Pakistan Super League Analysis

1 Roll Number Name Section

2 Roll Number Name Section

3 Roll Number Name Section

4 Roll Number Name Section

1. Problem Statement

Issue or problem that you will address XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXX

2. Objective

What will be the object of this work XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

3. Data Description

Link of the data (from where data is taken). The details description of the data. XXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXX

4. Results

Consist of detailed screenshots of all results presented on web interference with suitable

heading and description. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXX

5. Codes

Paste code with 1 line space and no extra space with proper comments

## Histogram:

### Frequency of interval of wickets

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv("PSL.csv")

# Step 1: Extract "Wickets" and "season" columns

wickets\_season = df[['Wickets', 'season']]

# Step 2: Group the data by season

grouped\_data = wickets\_season.groupby('season')

# Step 3: Define class intervals for the histogram

class\_intervals = [(0, 2), (3, 4), (5, 7), (8, 10)] # Modify as needed

# Step 4: Calculate the frequency of matches with wickets in each interval for each season

frequency\_data = {}

for season, group in grouped\_data:

# Initialize frequency counts for each class interval

freq\_counts = [0] \* len(class\_intervals)

# Count the number of matches falling within each class interval

for idx, interval in enumerate(class\_intervals):

lower\_bound, upper\_bound = interval

freq\_counts[idx] = group['Wickets'].between(lower\_bound, upper\_bound).sum()

frequency\_data[season] = freq\_counts

# Step 5: Plot the histogram

plt.figure(figsize=(10, 6))

# Prepare x-axis labels

x\_labels = [f"{interval[0]}-{interval[1]}" for interval in class\_intervals]

x\_positions = range(len(x\_labels))

# Plot each season's histogram as bars

for idx, (season, freq\_counts) in enumerate(frequency\_data.items()):

plt.bar([pos + idx \* 0.1 for pos in x\_positions], freq\_counts, width=0.1, align='center', label=season)

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.xlabel('Wickets per Match')

plt.ylabel('Frequency')

plt.title('Wickets per Season Histogram')

plt.xticks([pos + (len(frequency\_data) - 1) \* 0.05 for pos in x\_positions], x\_labels)

plt.legend(title='Season')

plt.grid(True)

plt.show()

## Frequency Polygon:

### Frequency of Extras in whole PSL:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv("PSL.csv")

# Extract columns related to extras

extras\_columns = ['Byes', 'Legbyes', 'Noballs', 'Penalty', 'Wides']

extras\_data = df[extras\_columns]

# Calculate frequency of each type of extra

extras\_frequency = extras\_data.sum()

# Plotting the frequency polygon

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.plot(extras\_frequency.index, extras\_frequency.values, marker='o', linestyle='-')

plt.title('Frequency Polygon of Extras in PSL')

plt.xlabel('Type of Extra')

plt.ylabel('Frequency')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

### Frequency of review decisions:

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('PSL.csv')

# Filter the data for 'struckdown' and 'upheld' review decisions

review\_decision\_data = data[data['Review Decision'].isin(['struck down', 'upheld'])]['Review Decision']

# Calculate frequency of 'struckdown' and 'upheld' review decisions

review\_decision\_frequency = review\_decision\_data.value\_counts()

# Plotting the frequency polygon

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.plot(review\_decision\_frequency.index, review\_decision\_frequency.values, marker='o', linestyle='-')

plt.title('Frequency Polygon of Review Decisions (Struckdown vs. Upheld) in PSL')

plt.xlabel('Review Decision Type')

plt.ylabel('Frequency')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

### Frequency of dismissal type:

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('PSL.csv')

# Exclude rows where Wicket Type is 0

dismissal\_type\_data = data[data['Wicket Type'] != '0']['Wicket Type']

# Calculate frequency of each dismissal type

dismissal\_type\_frequency = dismissal\_type\_data.value\_counts()

# Plotting the frequency polygon

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.plot(dismissal\_type\_frequency.index, dismissal\_type\_frequency.values, marker='o', linestyle='-')

plt.title('Frequency Polygon of Dismissal Types in PSL')

plt.xlabel('Dismissal Type')

plt.ylabel('Frequency')

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

### Frequency of runs made in each over:

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('PSL.csv')

# Group the data by over number and calculate the frequency of runs made in each over

runs\_per\_over = data.groupby('Over Number')['Total Runs'].value\_counts().unstack().fillna(0)

# Calculate the total frequency of runs made in each over

total\_runs\_per\_over = runs\_per\_over.sum(axis=1)

# Plotting the frequency polygon

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.plot(total\_runs\_per\_over.index, total\_runs\_per\_over.values, marker='o', linestyle='-')

plt.title('Frequency Polygon of Runs Made in Every Over in PSL')

plt.xlabel('Over Number')

plt.ylabel('Frequency of Runs Made')

plt.grid(True)

plt.xticks(range(1, 21))

plt.tight\_layout()

plt.show()

## Bar Charts:

### Top 5 batters per season:

#1. Top 5 Batters per season.

