AJEENKYA
D Y PATIL UNIVERSITY
THE INNOVATION UNIVERSITY

A

MINI PROJECT REPORT ON

"T20 Player Performance Dataset (Batting & Bowling): 2021–2025"

FOR

Term Work Examination

Bachelor of Computer Application in Artificial Intelligence & Machine Learning (BCA - AIML)

Year 2025-2026

Ajeenkya DY Patil University, Pune

-Submitted By-

Mr. Manish Perjabadkar

Under the guidance of

Prof. Vivek More

Ajeenkya DY Patil University

D Y Patil Knowledge City,
Charholi Bk. Via Lohegaon,
Pune - 412105
Maharashtra (India)

Date: 15/ 04 / 2025

CERTIFICATE

This is to certified that___Manish Perjabadkar ______________
A student of **BCA(AIML) Sem-IV** URN No _2023-B-28042003
has Successfully Completed the Dashboard Report On

"T20 Player Performance Dataset (Batting & Bowling): 2021–2025"

As per the requirement of
Ajeenkya DY Patil University, Pune was carried out under my supervision.
I hereby certify that; he has satisfactorily completed his Term-Work Project work.

Place: Pune

Examiner

Introduction

Cricket is not just a sport but a data-rich field that offers immense opportunities for analytical exploration. With the rapid integration of technology and data science in modern-day cricket, analyzing player performance, team strategies, and match outcomes has become increasingly sophisticated. The availability of structured datasets allows enthusiasts, analysts, and researchers to derive meaningful insights that can influence decisions both on and off the field.

The dataset titled **"cricket_data_2025.csv"** comprises comprehensive statistical records from cricket matches played during the 2025 season. It includes crucial information such as player performance metrics, team scores, match outcomes, and other relevant indicators. By analyzing this dataset, we can uncover patterns, assess player consistency, evaluate team strategies, and even attempt to forecast outcomes using predictive models.

This project aims to utilize the dataset for an in-depth analysis of the 2025 cricket season, leveraging various data processing and visualization techniques to deliver valuable insights into the game's dynamics.

Objectives

The primary objectives of analyzing the **cricket_data_2025.csv** dataset are:

1. **To analyze player performance trends**
   Identify top-performing players across various metrics such as runs scored, wickets taken, strike rates, and economy rates.
2. **To evaluate team performances**
   Compare teams based on their win-loss ratios, average scores, bowling strength, and consistency throughout the 2025 season.
3. **To identify key match-winning factors**
   Determine which variables (e.g., toss outcome, batting first vs. second, powerplay performance) significantly influence match results.
4. **To visualize statistical patterns**
   Create visual representations (charts, graphs, heatmaps) to better understand trends in performance, scoring, and match outcomes.
5. **To develop predictive insights**
   Explore the potential for predictive modeling to forecast match results or player outputs using machine learning or statistical techniques.
6. **To generate actionable insights for strategy**
   Provide data-driven recommendations that could be useful for coaches, players, or analysts in forming match strategies.

Methodology and Approach

The approach combines data preprocessing, exploratory data analysis (EDA), visualization, and optional predictive modeling, structured as follows:

1. Data Collection and Loading

- The dataset containing cricket match data for the 2025 season was obtained in CSV format.
- It was loaded using data processing libraries such as **Pandas** in Python for further analysis.

2. Data Cleaning and Preprocessing

- Checked for missing values, duplicate entries, and inconsistent data types.
- Cleaned the data by handling null values, converting data types where necessary, and formatting fields such as dates, player names, and match outcomes.
- Created additional derived features (e.g., batting strike rate, bowling economy, win margins) for deeper analysis.

3. Exploratory Data Analysis (EDA)

- Conducted descriptive statistical analysis to understand distributions and averages across teams and players.
- Used grouping and aggregation techniques to identify trends in performance over time.
- Analyzed metrics like total runs, wickets, boundaries, and partnerships.

4. Data Visualization

- Developed charts using **Matplotlib**, **Seaborn**, or **Plotly** to visualize player statistics, team comparisons, win trends, and performance heatmaps.
- Created interactive dashboards or summaries for clearer insight delivery.

5. Correlation and Insights Extraction

- Investigated relationships between different match elements (e.g., toss result vs. match result).
- Identified significant indicators contributing to a team's success or failure.

6. Predictive Modeling (Optional)

- If predictive insights were pursued, applied machine learning algorithms such as **Logistic Regression**, **Random Forest**, or **XGBoost** to forecast outcomes like match winners or player performance.

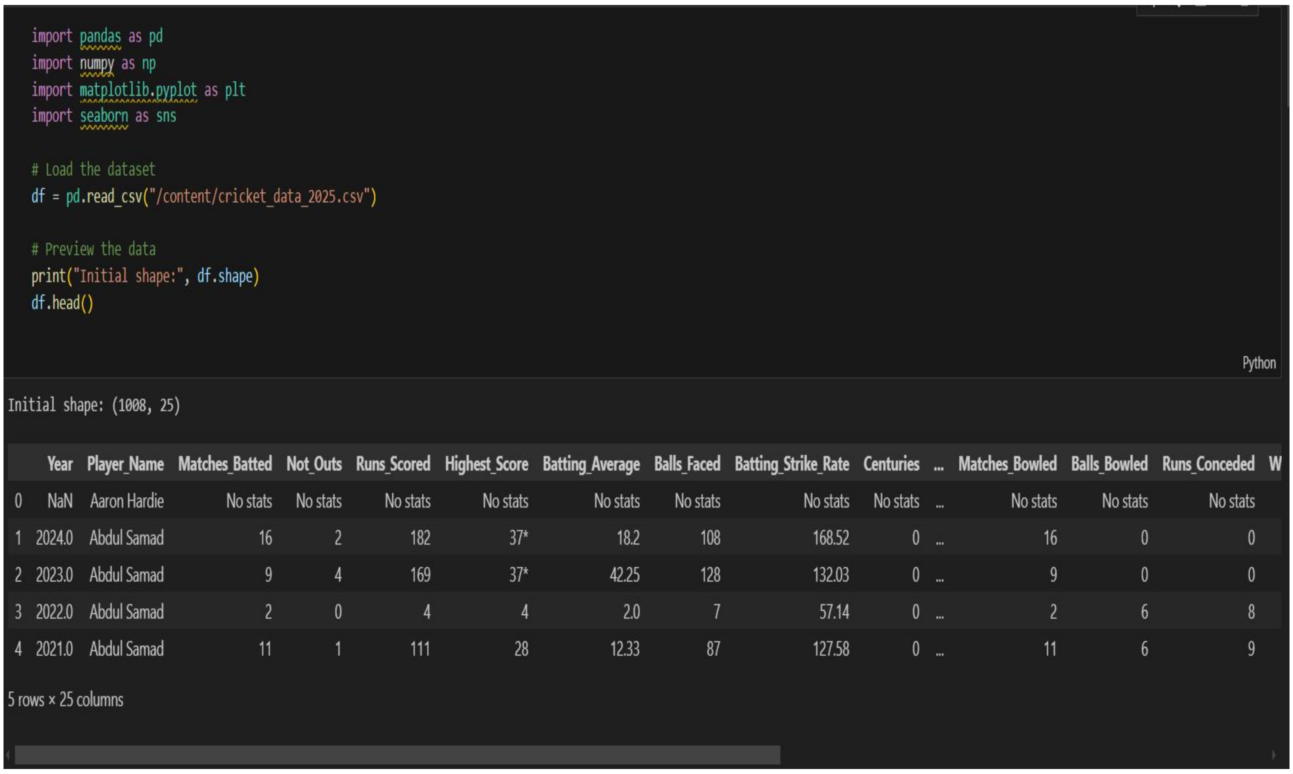
- Split data into training and testing sets, and evaluated model accuracy using appropriate metrics.

7. Result Interpretation and Reporting

- Summarized key findings in the form of reports or presentations.
- Highlighted actionable insights for coaches, analysts, or cricket enthusiasts based on the data patterns observed.

Implementation & Code:-

◇ Importing Libraries & Loading the Dataset



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv("/content/cricket_data_2025.csv")

# Preview the data
print("Initial shape:", df.shape)
df.head()
```

```
Initial shape: (1008, 25)
```

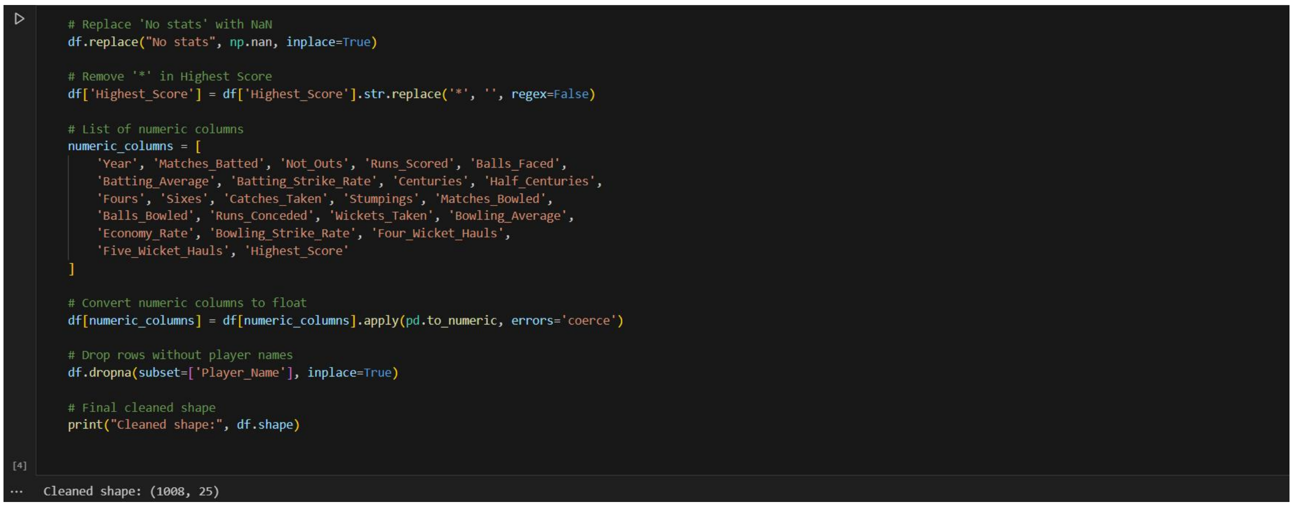
	Year	Player_Name	Matches_Batted	Not_Outs	Runs_Scored	Highest_Score	Batting_Average	Balls_Faced	Batting_Strike_Rate	Centuries	...	Matches_Bowled	Balls_Bowled	Runs_Conceded	W
0	NaN	Aaron Hardie	No stats	No stats	No stats	No stats	No stats	No stats	No stats	No stats	...	No stats	No stats	No stats	
1	2024.0	Abdul Samad	16	2	182	37*	18.2	108	168.52	0	...	16	0	0	
2	2023.0	Abdul Samad	9	4	169	37*	42.25	128	132.03	0	...	9	0	0	
3	2022.0	Abdul Samad	2	0	4	4	2.0	7	57.14	0	...	2	6	8	
4	2021.0	Abdul Samad	11	1	111	28	12.33	87	127.58	0	...	11	6	9	

5 rows × 25 columns

Explanation:

- Imports essential libraries:
  - pandas for data manipulation.
  - numpy for numerical operations.
  - matplotlib & seaborn for data visualization.
- Reads a CSV file named cricket_data_2025.csv into a DataFrame df.
- Prints the initial shape of the dataset.
- Shows the first few rows (df.head()).

◇ Data Cleaning



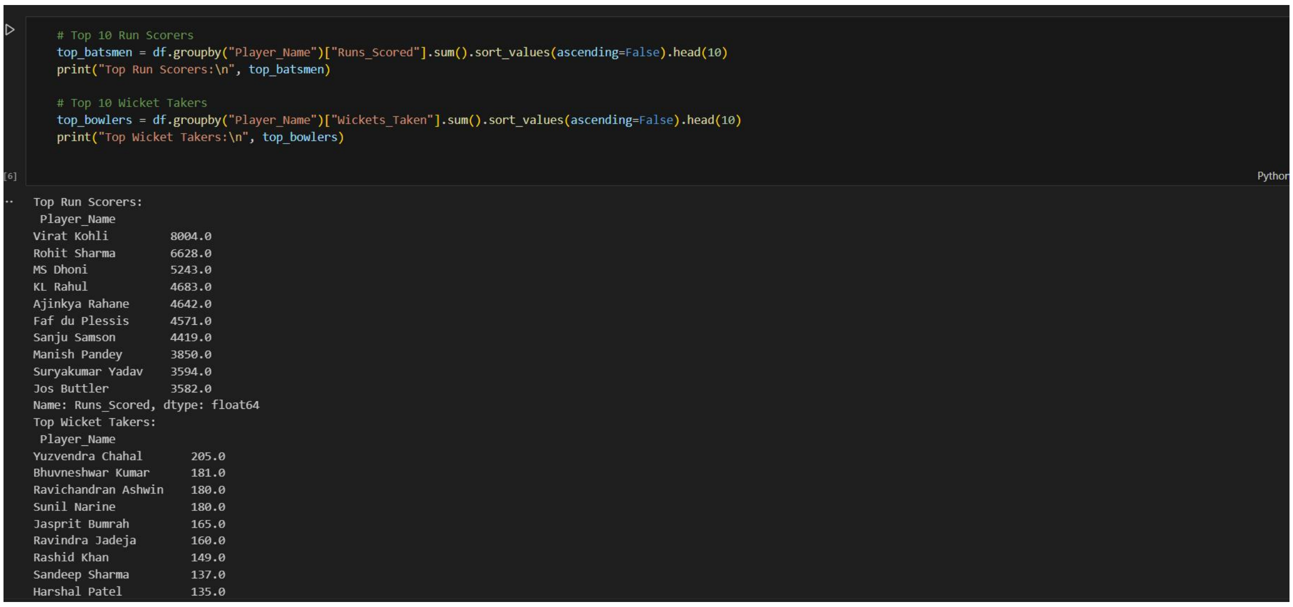
```
# Replace 'No stats' with NaN
df.replace("No stats", np.nan, inplace=True)

# Remove '*' in Highest Score
df['Highest_Score'] = df['Highest_Score'].str.replace('*', '', regex=False)

# List of numeric columns
numeric_columns = [
    'Year', 'Matches_Batted', 'Not_Outs', 'Runs_Scored', 'Balls_Faced',
    'Batting_Average', 'Batting_Strike_Rate', 'Centuries', 'Half_Centuries',
    'Fours', 'Sixes', 'Catches_Taken', 'Stumpings', 'Matches_Bowled',
    'Balls_Bowled', 'Runs_Conceded', 'Wickets_Taken', 'Bowling_Average',
    'Economy_Rate', 'Bowling_Strike_Rate', 'Four_Wicket_Hauls',
    'Five_Wicket_Hauls', 'Highest_Score'
]

# Convert numeric columns to float
df[numeric_columns] = df[numeric_columns].apply(pd.to_numeric, errors='coerce')

# Drop rows without player names
df.dropna(subset=['Player_Name'], inplace=True)

# Final cleaned shape
print("Cleaned shape:", df.shape)
```

```
Cleaned shape: (1008, 25)
```

Explanation:

- Replaces "No stats" text with NaN to mark missing values.
- Removes * symbols from 'Highest_Score' (common in cricket data to indicate "not out").
- Lists all columns that should be numeric.
- Converts those columns to numeric types, coercing errors to NaN.
- Drops rows that are missing player names (essential for analysis).
- Prints the shape of the cleaned DataFrame.

◇ Summary Statistics



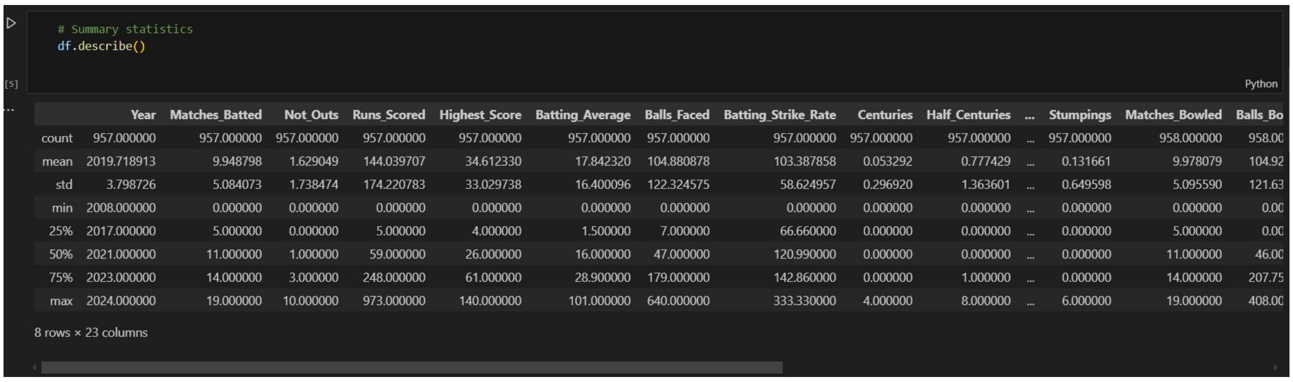
```
# Top 10 Run Scorers
top_batsmen = df.groupby("Player_Name")["Runs_Scored"].sum().sort_values(ascending=False).head(10)
print("Top Run Scorers:\n", top_batsmen)

