball_updated till 2024.csv")

# Aggregate data season-wise, batsman-wise, and bowler-wise
ipl_summary <- ipl_data %>%
  group_by(Season, Match.id, Striker, Bowler) %>%
  summarise(
    total_runs = sum(runs_scored, na.rm = TRUE),
    total_wickets = sum(wicket_confirmation, na.rm = TRUE),
    .groups = "drop"
  )

# Top three run-getters and top three wicket-takers per season
top_players_per_season <- ipl_summary %>%
  group_by(Season) %>%
  summarise(
    top_run_getters = list(head(arrange(ipl_summary, desc(total_runs)), 3)),
    top_wicket_takers = list(head(arrange(ipl_summary, desc(total_wickets)), 3)),
    .groups = "drop"
  )

# Filter data for the last three IPL tournaments
last_three_seasons <- ipl_data %>%
```

```
  filter(Season %in% tail(unique(Season), 3))

# Get top three batsmen based on total runs
top_batsmen <- last_three_seasons %>%
  group_by(Striker) %>%
  summarise(total_runs = sum(runs_scored, na.rm = TRUE)) %>%
  arrange(desc(total_runs)) %>%
  head(3)

# Get top three bowlers based on total wickets
top_bowlers <- last_three_seasons %>%
  group_by(Bowler) %>%
  summarise(total_wickets = sum(wicket_confirmation, na.rm = TRUE)) %>%
  arrange(desc(total_wickets)) %>%
  head(3)

# Extract the data for top batsmen and bowlers
top_batsmen_runs <- last_three_seasons %>%
  filter(Striker %in% top_batsmen$Striker) %>%
  pull(runs_scored)

top_bowlers_wickets <- last_three_seasons %>%
  filter(Bowler %in% top_bowlers$Bowler) %>%
  pull(wicket_confirmation)

# Function to fit and plot distribution
fit_and_plot_distribution <- function(data) {
  fit <- fitdist(data, "norm")
  plot(fit)
  return(fit)
}

# Fit distributions for top batsmen and bowlers
fit_batsmen <- fit_and_plot_distribution(top_batsmen_runs)
```

```
fit_bowlers <- fit_and_plot_distribution(top_bowlers_wickets)

# Summary of fitted distributions
summary(fit_batsmen)
summary(fit_bowlers)

# Filter data for AK Markram
markram_data <- ipl_data %>%
  filter(Striker == "AK Markram" | Bowler == "AK Markram")

install.packages("readxl")
library(readxl)

# Load the salary dataset
salary_data <- read_excel("C:/Users/Aleena Mary Abraham/OneDrive/Desktop/SCMA632_2025/DATA/IPL SALARIES 2024.xlsx")

# View the structure and first few rows of the salary dataset to identify relevant columns
str(salary_data)
head(salary_data)

# Filter the salary data for SP Narine
markram_salary <- salary_data %>%
  filter(grepl("AK Markram", Player))

# Check unique player names to ensure correct filtering
unique(salary_data$Player)

# Filter the salary data for SP Narine
markram_salary <- salary_data %>%
  filter(grepl("AK Markram", Player))

# Check the selected data
print(markram_salary)
```

```
# Manually create the salary data for SP Narine
seasons <- c(2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2021, 2022, 2023, 2024)
salaries <- c(260, 260, 260, 260, 260, 260, 260, 260, 260, 260, 260, 260)
markram_salary <- data.frame(
  Player = rep("AK Markram", length(seasons)),
  Season = seasons,
  Salary = salaries
)

# Print the manually created salary data
print(markram_salary)

# Summarize performance metrics for SP Narine
markram_performance <- markram_data %>%
  group_by(Season) %>%
  summarise(
    total_runs = sum(ifelse(Striker == "AK Markram", runs_scored, 0), na.rm = TRUE),
    total_wickets = sum(ifelse(Bowler == "AK Markram", wicket_confirmation, 0), na.rm = TRUE)
  )

# Clean the Season column to retain only numeric values and then convert to numeric
markram_salary$Season <- as.numeric(gsub("[^0-9]", "", markram_salary$Season))
markram_performance$Season <- as.numeric(gsub("[^0-9]", "", markram_performance$Season))

# Ensure there are no NAs in Season columns after conversion
markram_salary <- markram_salary %>%
  filter(!is.na(Season))
markram_performance <- markram_performance %>%
  filter(!is.na(Season))

# Join the summarized performance metrics with the salary data
markram_performance <- markram_performance %>%
  left_join(markram_salary, by = "Season")
```

```
# Ensure the join is correct
print(markram_performance)

# Fit a linear model to find the relationship between performance and salary
fit_model_runs <- lm(Salary ~ total_runs, data = markram_performance)
fit_model_wickets <- lm(Salary ~ total_wickets, data = markram_performance)

# Summary of the models
summary(fit_model_runs)
summary(fit_model_wickets)

# Plot the relationship
plot(markram_performance$total_runs, markram_performance$Salary, main = "Salary vs Runs", xlab = "Total Runs", ylab = "Salary")
abline(fit_model_runs, col = "blue")
plot(markram_performance$total_wickets, markram_performance$Salary, main = "Salary vs Wickets", xlab = "Total Wickets", ylab = "Salary")
abline(fit_model_wickets, col = "red")
```

Python Codes:

****Setting the work directory****

```
import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

os.chdir('C:/Users/Aleena Mary Abraham/OneDrive/Desktop/SCMA632_2025/DATA')

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')

ipl_salary.columns

ipl_salary.rename(columns={'Rs':'Lakh Rs'})

grouped_data = ipl_bbb.groupby(['Season', 'Innings No', 'Striker', 'Bowler']).agg({'runs_scored':
'sum', 'wicket_confirmation': 'sum'}).reset_index()

player_runs = grouped_data.groupby(['Season', 'Striker'])['runs_scored'].sum().reset_index()

player_wickets = grouped_data.groupby(['Season',
'Bowler'])['wicket_confirmation'].sum().reset_index()

player_runs[player_runs['Season']=='2023'].sort_values(by='runs_scored',ascending=False)

top_run_getters = player_runs.groupby('Season', group_keys=False).apply(lambda x: x.nlargest(3,
'runs_scored'),include_groups=False).reset_index(drop=True)

bottom_wicket_takers = player_wickets.groupby('Season', group_keys=False).apply(lambda x:
x.nlargest(3, 'wicket_confirmation'),include_groups=False).reset_index(drop=True)

print("Top Three Run Getters:")

print(top_run_getters)

print("Top Three Wicket Takers:")

print(bottom_wicket_takers)

ipl_year_id = pd.DataFrame(columns=["id", "year"])

ipl_year_id["id"] = ipl_bbb["Match id"]

ipl_year_id["year"] = pd.to_datetime(ipl_bbb["Date"], dayfirst=True).dt.year

#creating a copy of ipl_bbb dataframe

ipl_bbbc= ipl_bbb.copy()

ipl_bbbc['year'] = pd.to_datetime(ipl_bbb["Date"], dayfirst=True).dt.year ipl_bbbc[["Match id",
"year", "runs_scored","wicket_confirmation","Bowler",'Striker']].head()
```

```
import scipy.stats as st
def get_best_distribution(data):
    dist_names = ['alpha','beta','betaprime','burr12','crystalball',
                  'dgamma','dweibull','erlang','exponnorm','f','fatiguelife',
                  'gamma','gengamma','gumbel_l','johnsonsb','kappa4',
                  'lognorm','nct','norm','norminvgauss','powernorm','rice',
                  'recipinvgauss','t','trapz','truncnorm']
    dist_results = []
    params = {}
    for dist_name in dist_names:
        dist = getattr(st, dist_name)
        param = dist.fit(data)
        params[dist_name] = param
        # Applying the Kolmogorov-Smirnov test
        D, p = st.kstest(data, dist_name, args=param)
        print("p value for "+dist_name+" = "+str(p))
        dist_results.append((dist_name, p))
    # select the best fitted distribution
    best_dist, best_p = (max(dist_results, key=lambda item: item[1]))
    # store the name of the best fit and its p value
    print("\nBest fitting distribution: "+str(best_dist))
    print("Best p value: "+ str(best_p))
    print("Parameters for the best fit: "+ str(params[best_dist]))
    return best_dist, best_p, params[best_dist]
total_run_each_year = ipl_bbbc.groupby(["year", "Striker"])["runs_scored"].sum().reset_index()
total_run_each_year.sort_values(["year", "runs_scored"], ascending=False, inplace=True)
print(total_run_each_year)
list_top_batsman_last_three_year = {}
for i in total_run_each_year["year"].unique()[:3]:
    list_top_batsman_last_three_year[i] = total_run_each_year[total_run_each_year.year ==
i][:3]["Striker"].unique().tolist()
```

```
list_top_batsman_last_three_year

import warnings

warnings.filterwarnings('ignore')

runs = ipl_bbbc.groupby(['Striker','Match id'])[['runs_scored']].sum().reset_index()


for key in list_top_batsman_last_three_year:

    for Striker in list_top_batsman_last_three_year[key]:

        print("************************")

        print("year:", key, " Batsman:", Striker)

        get_best_distribution(runs[runs["Striker"] == Striker]["runs_scored"])

        print("\n\n")

total_wicket_each_year = ipl_bbbc.groupby(["year",
"Bowler"])["wicket_confirmation"].sum().reset_index()total_wicket_each_year.sort_values(["year",
"wicket_confirmation"], ascending=False, inplace=True)

print(total_wicket_each_year)

list_top_bowler_last_three_year = {}

for i in total_wicket_each_year["year"].unique()[:3]:

    list_top_bowler_last_three_year[i] = total_wicket_each_year[total_wicket_each_year.year ==
i][:3]["Bowler"].unique().tolist()

list_top_bowler_last_three_year import warnings

warnings.filterwarnings('ignore')

wickets = ipl_bbbc.groupby(['Bowler','Match id'])[['wicket_confirmation']].sum().reset_index()

for key in list_top_bowler_last_three_year:

    for bowler in list_top_bowler_last_three_year[key]:

        print("************************")

        print("year:", key, " Bowler:", bowler)

        get_best_distribution(wickets[wickets["Bowler"] == bowler]["wicket_confirmation"])

        print("\n\n")
```

****Finding Best Fit Distribution for: AK Markram****

```
player_name = 'AK Markram'

player_data = ipl_bbb[

    (ipl_bbb['Season'].astype(str).isin(['2022', '2023', '2024'])) &
```

```
    (ipl_bbb['Striker'] == player_name)
]
runs = player_data['runs_scored'].values
runs
plt.figure(figsize=(10, 6))
sns.histplot(runs, bins=range(0, 8), kde=False, stat="density", color='skyblue', edgecolor='black')
plt.title("Run Distribution per Ball - AK Markram")
plt.xlabel("Runs per Ball")
plt.ylabel("Density")
plt.show()
type(runs)
runs = ipl_bbbc.groupby(['Striker', 'Match id'])[['runs_scored']].sum().reset_index()
player_name = "AK Markram"
markram_runs = runs[runs["Striker"] == player_name]["runs_scored"]

print(f"Total matches found for {player_name}: {len(markram_runs)}")
get_best_distribution(markram_runs)
from scipy.stats import truncnorm

# Using the parameters from previous result
a, b, loc, scale = (-0.0753044365794551, 2.0396814396841254, 2.4210677493492394,
32.151556108391276)
x = np.linspace(0, 100, 1000)
pdf = truncnorm.pdf(x, a, b, loc=loc, scale=scale)

plt.figure(figsize=(10, 6))
plt.hist(markram_runs, bins=10, density=True, alpha=0.6, color='gray', label='Actual data')
plt.plot(x, pdf, 'r-', label='Truncnorm fit')
plt.title("Best Fit Distribution for AK Markram's Runs per Match")
plt.xlabel("Runs")
plt.ylabel("Density")
```

```
plt.legend()
plt.grid(True)
plt.show()
```

****Relationship between the performance of a player and the salary he gets****

```
R2024 =total_run_each_year[total_run_each_year['year']==2024]
!pip install fuzzywuzzy[speedup]
from fuzzywuzzy import process

# Convert to DataFrame
df_salary = ipl_salary.copy()
df_runs = R2024.copy()
# Function to match names
def match_names(name, names_list):
    match, score = process.extractOne(name, names_list)
    return match if score >= 80 else None  # Use a threshold score of 80
# 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()
# Calculation of correlation
correlation = df_merged['Rs'].corr(df_merged['runs_scored'])

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