import matplotlib.pyplot as plt

import pandas as pd

def plot\_top\_5\_batters(season):

# Read the data from CSV

data = pd.read\_csv('PSL.csv')

season\_data = data[data['season'] == season]

batter\_totals = season\_data.groupby('Batter')['Batter runs'].sum()

top\_5\_batters = batter\_totals.sort\_values(ascending=False).head(5)

X = top\_5\_batters.index.tolist()

Y = top\_5\_batters.values.tolist()

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.bar(X, Y, color='g')

plt.title("Top 5 Batters in Season {}".format(season))

plt.xlabel("Batter")

plt.ylabel("Total Runs")

plt.xticks(rotation=12)

plt.show()

# Take input for the season

season = "2018/19" # you can change seasons here

plot\_top\_5\_batters(season)

### Top 5 bowlers per season:

# 2. Top 5 Bowlers per season.

import matplotlib.pyplot as plt

import pandas as pd

def plot\_top\_bowlers\_by\_season(season):

data = pd.read\_csv('PSL.csv')

season\_data = data[data['season'] == season]

bowler\_totals = season\_data.groupby('Bowler')['Wickets'].sum()

top\_5\_bowlers = bowler\_totals.sort\_values(ascending=False).head(5)

X = top\_5\_bowlers.index.tolist()

Y = top\_5\_bowlers.values.tolist()

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.bar(X, Y, color='b')

plt.title(f"Top 5 Bowlers in PSL Season {season}")

plt.xlabel("Bowler")

plt.ylabel("Total Wickets")

plt.xticks(rotation=10)

plt.show()

# Take input for the season

season\_input = "2016/17"# you can tak input of every season

# Plot the top five bowlers for the specified season

plot\_top\_bowlers\_by\_season(season\_input)

### Boundaries per season:

# 3. Boundaries per season.

import matplotlib.pyplot as plt

import pandas as pd

# Read the data from CSV file

data = pd.read\_csv('PSL.csv')

# Filter data to include only rows where Total Runs is 4 or 6 (boundaries)

boundary\_data = data[data['Total Runs'].isin([4, 6])]

# Group data by season and count the number of boundaries for each season

season\_boundary\_counts = boundary\_data.groupby('season').size().reset\_index(name='Boundary Count')

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

# Extract X and Y values

X = season\_boundary\_counts['season']

Y = season\_boundary\_counts['Boundary Count']

# Plot the data using bar() method

plt.bar(X, Y, color='g')

plt.title("Total Boundaries (Fours and Sixes) Per Season")

plt.xlabel("Season")

plt.ylabel("Total Boundaries")

plt.xticks(rotation=15)

# Show the plot

plt.show()

### Boundaries of death over and powerplay (component bar-chart):

#4. Boundaries of death over and powerplay (component bar-chart)

import matplotlib.pyplot as plt

import pandas as pd

# Read the data from CSV file

data = pd.read\_csv('PSL.csv')

# Filter data to include only rows where Total Runs represent boundaries (4 or 6)

boundary\_data = data[data['Total Runs'].isin([4, 6])]

# Filter data to include only boundaries in overs 1-6 and 16-20

boundary\_data = boundary\_data[(boundary\_data['Over Number'].isin(range(1, 7))) | (boundary\_data['Over Number'].isin(range(16, 21)))]

# Group data by season and over number range, and count the number of boundaries for each group

season\_over\_boundary\_counts = boundary\_data.groupby(['season', pd.cut(boundary\_data['Over Number'], bins=[0, 6, 20], labels=['Powerplay', 'Death Over'])]).size().unstack(fill\_value=0)

# Plot the component bar chart

season\_over\_boundary\_counts.plot(kind='bar', stacked=True)

plt.title("Total Boundaries (Fours and Sixes) in Powerplay and Death Overs Per Season")

plt.xlabel("Season")

plt.ylabel("Total Boundaries")

plt.xticks(rotation=45)

# Show the plot

plt.legend(title='Over Range')

plt.tight\_layout()

plt.show()

## Line Graph:

### Average boundaries per match per season:

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv("PSL.csv", low\_memory=False)

# Filter the relevant columns

filtered\_data = data[["season", "Batter runs", "event match\_number"]]

# Filter for fours (4) and sixes (6) only

filtered\_data = filtered\_data[filtered\_data["Batter runs"].isin([4, 6])]

# Group by season and calculate total fours and sixes

season\_data = filtered\_data.groupby("season").size().reset\_index(name='Total Boundaries')

# Calculate the total number of matches per season

matches\_per\_season = filtered\_data.groupby("season")["event match\_number"].nunique().reset\_index(name='Total Matches')

# Merge total boundaries and total matches dataframes

season\_data = pd.merge(season\_data, matches\_per\_season, on="season")

# Calculate average boundaries per match per season

season\_data["Average Boundaries per Match"] = season\_data["Total Boundaries"] / season\_data["Total Matches"]

