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

# 2. Objective

# 3. Data Description

# 4. Results

# 5. Codes

## 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)")

### Teams with most wins in current season:

import pandas as pd

import matplotlib.pyplot as plt

# Load the CSV file into a DataFrame

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

# Group the DataFrame by season

season\_groups = df.groupby('season')

# Define colors for the pie chart

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

# Iterate over each season

for season, season\_df in season\_groups:

# Count the number of matches won by each team in the current season

matches\_won = season\_df['Winner'].value\_counts()

# Get the team with the most wins in the current season

team\_with\_most\_wins = matches\_won.idxmax()

# Plot a pie chart for the current 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.pie(matches\_won, labels=matches\_won.index, autopct='%1.1f%%', startangle=140, colors=colors)

plt.title(f'Teams with Most Matches Won in Season {season}', 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()

### 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)")

## Statistical methods:

### Average runs in powerplay, middle, death overs per season:

import tkinter as tk

from tkinter import font

import pandas as pd

# Read the dataset

psl\_data = pd.read\_csv("PSL.csv")

# Extracting the over number

psl\_data['Over'] = psl\_data['Over Number'] + 1

# Creating a new column to categorize overs

psl\_data['Over\_Category'] = pd.cut(psl\_data['Over'], bins=[0, 6, 15, 20], labels=['Powerplay', 'Middle Overs', 'Death Overs'])

# Calculating total runs scored in each over category per season

runs\_per\_season\_over\_category = psl\_data.groupby(['season', 'Over\_Category'])['Total Runs'].sum().reset\_index()

# Calculating total matches per season

matches\_per\_season = psl\_data.groupby('season')['event match\_number'].nunique().reset\_index()

# Merging total runs and total matches per season

average\_runs\_per\_season\_over\_category = pd.merge(runs\_per\_season\_over\_category, matches\_per\_season, on='season')

# Calculating average runs per over category per season

average\_runs\_per\_season\_over\_category['Average Runs'] = average\_runs\_per\_season\_over\_category['Total Runs'] / average\_runs\_per\_season\_over\_category['event match\_number']

# Pivoting the table to get the desired format

average\_runs\_per\_season\_over\_category\_pivot = average\_runs\_per\_season\_over\_category.pivot\_table(index='season', columns='Over\_Category', values='Average Runs')

# Function to open a Tkinter window and print the results

def print\_results():

root = tk.Tk()

root.title("Average Score of Teams")

root.configure(bg="#282c35") # Set background color

# Create a bold font for the heading

heading\_font = font.Font(family="Helvetica", size=16, weight="bold")

# Create a font for the content

content\_font = font.Font(family="Helvetica", size=9)

# Set resolution to 1280x720

width = 1280

height = 720

# Get screen width and height

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

# Calculate x and y position for centering the window

x = (screen\_width // 2) - (width // 2)

y = (screen\_height // 2) - (height // 2)

# Set window size and position

root.geometry(f"{width}x{height}+{x}+{y}")

# Create a label for the heading with bold font and modern look

heading\_label = tk.Label(root, text="Average Score of Teams in Powerplay, Middle Overs, and Death Overs per Season:", bg="#282c35", fg="white", font=heading\_font)

heading\_label.pack()

# Create labels for each season's averages

for season in average\_runs\_per\_season\_over\_category\_pivot.index:

powerplay\_avg = average\_runs\_per\_season\_over\_category\_pivot.loc[season, 'Powerplay']

middle\_overs\_avg = average\_runs\_per\_season\_over\_category\_pivot.loc[season, 'Middle Overs']

death\_overs\_avg = average\_runs\_per\_season\_over\_category\_pivot.loc[season, 'Death Overs']

label\_text = f"Season {season}:\nPowerplay Average: {powerplay\_avg}\nMiddle Overs Average: {middle\_overs\_avg}\nDeath Overs Average: {death\_overs\_avg}"

season\_label = tk.Label(root, text=label\_text, bg="#282c35", fg="white", font=content\_font)

season\_label.pack()

root.mainloop()

# Call the function to open the Tkinter window and print the results

print\_results()

### Average wickets in powerplay, middle, death overs per season:

import tkinter as tk

from tkinter import font

import pandas as pd

# Read the dataset

psl\_data = pd.read\_csv("PSL.csv")

# Creating a new column to categorize overs

psl\_data['Over\_Category'] = pd.cut(psl\_data['Over Number'], bins=[0, 6, 15, 20], labels=['Powerplay', 'Middle Overs', 'Death Overs'])

# Filtering out rows where a wicket fell

wickets\_data = psl\_data[psl\_data['Wickets'] == 1]

# Counting the number of wickets per over category per season

wickets\_per\_season\_over\_category = wickets\_data.groupby(['season', 'Over\_Category'])['Wickets'].count().reset\_index()

# Calculating total matches per season

matches\_per\_season = psl\_data.groupby('season')['event match\_number'].nunique().reset\_index()

# Merging total wickets and total matches per season

average\_wickets\_per\_season\_over\_category = pd.merge(wickets\_per\_season\_over\_category, matches\_per\_season, on='season')

# Calculating average wickets per over category per season

average\_wickets\_per\_season\_over\_category['Average Wickets'] = average\_wickets\_per\_season\_over\_category['Wickets'] / average\_wickets\_per\_season\_over\_category['event match\_number']

# Pivoting the table to get the desired format

average\_wickets\_per\_season\_over\_category\_pivot = average\_wickets\_per\_season\_over\_category.pivot\_table(index='season', columns='Over\_Category', values='Average Wickets')

# Function to open a Tkinter window and print the results

def print\_results():

root = tk.Tk()

root.title("Average Wickets of Teams")

root.configure(bg="#282c35") # Set background color

# Create a bold font for the heading

heading\_font = font.Font(family="Helvetica", size=16, weight="bold")

# Create a font for the content

content\_font = font.Font(family="Helvetica", size=9)

# Set resolution to 1280x720

width = 1280

height = 720

# Get screen width and height

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

# Calculate x and y position for centering the window

x = (screen\_width // 2) - (width // 2)

y = (screen\_height // 2) - (height // 2)

# Set window size and position

root.geometry(f"{width}x{height}+{x}+{y}")

# Create a label for the heading with bold font and modern look

heading\_label = tk.Label(root, text="Average Wickets of Teams in Powerplay, Middle Overs, and Death Overs per Season:", bg="#282c35", fg="white", font=heading\_font)

heading\_label.pack()

# Print results in the Tkinter window

for season in average\_wickets\_per\_season\_over\_category\_pivot.index:

powerplay\_avg = average\_wickets\_per\_season\_over\_category\_pivot.loc[season, 'Powerplay']

middle\_overs\_avg = average\_wickets\_per\_season\_over\_category\_pivot.loc[season, 'Middle Overs']

death\_overs\_avg = average\_wickets\_per\_season\_over\_category\_pivot.loc[season, 'Death Overs']

label\_text = f"Season {season}:\nPowerplay Average Wickets: {powerplay\_avg}\nMiddle Overs Average Wickets: {middle\_overs\_avg}\nDeath Overs Average Wickets: {death\_overs\_avg}"

season\_label = tk.Label(root, text=label\_text, bg="#282c35", fg="white", font=content\_font)

season\_label.pack()

root.mainloop()

# Call the function to open the Tkinter window and print the results

print\_results()

### Target score’s mean, median, mode of team in every season:

import tkinter as tk

from tkinter import font

from tkinter import ttk

import pandas as pd

# Read the dataset

psl\_data = pd.read\_csv("PSL.csv")

# Calculating the average target per season of every team

average\_target\_per\_season = psl\_data.groupby(['season', 'Team'])['Target Runs'].mean().reset\_index()

# Calculating the median target per season of every team

median\_target\_per\_season = psl\_data.groupby(['season', 'Team'])['Target Runs'].median().reset\_index()

# Calculating the mode target per season of every team

mode\_target\_per\_season = psl\_data.groupby(['season', 'Team'])['Target Runs'].apply(lambda x: x.mode()[0] if len(x.mode()) > 0 else None).reset\_index()

# Merging the mean, median, and mode target per season

average\_target\_stats\_per\_season = pd.merge(average\_target\_per\_season, median\_target\_per\_season, on=['season', 'Team'], suffixes=('\_mean', '\_median'))

average\_target\_stats\_per\_season = pd.merge(average\_target\_stats\_per\_season, mode\_target\_per\_season, on=['season', 'Team'])

# Function to open a Tkinter window and print the results

def print\_results():

root = tk.Tk()

root.title("Average Target per Season of Every Team")

root.configure(bg="#282c35") # Set background color

# Create a bold font for the heading

heading\_font = font.Font(family="Helvetica", size=16, weight="bold")

# Create a font for the content

content\_font = font.Font(family="Helvetica", size=9)

# Set resolution to 1280x720

width = 1280

height = 720

# Get screen width and height

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

# Calculate x and y position for centering the window

x = (screen\_width // 2) - (width // 2)

y = (screen\_height // 2) - (height // 2)

# Set window size and position

root.geometry(f"{width}x{height}+{x}+{y}")

# Create a label for the heading with bold font and modern look

heading\_label = tk.Label(root, text="Average Target per Season of Every Team:", bg="#282c35", fg="white", font=heading\_font)

heading\_label.pack()

# Create a Canvas widget with a scrollbar

canvas = tk.Canvas(root, bg="#282c35", highlightthickness=0)

canvas.pack(side="left", fill="both", expand=True)

scrollbar = ttk.Scrollbar(root, orient="vertical", command=canvas.yview)

scrollbar.pack(side="right", fill="y")

canvas.configure(yscrollcommand=scrollbar.set)

canvas.bind("<Configure>", lambda e: canvas.configure(scrollregion=canvas.bbox("all")))

# Create a frame to contain the labels

frame = tk.Frame(canvas, bg="#282c35")

canvas.create\_window((0, 0), window=frame, anchor="nw")

# Print results in the Tkinter window

for season in average\_target\_stats\_per\_season['season'].unique():

season\_label = tk.Label(frame, text=f"Season {season}:", bg="#282c35", fg="white", font=content\_font, justify="center")

season\_label.pack()

season\_data = average\_target\_stats\_per\_season[average\_target\_stats\_per\_season['season'] == season]

for index, row in season\_data.iterrows():

team = row['Team']

mean\_target = row['Target Runs\_mean']

median\_target = row['Target Runs\_median']

mode\_target = row['Target Runs']

team\_label = tk.Label(frame, text=f"Team: {team}", bg="#282c35", fg="white", font=content\_font, justify="center")

team\_label.pack()

mean\_label = tk.Label(frame, text=f" Mean Target: {mean\_target}", bg="#282c35", fg="white", font=content\_font, justify="center")

mean\_label.pack()

median\_label = tk.Label(frame, text=f" Median Target: {median\_target}", bg="#282c35", fg="white", font=content\_font, justify="center")

median\_label.pack()

mode\_label = tk.Label(frame, text=f" Mode Target: {mode\_target}", bg="#282c35", fg="white", font=content\_font, justify="center")

mode\_label.pack()

root.mainloop()

# Call the function to open the Tkinter window and print the results

print\_results()

### Standard deviation of run rate per season:

import tkinter as tk

from tkinter import font

import pandas as pd

# Read the dataset

psl\_data = pd.read\_csv("PSL.csv")

# Calculating run rate

psl\_data['Run Rate'] = psl\_data['Total Runs'] / psl\_data['Overs']

# Calculating standard deviation of run rate per season

std\_dev\_run\_rate\_per\_season = psl\_data.groupby('season')['Run Rate'].std().reset\_index()

# Function to open a Tkinter window and print the results

def print\_results():

root = tk.Tk()

root.title("Standard Deviation of Run Rate per Season")

root.configure(bg="#282c35") # Set background color

# Create a bold font for the heading

heading\_font = font.Font(family="Helvetica", size=16, weight="bold")

# Create a font for the content

content\_font = font.Font(family="Helvetica", size=15)

# Set resolution to 1280x720

width = 1280

height = 720

# Get screen width and height

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

# Calculate x and y position for centering the window

x = (screen\_width // 2) - (width // 2)

y = (screen\_height // 2) - (height // 2)

# Set window size and position

root.geometry(f"{width}x{height}+{x}+{y}")

# Create a label for the heading with bold font and modern look

heading\_label = tk.Label(root, text="Standard Deviation of Run Rate per Season:", bg="#282c35", fg="white", font=heading\_font)

heading\_label.pack()

# Print results in the Tkinter window

for index, row in std\_dev\_run\_rate\_per\_season.iterrows():

season = row['season']

std\_dev = row['Run Rate']

label\_text = f"Season {season}:\n Standard Deviation of Run Rate: {std\_dev}"

season\_label = tk.Label(root, text=label\_text, bg="#282c35", fg="white", font=content\_font)

season\_label.pack()

root.mainloop()

# Call the function to open the Tkinter window and print the results

print\_results()

### Variance of run rate per season:

import tkinter as tk

from tkinter import font

import pandas as pd

# Read the dataset

psl\_data = pd.read\_csv("PSL.csv")

# Calculating run rate

psl\_data['Run Rate'] = psl\_data['Total Runs'] / psl\_data['Overs']

# Calculating variance of run rate per season

variance\_run\_rate\_per\_season = psl\_data.groupby('season')['Run Rate'].var().reset\_index()

# Function to open a Tkinter window and print the results

def print\_results():

root = tk.Tk()

root.title("Variance of Run Rate per Season")

root.configure(bg="#282c35") # Set background color

# Create a bold font for the heading

heading\_font = font.Font(family="Helvetica", size=16, weight="bold")

# Create a font for the content

content\_font = font.Font(family="Helvetica", size=15)

# Set resolution to 1280x720

width = 1280

height = 720

# Get screen width and height

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

# Calculate x and y position for centering the window

x = (screen\_width // 2) - (width // 2)

y = (screen\_height // 2) - (height // 2)

# Set window size and position

root.geometry(f"{width}x{height}+{x}+{y}")

# Create a label for the heading with bold font and modern look

heading\_label = tk.Label(root, text="Variance of Run Rate per Season:", bg="#282c35", fg="white", font=heading\_font)

heading\_label.pack()

# Print results in the Tkinter window

for index, row in variance\_run\_rate\_per\_season.iterrows():

season = row['season']

variance = row['Run Rate']

label\_text = f"Season {season}:\n Variance of Run Rate: {variance}"

season\_label = tk.Label(root, text=label\_text, bg="#282c35", fg="white", font=content\_font)

season\_label.pack()

root.mainloop()

### Box plot: (# Create box plot for runs scored across different venues), (# Create box plot for wickets taken across different venues), (# Create box plot for extras conceded across different venues)

# Call the function to open the Tkinter window and print the results

print\_results()

import pandas as pd

import matplotlib.pyplot as plt

# Read the dataset

psl\_data = pd.read\_csv("PSL.csv")

# Aggregate data by date and venue

psl\_data\_agg = psl\_data.groupby(['dates', 'venue']).agg({

'Total Runs': 'sum',

'Wickets': 'sum',

'runs.extras': 'sum'

}).reset\_index()

# Create box plot for runs scored across different venues

plt.figure(figsize=(15, 6)) # Increase the figure size

ax = psl\_data\_agg.boxplot(column='Total Runs', by='venue', figsize=(15,6), rot=0)

plt.title('Distribution of Runs Scored Across Different Venues', fontsize=16) # Increase title font size

plt.xlabel('Venue', fontsize=14) # Increase x-axis label font size

plt.ylabel('Runs Scored', fontsize=14) # Increase y-axis label font size

plt.xticks(fontsize=5) # Increase x-axis tick label font size

plt.show()

# Create box plot for wickets taken across different venues

plt.figure(figsize=(15, 6)) # Increase the figure size

ax = psl\_data\_agg.boxplot(column='Wickets', by='venue', figsize=(15,6), rot=0)

plt.title('Distribution of Wickets Taken Across Different Venues', fontsize=16) # Increase title font size

plt.xlabel('Venue', fontsize=14) # Increase x-axis label font size

plt.ylabel('Wickets Taken', fontsize=14) # Increase y-axis label font size

plt.xticks(fontsize=5) # Increase x-axis tick label font size

plt.show()

# Create box plot for extras conceded across different venues

plt.figure(figsize=(15, 6)) # Increase the figure size

ax = psl\_data\_agg.boxplot(column='runs.extras', by='venue', figsize=(15,6), rot=0)

plt.title('Distribution of Extras Conceded Across Different Venues', fontsize=16) # Increase title font size

plt.xlabel('Venue', fontsize=14) # Increase x-axis label font size

plt.ylabel('Extras Conceded', fontsize=14) # Increase y-axis label font size

plt.xticks(fontsize=5) # Increase x-axis tick label font size

plt.show()

