VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A1b: Preliminary preparation and analysis of data – Descriptive Statistics

POLAMREDDY MADHUMITHA

V01107517

Date of Submission: 18-06-2024

CONTENTS

Analysing

INTRODUCTION

The Indian Premier League (IPL) is a premier professional Twenty20 cricket league in India, attracting a global audience and generating significant economic interest. This study examines the IPL using two datasets: "Cricket_data.csv," which includes match details and player performance data, and "Salary 2024.csv," which contains player salaries for the year 2024. Through an in-depth analysis of these datasets, we aim to uncover valuable insights into player performance, salary trends, and potential biases within the IPL. The findings will enhance our understanding of the league's dynamics and could impact future player recruitment and salary negotiations.

OBJECTIVES

This report aims to analyze Indian Premier League (IPL) data to uncover insights on player performance and salary. This will involve:

1. We will carefully extract and organize data from "Cricket_data.csv" and "Salary_2024.csv" to ensure accuracy and reliability. The data will be meticulously organized by round, player, and performance metrics (such as runs and wickets).
2. determining the top three wicket-takers and run-scorers for each IPL round for the previous three years.
3. Appropriate probability distributions will be fitted to the runs scored and wickets claimed by the top players. We will be able to obtain trustworthy insights into performance patterns thanks to this accurate modeling.
4. examining the relationship between a player's 2024 salary and their historical performance.

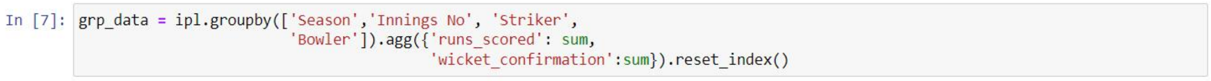
BUSINESS SIGNIFICANCE

This analysis of IPL player performance and salaries holds significant value for franchise owners and team management. By examining the distribution of runs and wickets for top performers, franchises can gain insights into player consistency and identify undervalued talent. Additionally, exploring the relationship between performance and salary can inform future player acquisition strategies, potentially leading to more efficient allocation of resources and a competitive edge. Furthermore, highlighting the salary discrepancies between top batters and bowlers provides valuable context for salary negotiations and helps ensure fair compensation across player roles. Ultimately, these findings empower data-driven decision-making, optimizing team rosters and maximizing on-field success.

RESULTS AND INTERPRETATION

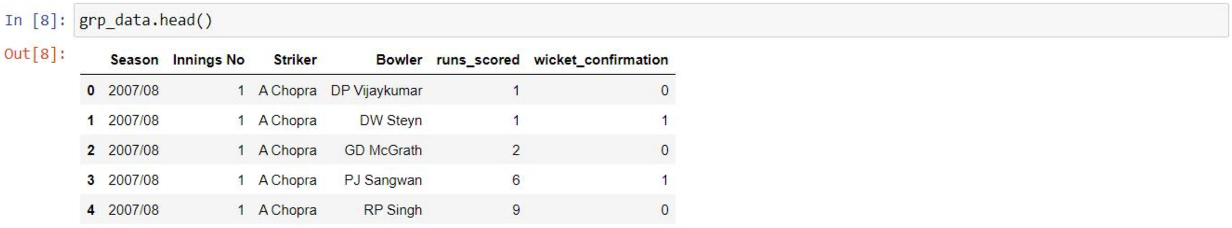
A. Arranging the data IPL round-wise and batsmen, ball, runs, and wickets per player per match

Code used:



```
In [7]: grp_data = ipl.groupby(['Season','Innings No', 'Striker',
                              'Bowler']).agg({'runs_scored': sum,
                                             'wicket_confirmation':sum}).reset_index()
```

Output:



```
In [8]: grp_data.head()
```

Out[8]:

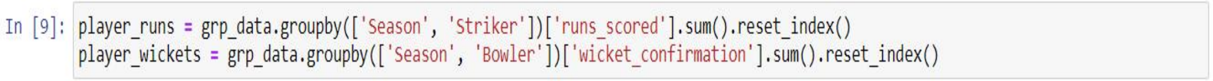
	Season	Innings No	Striker	Bowler	runs_scored	wicket_confirmation
0	2007/08	1	A Chopra	DP Vijaykumar	1	0
1	2007/08	1	A Chopra	DW Steyn	1	1
2	2007/08	1	A Chopra	GD McGrath	2	0
3	2007/08	1	A Chopra	PJ Sangwan	6	1
4	2007/08	1	A Chopra	RP Singh	9	0

Interpretation: The IPL data is reorganized by this code based on the season, innings played, batsman, and bowler. Next, it figures out how many runs each batter has scored and how many wickets each bowler has taken in each particular group.

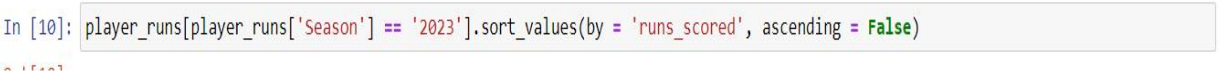
B. Fitting the most appropriate distribution for runs scored and wickets take by the top three batsmen and bowlers in the three IPL tournaments

Code and Result:

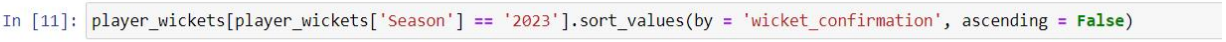
1. ***Grouping the data by Season, Striker and Bowler.***



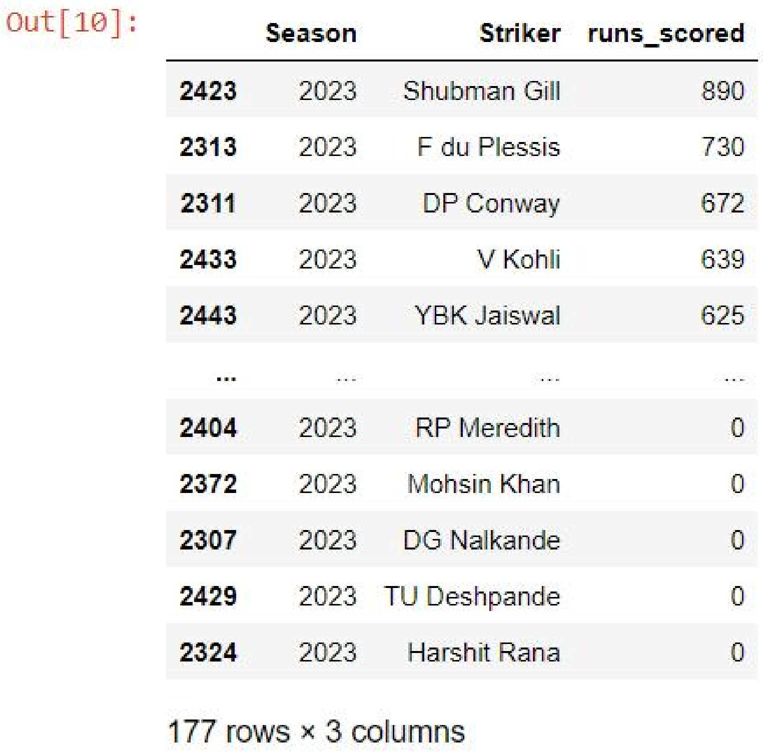
```
In [9]: player_runs = grp_data.groupby(['Season', 'Striker'])['runs_scored'].sum().reset_index()
        player_wickets = grp_data.groupby(['Season', 'Bowler'])['wicket_confirmation'].sum().reset_index()
```



```
In [10]: player_runs[player_runs['Season'] == '2023'].sort_values(by = 'runs_scored', ascending = False)
```



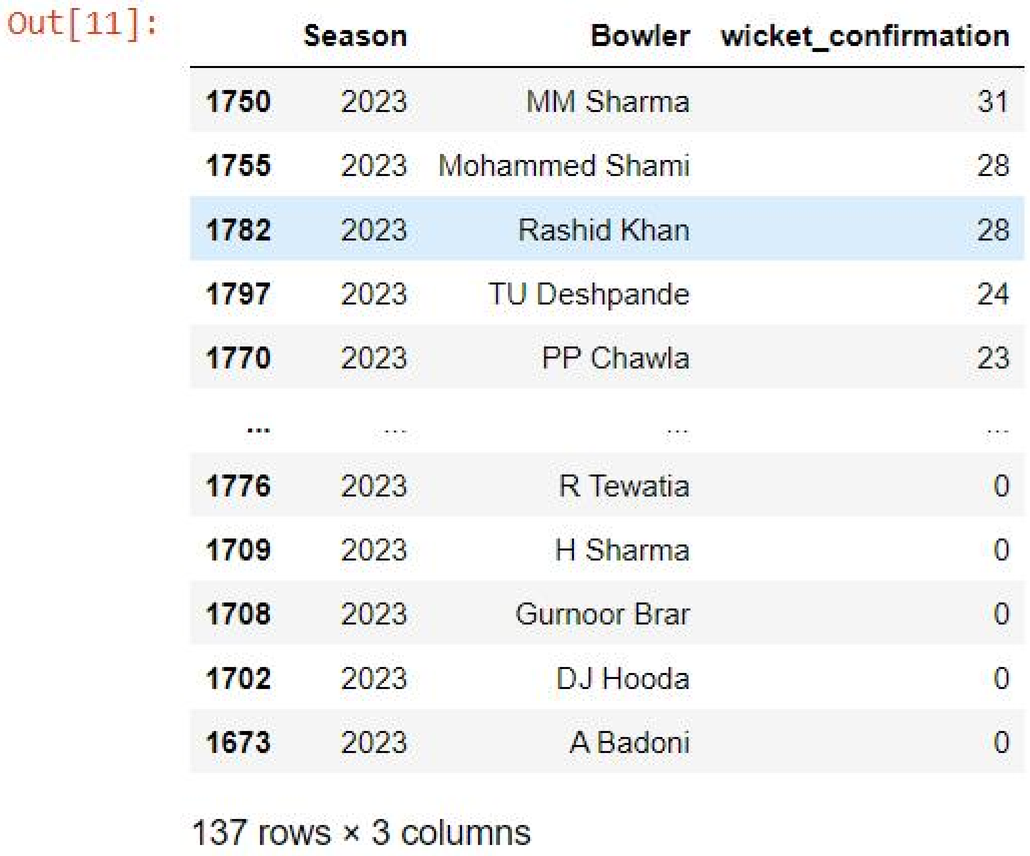
```
In [11]: player_wickets[player_wickets['Season'] == '2023'].sort_values(by = 'wicket_confirmation', ascending = False)
```



Out[10]:

	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

177 rows × 3 columns

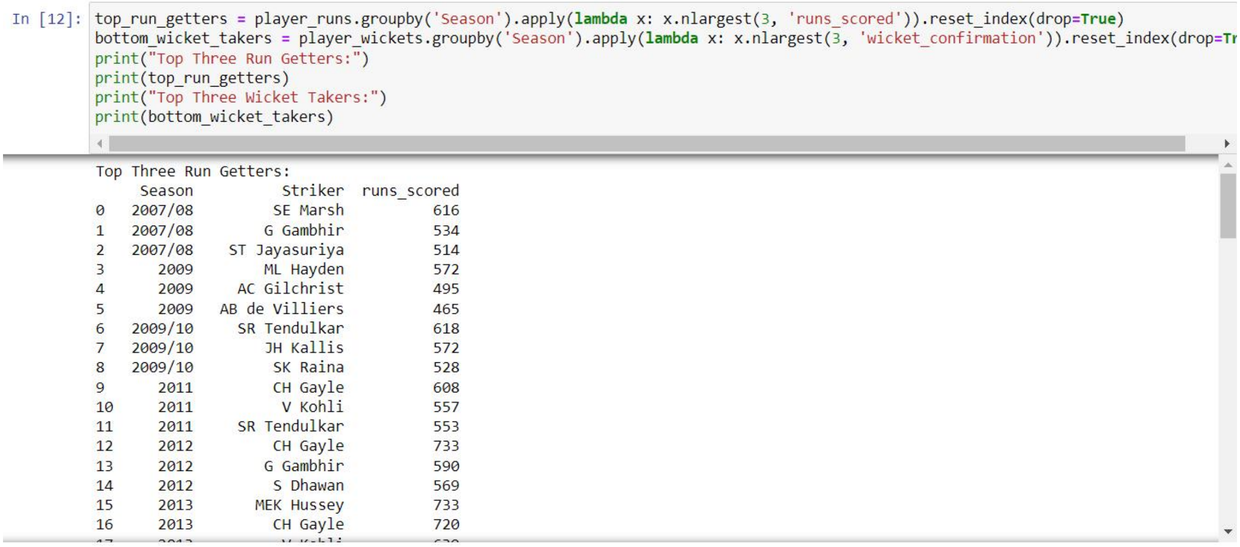


Out[11]:

	Season	Bowler	wicket_confirmation
1750	2023	MM Sharma	31
1755	2023	Mohammed Shami	28
1782	2023	Rashid Khan	28
1797	2023	TU Deshpande	24
1770	2023	PP Chawla	23
...	...	...	...
1776	2023	R Tewatia	0
1709	2023	H Sharma	0
1708	2023	Gurnoor Brar	0
1702	2023	DJ Hooda	0
1673	2023	A Badoni	0

137 rows × 3 columns

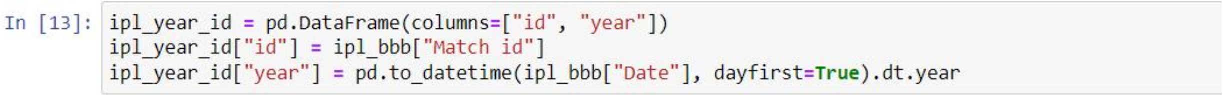
2. ***Identifying*** top three run getters and wicket taker in all seasons



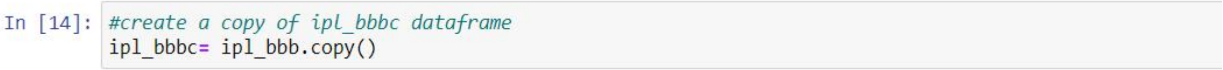
```
In [12]: top_run_getters = player_runs.groupby('Season').apply(lambda x: x.nlargest(3, 'runs_scored')).reset_index(drop=True)
         bottom_wicket_takers = player_wickets.groupby('Season').apply(lambda x: x.nlargest(3, 'wicket_confirmation')).reset_index(drop=Tr
         print("Top Three Run Getters:")
         print(top_run_getters)
         print("Top Three Wicket Takers:")
         print(bottom_wicket_takers)
```

```
Top Three Run Getters:
     Season          Striker  runs_scored
0   2007/08         SE Marsh          616
1   2007/08        G Gambhir          534
2   2007/08    ST Jayasuriya          514
3      2009        ML Hayden          572
4      2009     AC Gilchrist          495
5      2009   AB de Villiers          465
6   2009/10     SR Tendulkar          618
7   2009/10        JH Kallis          572
8   2009/10         SK Raina          528
9      2011         CH Gayle          608
10     2011          V Kohli          557
11     2011     SR Tendulkar          553
12     2012         CH Gayle          733
13     2012        G Gambhir          590
14     2012         S Dhawan          569
15     2013        MEK Hussey          733
16     2013         CH Gayle          720
```

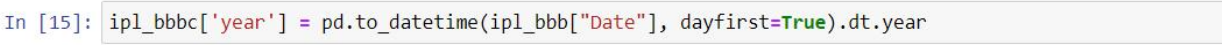
3. Creating a consolidated data frame containing Strikers, Bowlers and Seasons



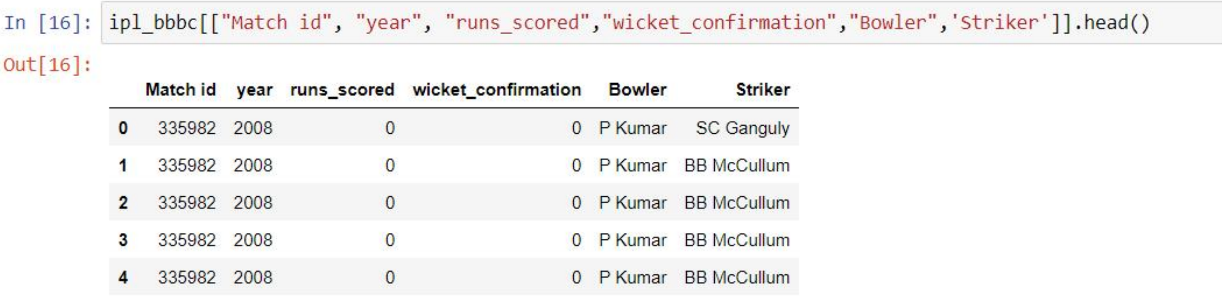
```
In [13]: ipl_year_id = pd.DataFrame(columns=["id", "year"])
         ipl_year_id["id"] = ipl_bbb["Match id"]
         ipl_year_id["year"] = pd.to_datetime(ipl_bbb["Date"], dayfirst=True).dt.year
```



```
In [14]: #create a copy of ipl_bbbc dataframe
         ipl_bbbc= ipl_bbb.copy()
```



```
In [15]: ipl_bbbc['year'] = pd.to_datetime(ipl_bbb["Date"], dayfirst=True).dt.year
```



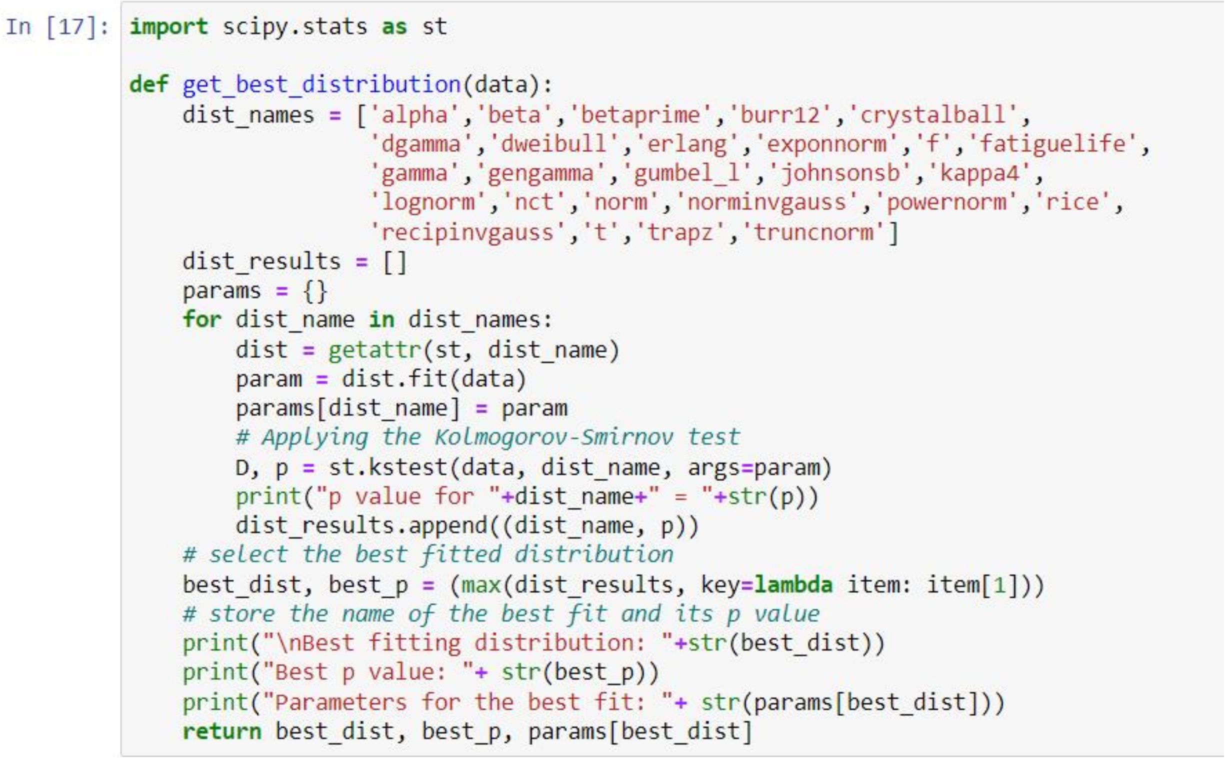
```
In [16]: ipl_bbbc[["Match id", "year", "runs_scored","wicket_confirmation","Bowler",'Striker']].head()
```

Out[16]:

	Match id	year	runs_scored	wicket_confirmation	Bowler	Striker
0	335982	2008	0	0	P Kumar	SC Ganguly
1	335982	2008	0	0	P Kumar	BB McCullum
2	335982	2008	0	0	P Kumar	BB McCullum
3	335982	2008	0	0	P Kumar	BB McCullum
4	335982	2008	0	0	P Kumar	BB McCullum

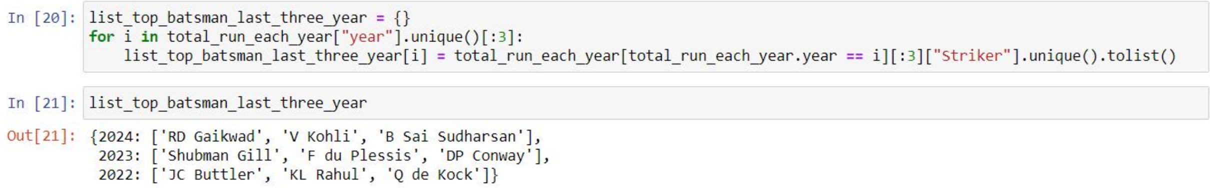
4. Finding the most appropriate distribution for runs scored and wickets take by the top three batsmen and bowlers in the three IPL tournaments

- **Goodness-of-fit test:** A strong method for accurately determining how well a theoretical distribution (such as the normal or Poisson) fits a specific dataset is the KS test. This makes it easier to conclude with confidence whether the data most likely came from that particular distribution.
- **Comparing two samples:** To determine if two independent samples' distributions differ statistically, the KS test can be used to compare them. This is helpful when comparing two players' batting performances or two teams' bowling strategy.

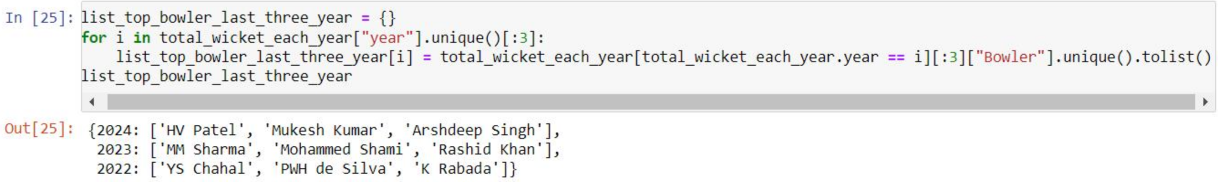


```
In [17]: import scipy.stats as st

         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]
```

5. *Listing the top three Strikers and Bowlers in last three years*

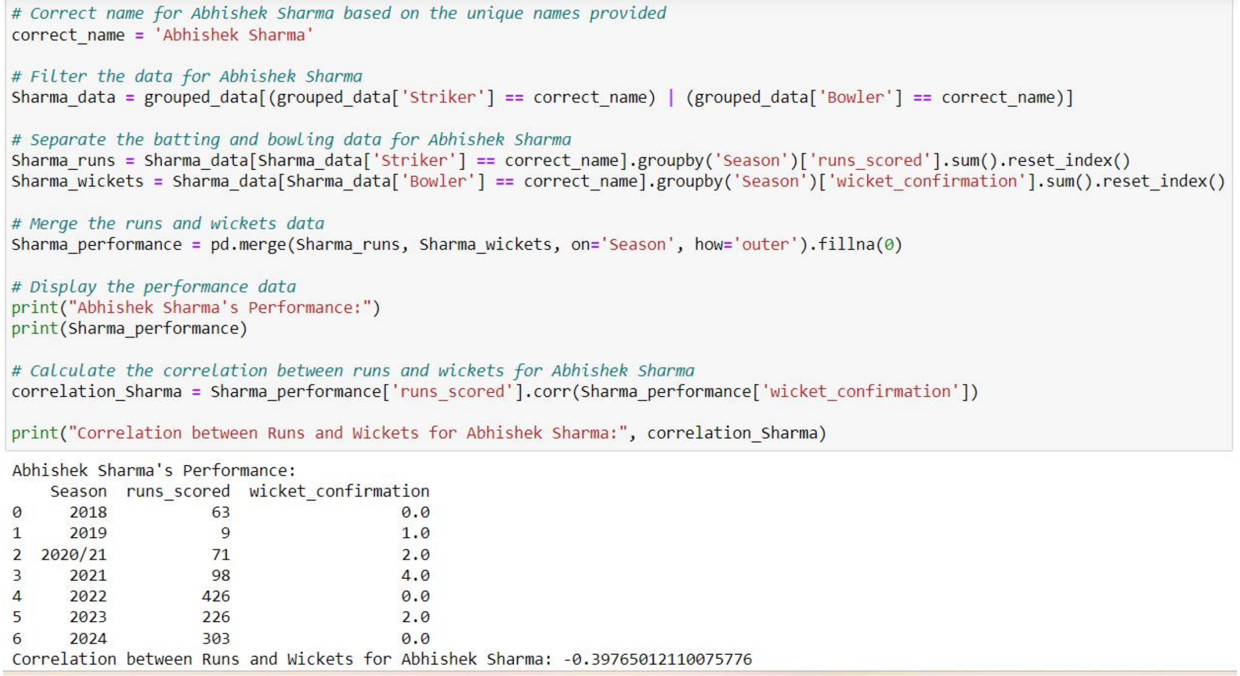


```
In [20]: list_top_batsman_last_three_year = {}
         for i in total_run_each_year["year"].unique()[:3]:
             list_top_batsman_last_three_year[i] = total_run_each_year[total_run_each_year.year == i][:3]["Striker"].unique().tolist()

In [21]: list_top_batsman_last_three_year

Out[21]: {2024: ['RD Gaikwad', 'V Kohli', 'B Sai Sudharsan'],
          2023: ['Shubman Gill', 'F du Plessis', 'DP Conway'],
          2022: ['JC Buttler', 'KL Rahul', 'Q de Kock']}
```



```
In [25]: 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

Out[25]: {2024: ['HV Patel', 'Mukesh Kumar', 'Arshdeep Singh'],
          2023: ['MM Sharma', 'Mohammed Shami', 'Rashid Khan'],
          2022: ['YS Chahal', 'PWH de Silva', 'K Rabada']}
```

6. ***Fitting the most appropriate distribution for wickets take by Abhishek Sharma. The same code can be used to find the most appropriate distribution for runs scored or wickets taken by a Striker or a Bowler respectively.***



```
# Correct name for Abhishek Sharma based on the unique names provided
correct_name = 'Abhishek Sharma'

# Filter the data for Abhishek Sharma
Sharma_data = grouped_data[(grouped_data['Striker'] == correct_name) | (grouped_data['Bowler'] == correct_name)]

# Separate the batting and bowling data for Abhishek Sharma
Sharma_runs = Sharma_data[Sharma_data['Striker'] == correct_name].groupby('Season')['runs_scored'].sum().reset_index()
Sharma_wickets = Sharma_data[Sharma_data['Bowler'] == correct_name].groupby('Season')['wicket_confirmation'].sum().reset_index()

# Merge the runs and wickets data
Sharma_performance = pd.merge(Sharma_runs, Sharma_wickets, on='Season', how='outer').fillna(0)

# Display the performance data
print("Abhishek Sharma's Performance:")
print(Sharma_performance)

# Calculate the correlation between runs and wickets for Abhishek Sharma
correlation_Sharma = Sharma_performance['runs_scored'].corr(Sharma_performance['wicket_confirmation'])

print("Correlation between Runs and Wickets for Abhishek Sharma:", correlation_Sharma)
```

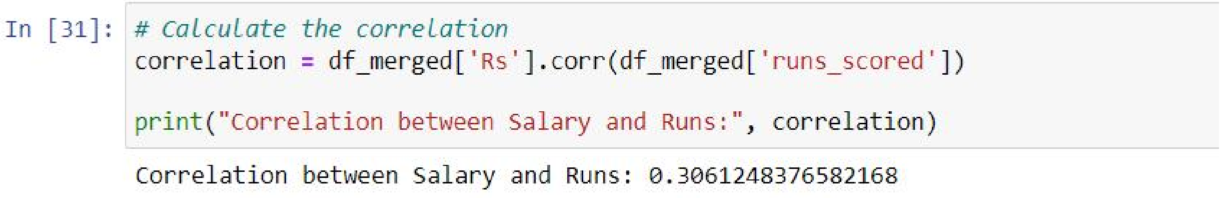
```
Abhishek Sharma's Performance:
    Season  runs_scored  wicket_confirmation
0     2018           63                  0.0
1     2019            9                  1.0
2  2020/21           71                  2.0
3     2021           98                  4.0
4     2022          426                  0.0
5     2023          226                  2.0
6     2024          303                  0.0
Correlation between Runs and Wickets for Abhishek Sharma: -0.39765012110075776
```

<ins>Interpretation:</ins> Thus the fitting distribution for wickets taken by Abhishek Sharma is the T-Test.

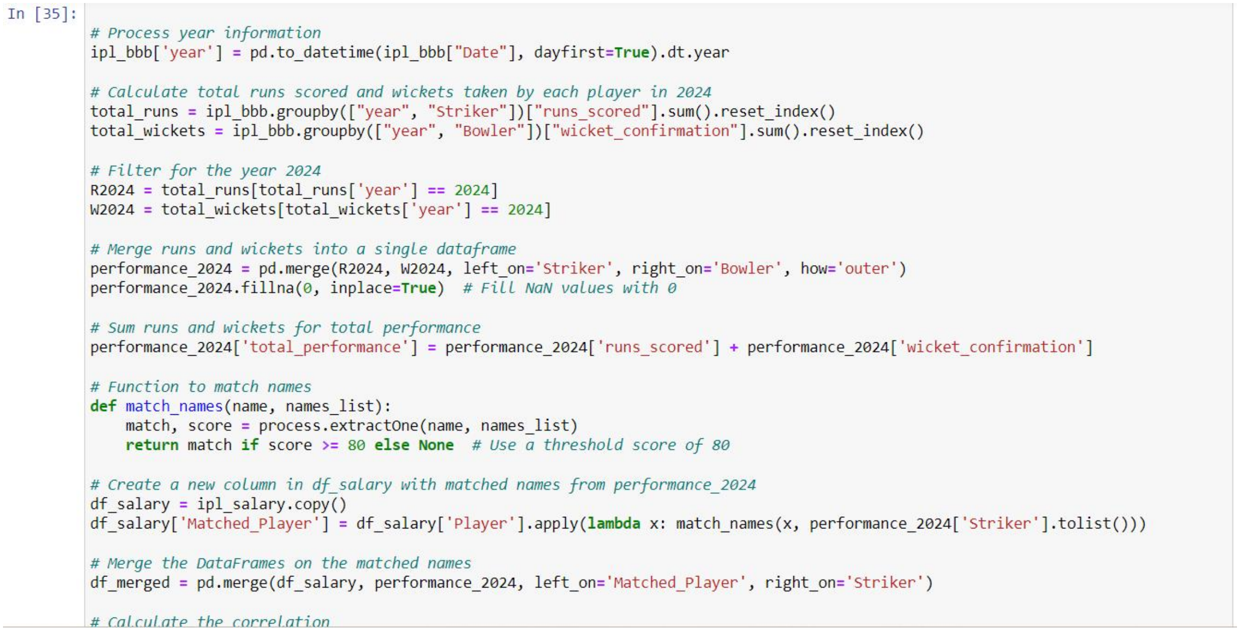
C. Finding the relationship between a player’s performance and the salary he gets as per the data.

The names of players are in different format in database. Thus, it is required to regularize the names to proceed with further analysis.