# Plot average boundaries per match per season

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.plot(season\_data["season"], season\_data["Average Boundaries per Match"], marker='o', color='green')

plt.title("Average Boundaries per Match per Season")

plt.xlabel("Season")

plt.ylabel("Average Boundaries per Match")

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

### Dot balls per season:

import pandas as pd

import matplotlib.pyplot as plt

# Load data from CSV file

data = pd.read\_csv("PSL.csv", low\_memory=False)

# Filter the relevant columns

filtered\_data = data[["season", "Over Number", "Total Runs"]]

# Filter for dot balls (Total Runs == 0)

dot\_balls\_data = filtered\_data[filtered\_data["Total Runs"] == 0]

# Group by season and count the number of balls and dot balls

season\_dot\_balls = dot\_balls\_data.groupby("season").agg({'Over Number': 'count'}).reset\_index()

season\_dot\_balls.rename(columns={'Over Number': 'Dot Balls'}, inplace=True)

# Group by season and count the total number of balls bowled

season\_total\_balls = filtered\_data.groupby("season").agg({'Over Number': 'size'}).reset\_index()

season\_total\_balls.rename(columns={'Over Number': 'Total Balls'}, inplace=True)

season\_dot\_balls\_percentage = pd.merge(season\_dot\_balls, season\_total\_balls, on='season')

season\_dot\_balls\_percentage['Dot Balls Percentage'] = (season\_dot\_balls\_percentage['Dot Balls'] / season\_dot\_balls\_percentage['Total Balls']) \* 100

# Plot dot balls percentage per season

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.plot(season\_dot\_balls\_percentage["season"], season\_dot\_balls\_percentage["Dot Balls Percentage"], marker='o', color='blue')

plt.title("Dot Balls Percentage per Season")

plt.xlabel("Season")

plt.ylabel("Dot Balls Percentage")

plt.grid(True)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

## Pie chart:

### Season won by each team:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CSV file into a DataFrame

df = pd.read\_csv("PSL.csv")

# Convert the 'dates' column to datetime format

df['dates'] = pd.to\_datetime(df['dates'], errors='coerce')

# Sort the DataFrame by the date of the matches

df = df.sort\_values(by='dates')

# Group the DataFrame by season and find the last match of each season

last\_matches = df.groupby('season').tail(1)

# Get the winner of the last match of each season

season\_winners = last\_matches['Winner']

# Count the number of seasons won by each team

seasons\_won\_count = season\_winners.value\_counts()

# Define custom colors

colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99']

# Explode the slices

# Plotting a pie chart with customizations

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.pie(seasons\_won\_count, labels=seasons\_won\_count.index, autopct='%1.1f%%', startangle=140, colors=colors, shadow=True)

plt.title('PSL Seasons Won by Teams', fontsize=25, fontweight='bold',pad=20)

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.tight\_layout() # Adjust layout to prevent labels from being cut off

plt.show()

### Teams that won most tosses:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CSV file into a DataFrame

df = pd.read\_csv("PSL.csv")

# Count the number of tosses won by each team

toss\_winner\_counts = df['Toss Winner'].value\_counts()

# Get the teams that won the most tosses

most\_toss\_winner = toss\_winner\_counts.idxmax()

# Plotting a pie chart for teams with most toss wins

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

plt.pie(toss\_winner\_counts, labels=toss\_winner\_counts.index, autopct='%1.1f%%', startangle=140)

plt.title('Teams with Most Toss Wins', fontsize=16, fontweight='bold', pad=20)

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.tight\_layout() # Adjust layout to prevent labels from being cut off

plt.show()

print(f"The team with the most toss wins: {most\_toss\_winner} ({toss\_winner\_counts[most\_toss\_winner]} wins)")

### Venue with most matches:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CSV file into a DataFrame

df = pd.read\_csv("PSL.csv")

# Count the number of matches held at each venue

venue\_counts = df['venue'].value\_counts()

# Get the venue with the maximum number of matches

max\_venue = venue\_counts.idxmax()

# Define colors for the pie chart

colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf']

# Plotting a pie chart without labels

plt.figure(figsize=(12.8, 7.2)) # 1280x720 pixels

# Center the plot window on the screen

manager = plt.get\_current\_fig\_manager()

manager.window.wm\_geometry("+{}+{}".format(128, 22))

patches, texts, autotexts = plt.pie(venue\_counts, labels=None, autopct='%1.1f%%', startangle=140, colors=colors, textprops={'fontsize': 12})

plt.title('Distribution of Matches Held at Different Venues', fontsize=18, fontweight='bold', pad=20)

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

# Create legend with custom labels and colors

plt.legend(patches, venue\_counts.index, loc='center left', bbox\_to\_anchor=(1, 0.5), fontsize=12)

plt.tight\_layout() # Adjust layout to prevent labels from being cut off

plt.show()

print(f"The venue with the most matches: {max\_venue} ({venue\_counts[max\_venue]} matches)")

6. Conclusion

Conclude your results in one paragraph XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXX