Documentation

IPL Data Analysis Project

Project Overview

Project Title: IPL Data Analysis

Course : Data Analytics

Institute : Anudip Foundation

Duration : [11 Nov – Present]

Date : [21/02/2025]

Overview :

This project focuses on performing data analysis on the Indian Premier League (IPL) dataset to uncover insights and trends from the historical data of IPL matches. The analysis includes player statistics, team performance, match outcomes, and other key metrics. The goal is to use data analytics techniques to derive actionable insights and provide a deeper understanding of IPL trends.

Objective :

The main objectives of this project are:

1) Understand player performance: Analyzing individual player statistics (runs, wickets, economy rate, etc.) to evaluate performance trends.

2) Analyze team performance: Identifying winning patterns, the best-performing teams, and factors contributing to victories.

3) Match outcome prediction: Exploring factors that influence the outcomes of IPL matches.

4) Trend analysis: Investigating trends in IPL over the years, including top players, team strategies, and performance improvements.

Scenario :

IPL Data Analysis And Visualizations using Python

Tools and Technologies Used

- **Programming Languages**: Python

- **Libraries**:

  - **Pandas**: For data manipulation and cleaning.
  - **Matplotlib/Seaborn**: For data visualization.
  - **NumPy**: For numerical operations.
  - **Jupyter Notebook**: For documentation and interactive analysis.

Methodology :

1. Data Collection

Gather IPL data from publicly available sources like:

- IPL Kaggle Dataset

2. Data Cleaning

- **Handling missing values**: Ensuring that all missing or NaN values are either filled or removed to maintain data integrity.
- **Data type conversions**: Ensuring all columns are of the correct data type (e.g., dates as datetime, categorical variables as category, etc.).
- **Removing duplicates**: Identifying and removing any duplicate entries in the dataset.

3. Exploratory Data Analysis (EDA)

- **Data Summary**: Using `describe()` and `info()` functions to generate a statistical summary of the dataset.
- **Data Visualization**:
  - Distribution of runs, wickets, and other key statistics for players.
  - Team-wise performance across seasons.
  - Visual representation of match outcomes (winners and losers).

4. Performance Analysis

- **Player Performance**: Analyzing top-performing players in terms of runs, wickets, strike rate, economy rate, etc.
- **Team Analysis**: Evaluating which teams have the best win rates and identifying the factors contributing to their success.
- **Winning Patterns**: Analyzing the frequency of team wins by toss winner, home/away games, and more.

5. Visualization and Reporting

Using libraries such as Matplotlib, Seaborn, and Plotly, generate visualizations to present the data and results in a more understandable format:

- **Bar Chart**: To show top teams with the highest win rates or player rankings.

- **Line Plot**: To showcase the change in performance over seasons for a team or player.

Key Insights :

- **Top Players:** Based on the analysis, the top-performing players in terms of batting, bowling, and all-round performance were identified.
- **Winning Teams:** Teams with the highest winning percentage were highlighted, along with their key strategies for success.
- **Match Outcomes:** Teams winning the toss have a slightly higher chance of winning, but home teams tend to win more matches.
- **Trends:** There was a clear trend of increasing player performance over the years as the league matured.

Results and Conclusion :

This project successfully analyzed IPL data to uncover performance trends, key players, and winning patterns. The key takeaways include:

- **Top Performers**: Players like [Rohit Sharma, Chris Gyale] have dominated the IPL with consistent performance across seasons.

- **Winning Teams**: [Mumbai Indians , Chennai Super Kings] consistently outperformed others due to factors like [relevant factors, e.g., strong batting line-up, effective bowling attack].

Future Work :

- Extend analysis to predict individual player performance in upcoming seasons.
- Develop a more advanced predictive model using machine learning to account for multiple factors affecting match outcomes.
- Explore more granular data, such as player injuries and their impact on match outcomes.

Code Repository :

https://github.com/Rushikesh-Jadhav-45/Project_Data_Anaylisis.git

Code Snippets :

1. Data Loading and Preprocessing :

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[2]: dataset = pd.read_csv("E:\Data_Analytics\\IPL_Data.csv")
```

2. Data Cleaning :

1) This will fill the null values

```
[7]: dataset.dropna(inplace=True)   # Removes rows with null values
     dataset.dropna(axis=1,inplace=True) # Removes coulumns with null values
```

```
[8]: dataset.isnull().sum()
```

3. **Data Aggregation and Analysis** :

1) Keep date in correct format.

[10]:
```
# This will corrects the data format in existing table
datatset = pd.to_datetime(dataset['date'])
```

2) Calculate the total number of wins be each teams :

[6]:
```
# It will revert the total number of winns by each team
dataset['winner'].value_counts().head(10)
```

[6]:
```
winner
Mumbai Indians                  80
Chennai Super Kings             79
Royal Challengers Bangalore     70
Kolkata Knight Riders           68
Rajasthan Royals                63
Kings XI Punjab                 63
Delhi Daredevils                56
Sunrisers Hyderabad             34
Deccan Chargers                 29
Pune Warriors                   12
Name: count, dtype: int64
```

3) Calculate the Most Successful teams based on the win percentage

[11]:
```
# Most successful temas ( based on win percentage )
team_wins = dataset['winner'].value_counts()
total_matches = dataset['team1'].value_counts() + dataset['team2'].value_counts()
win_percentage = (team_wins / total_matches) * 100
print(win_percentage.sort_values(ascending=False))
```

```
Series([], Name: count, dtype: float64)
```

4) Calculate the player of the match per match

[4]:
```
# Reverts Player of match of 10 players
dataset['player_of_match'].value_counts().head(10)
```

[4]:
```
player_of_match
CH Gayle          17
YK Pathan         16
AB de Villiers    15
DA Warner         14
RG Sharma         13
SK Raina          13
MEK Hussey        12
MS Dhoni          12
G Gambhir         12
AM Rahane         12
Name: count, dtype: int64
```

5) Calculate the Player of the match from all season.

[11]:
```
# It will revert Unique teams and numbers of players of match
print(dataset['team1'].unique())
print(dataset['player_of_match'].value_counts())
```

```
['Kolkata Knight Riders' 'Chennai Super Kings' 'Rajasthan Royals'
 'Mumbai Indians' 'Deccan Chargers' 'Kings XI Punjab'
 'Royal Challengers Bangalore' 'Delhi Daredevils' 'Kochi Tuskers Kerala'
 'Pune Warriors' 'Sunrisers Hyderabad' 'Rising Pune Supergiants'
 'Gujarat Lions']
player_of_match
CH Gayle          17
YK Pathan         16
AB de Villiers    15
DA Warner         14
RG Sharma         13
                  ..
CR Brathwaite      1
RR Pant            1
A Zampa            1
HM Amla            1
BCJ Cutting        1
Name: count, Length: 187, dtype: int64
```

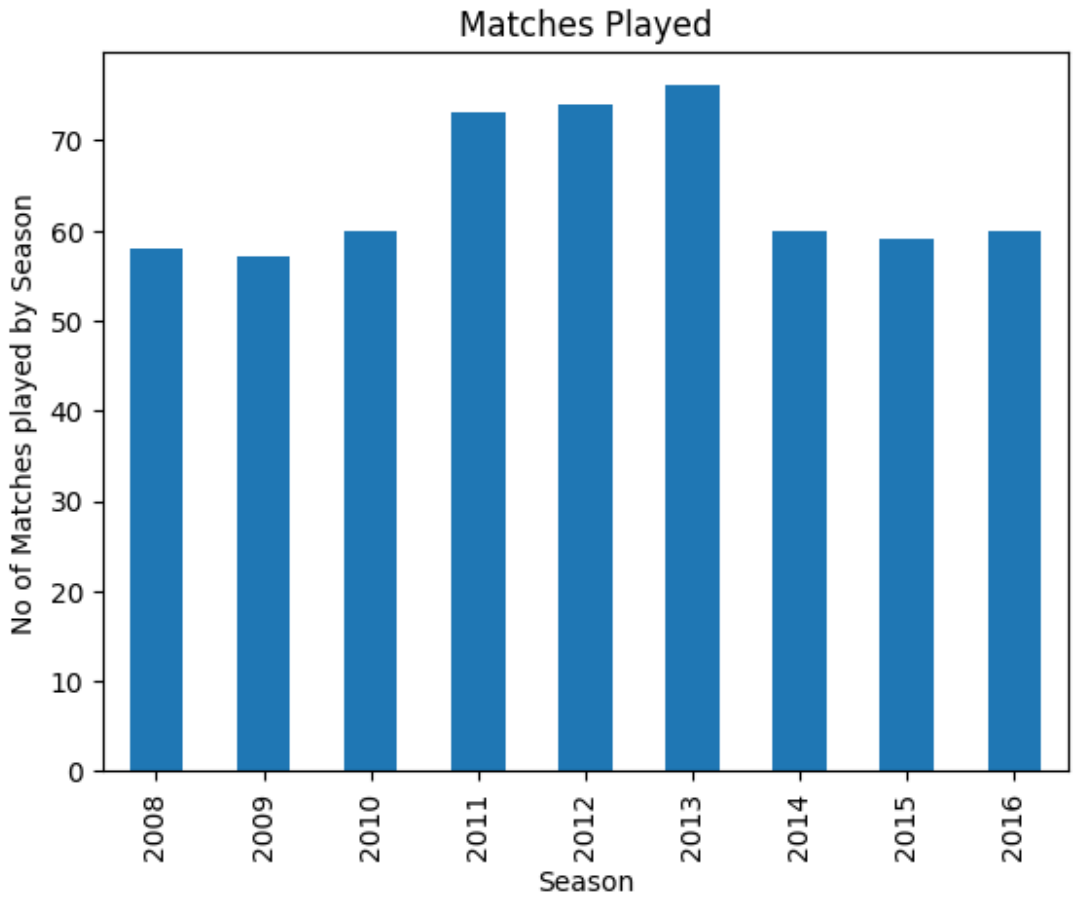
4. Data Visualization :

1) How many matches played by each season ?

Code :

```
[23]: #It will shows the no of matches played by each season
dataset.groupby('season')['id'].count().plot(kind='bar',title='Matches Played')
# Showing status by graph
plt.xlabel('Season')
plt.ylabel('No of Matches played by Season')
plt.show()
```

Graph :

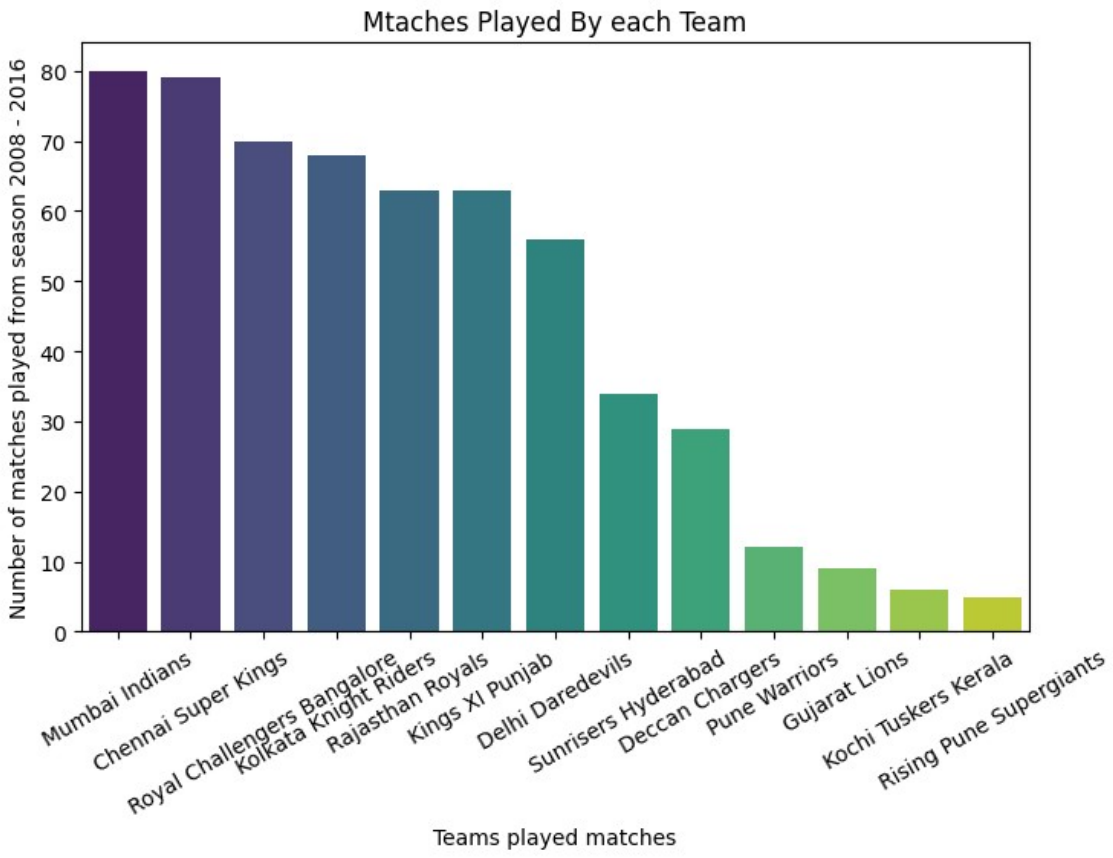


2)Calculate the total matches played by a team from all season 2008 to 2016 ?

Code :

```
[6]: # This graph will calculate that number of matches played by a teams in all seasons from 2008 to 2016
win_counts = dataset['winner'].value_counts().reset_index()
win_counts.columns = ["Team", "Titles"]
# Plotting the result
plt.figure(figsize=(8,5))
sns.barplot(data = win_counts, x="Team", y="Titles",hue="Team",palette="viridis",legend=False)
plt.xlabel("Teams played matches")
plt.ylabel("Number of matches played from season 2008 - 2016")
plt.title("Mtaches Played By each Team")
plt.xticks(rotation=30)
plt.show()
```

Graph :



3) Calculate the matches played by Mumbai Indians

Code :

[25]:
```
# This line will filters macthes won by Mumbai Indians
mi_wins = dataset[dataset["winner"] == "Mumbai Indians"]
#Counts wins per season
mi_wins_per_Season = mi_wins.groupby("season")["winner"].count().reset_index()
mi_wins_per_Season.columns = ["season","Wins"]
#Plotting the results
plt.figure(figsize=(10,6))
sns.barplot(data=mi_wins_per_Season, x="season",y="Wins", hue="season",palette="viridis",legend=False)
plt.xlabel("season")
plt.ylabel("Matches Played from Season 2008 - 2016")
plt.title("Mumbai Indians Matches Played By each season")
# plt.xticks(rotation=30)
plt.show()
```

Graph :