## Binomial:

### …:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import binom

# Read the CSV file into a DataFrame

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

# Initialize variables to store the previous winner and toss winner

prev\_winner = None

prev\_toss\_winner = None

prev\_eventmatchnumber=None

same\_toss\_and\_winner\_count = 0

match\_count=0

for index, row in df.iterrows():

current\_winner = row['Winner']

current\_toss\_winner = row['Toss Winner']

if (current\_winner != prev\_winner or current\_toss\_winner != prev\_toss\_winner) and current\_winner == current\_toss\_winner :

# Increment the count

same\_toss\_and\_winner\_count += 1

prev\_winner = current\_winner

prev\_toss\_winner = current\_toss\_winner

for index, row in df.iterrows():

current\_eventmatchnumber = row['Winner']

if current\_eventmatchnumber != prev\_eventmatchnumber :

match\_count += 1

prev\_eventmatchnumber = current\_eventmatchnumber

match\_count=440

n = match\_count # Number of coin flips

p = same\_toss\_and\_winner\_count/match\_count # Probability of heads (success)

k\_values = np.arange(0, n + 1)

pmf = binom.pmf(k\_values, n, p)

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(k\_values, pmf)

plt.xlabel('Number of Match Win when Toss win')

plt.ylabel('Probability')

plt.title('Binomial Distribution')

plt.show()

## Regression:

### Relation between Total runs scored(x) and wickets lost(y):

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

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

# Prepare the data

X = df['Wickets'] # Independent variable: Wickets lost

Y = df['Total Runs'] # Dependent variable: Total runs scored

# Calculate the mean of X and Y

mean\_X = X.mean()

mean\_Y = Y.mean()

# Calculate the covariance of X and Y

covariance\_XY = sum((X - mean\_X) \* (Y - mean\_Y))

# Calculate the variance of X

variance\_X = sum((X - mean\_X) \*\* 2)

# Calculate the slope and intercept

slope = covariance\_XY / variance\_X

intercept = mean\_Y - slope \* mean\_X

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))

# Plotting the actual data points

plt.scatter(X, Y, color='blue', alpha=0.7, edgecolors="k", label='Data Points')

# Calculate and plot the regression line

Y\_predicted = [slope \* x + intercept for x in X]

plt.plot(X, Y\_predicted, color='red', linewidth=2.5, label='Regression Line')

# Customizing the plot

plt.title('Total Runs vs Wickets (Ball-by-Ball Data)', fontsize=16)

plt.xlabel('Wickets Lost', fontsize=14)

plt.ylabel('Total Runs Scored', fontsize=14)

plt.legend(loc='upper left', fontsize=12)

plt.grid(True)

# Show the plot

plt.show()

# Print regression parameters

print("Slope:", slope)

print("Intercept:", intercept)

### Relation between total balls(x) and total runs(y):

import csv

from collections import defaultdict

# Load the data from the CSV file

data = []

with open('PSL.csv', 'r') as file:

reader = csv.reader(file)

next(reader) # Skip the header row

for row in reader:

batter = row[4]

batter\_runs = float(row[7])

ball\_number = float(row[3])

data.append((batter, batter\_runs, ball\_number))

# Group the data by batter

batter\_data = defaultdict(list)

for batter, batter\_runs, ball\_number in data:

batter\_data[batter].append((batter\_runs, ball\_number))

# Create feature and target variables

X = []

y = []

for batter, runs\_balls in batter\_data.items():

total\_runs = sum(run for run, \_ in runs\_balls)

total\_balls = sum(1 for \_, \_ in runs\_balls)

X.append(total\_balls)

y.append(total\_runs)

# Create a linear regression model

import numpy as np

from sklearn.linear\_model import LinearRegression

X = np.array(X).reshape(-1, 1)

y = np.array(y)

model = LinearRegression()

model.fit(X, y)

# Get the coefficients, intercept, and R-squared

coef = model.coef\_[0]

intercept = model.intercept\_

r\_squared = model.score(X, y)

# Print the results

print(f'Coefficient: {coef}')

print(f'Intercept: {intercept}')

print(f'R-squared: {r\_squared}')

# Visualize the results

import matplotlib.pyplot as plt

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.scatter(X, y, color='blue')

plt.plot(X, model.predict(X), color='red', linewidth=2)

plt.xlabel('Number of Balls Faced')

plt.ylabel('Total Runs Scored')

plt.title('Linear Regression: Total Runs vs Balls Faced')

plt.show()

# 6. Conclusion