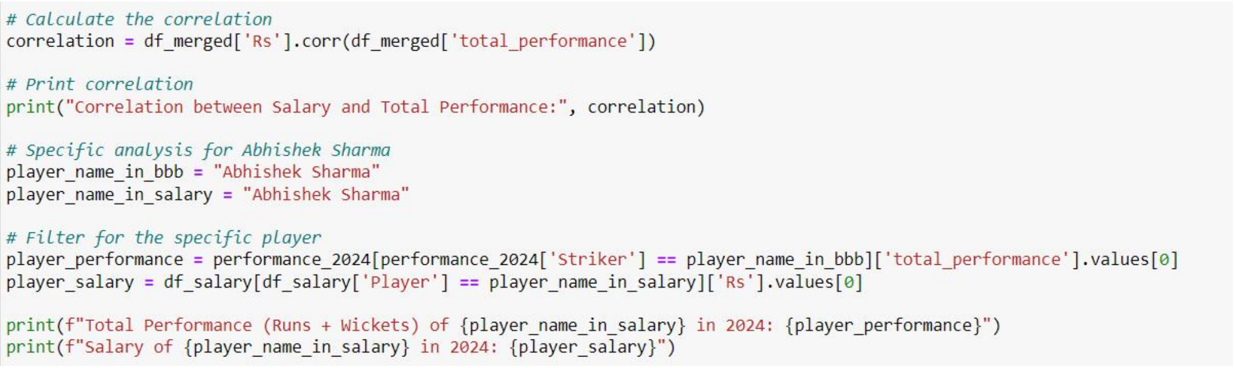
Code and Results:



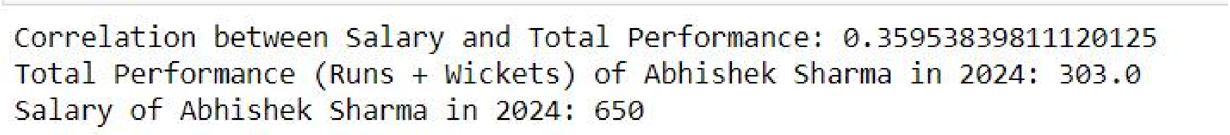
```
In [31]: # Calculate the correlation
         correlation = df_merged['Rs'].corr(df_merged['runs_scored'])

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

         Correlation between Salary and Runs: 0.3061248376582168
```



```
In [35]:
         # Process year information
         ipl_bbb['year'] = pd.to_datetime(ipl_bbb["Date"], dayfirst=True).dt.year

         # Calculate total runs scored and wickets taken by each player in 2024
         total_runs = ipl_bbb.groupby(["year", "Striker"])["runs_scored"].sum().reset_index()
         total_wickets = ipl_bbb.groupby(["year", "Bowler"])["wicket_confirmation"].sum().reset_index()

         # Filter for the year 2024
         R2024 = total_runs[total_runs['year'] == 2024]
         W2024 = total_wickets[total_wickets['year'] == 2024]

         # Merge runs and wickets into a single dataframe
         performance_2024 = pd.merge(R2024, W2024, left_on='Striker', right_on='Bowler', how='outer')
         performance_2024.fillna(0, inplace=True)  # Fill NaN values with 0

         # Sum runs and wickets for total performance
         performance_2024['total_performance'] = performance_2024['runs_scored'] + performance_2024['wicket_confirmation']

         # 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 performance_2024
         df_salary = ipl_salary.copy()
         df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x, performance_2024['Striker'].tolist()))

         # Merge the DataFrames on the matched names
         df_merged = pd.merge(df_salary, performance_2024, left_on='Matched_Player', right_on='Striker')

         # Calculate the correlation
```



```
# Calculate the correlation
correlation = df_merged['Rs'].corr(df_merged['total_performance'])

# Print correlation
print("Correlation between Salary and Total Performance:", correlation)

# Specific analysis for Abhishek Sharma
player_name_in_bbb = "Abhishek Sharma"
player_name_in_salary = "Abhishek Sharma"

# Filter for the specific player
player_performance = performance_2024[performance_2024['Striker'] == player_name_in_bbb]['total_performance'].values[0]
player_salary = df_salary[df_salary['Player'] == player_name_in_salary]['Rs'].values[0]

print(f"Total Performance (Runs + Wickets) of {player_name_in_salary} in 2024: {player_performance}")
print(f"Salary of {player_name_in_salary} in 2024: {player_salary}")
```



```
Correlation between Salary and Total Performance: 0.35953839811120125
Total Performance (Runs + Wickets) of Abhishek Sharma in 2024: 303.0
Salary of Abhishek Sharma in 2024: 650
```

Interpretation: A statistical indicator of the direction and intensity of a linear relationship between two variables is the correlation coefficient. It falls between -1 and 1. A positive correlation(+1) means that there is a tendency for both variables to rise as one increases. When one variable tends to decrease as the other grows, there is a negative correlation (-1). There is no linear relationship between the variables when the correlation value is 0. There is a relationship between a striker's pay and performance. As a result, the correlation value of 0.30612 suggests that a weak positive association exists. A player's performance is determined by a variety of criteria in addition to their salary, including experience, reputation, prior success, and so on.