# Top 10 Wicket Takers
top_bowlers = df.groupby("Player_Name")["Wickets_Taken"].sum().sort_values(ascending=False).head(10)
print("Top Wicket Takers:\n", top_bowlers)
```

```
Top Run Scorers:
 Player_Name
Virat Kohli          8004.0
Rohit Sharma         6628.0
MS Dhoni             5243.0
KL Rahul             4683.0
Ajinkya Rahane       4642.0
Faf du Plessis       4571.0
Sanju Samson         4419.0
Manish Pandey        3850.0
Suryakumar Yadav     3594.0
Jos Buttler          3582.0
Name: Runs_Scored, dtype: float64
Top Wicket Takers:
 Player_Name
Yuzvendra Chahal        205.0
Bhuvneshwar Kumar       181.0
Ravichandran Ashwin     180.0
Sunil Narine            180.0
Jasprit Bumrah          165.0
Ravindra Jadeja         160.0
Rashid Khan             149.0
Sandeep Sharma          137.0
Harshal Patel           135.0
```

Explanation:

- Displays basic summary statistics (mean, std, min, max, etc.) for all numeric columns using describe().

◇ Top Performers



```python
# Summary statistics
df.describe()
```

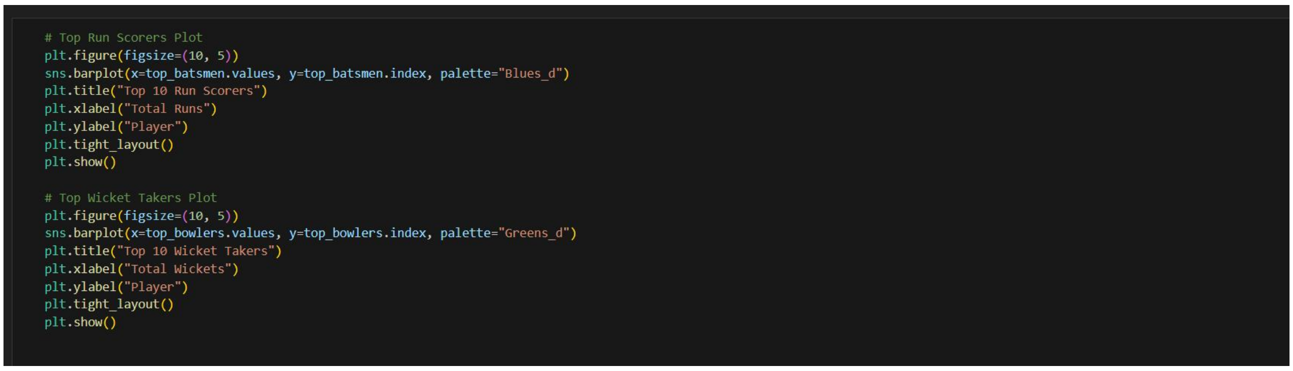
	Year	Matches_Batted	Not_Outs	Runs_Scored	Highest_Score	Batting_Average	Balls_Faced	Batting_Strike_Rate	Centuries	Half_Centuries	...	Stumpings	Matches_Bowled	Balls_Bo
count	957.000000	957.000000	957.000000	957.000000	957.000000	957.000000	957.000000	957.000000	957.000000	957.000000	...	957.000000	958.000000	958.00
mean	2019.718913	9.948798	1.629049	144.039707	34.612330	17.842320	104.880878	103.387858	0.053292	0.777429	...	0.131661	9.978079	104.92
std	3.798726	5.084073	1.738474	174.220783	33.029738	16.400096	122.324575	58.624957	0.296920	1.363601	...	0.649598	5.095590	121.63
min	2008.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.00
25%	2017.000000	5.000000	0.000000	5.000000	4.000000	1.500000	7.000000	66.660000	0.000000	0.000000	...	0.000000	5.000000	0.00
50%	2021.000000	11.000000	1.000000	59.000000	26.000000	16.000000	47.000000	120.990000	0.000000	0.000000	...	0.000000	11.000000	46.00
75%	2023.000000	14.000000	3.000000	248.000000	61.000000	28.900000	179.000000	142.860000	0.000000	1.000000	...	0.000000	14.000000	207.75
max	2024.000000	19.000000	10.000000	973.000000	140.000000	101.000000	640.000000	333.330000	4.000000	8.000000	...	6.000000	19.000000	408.00

8 rows × 23 columns

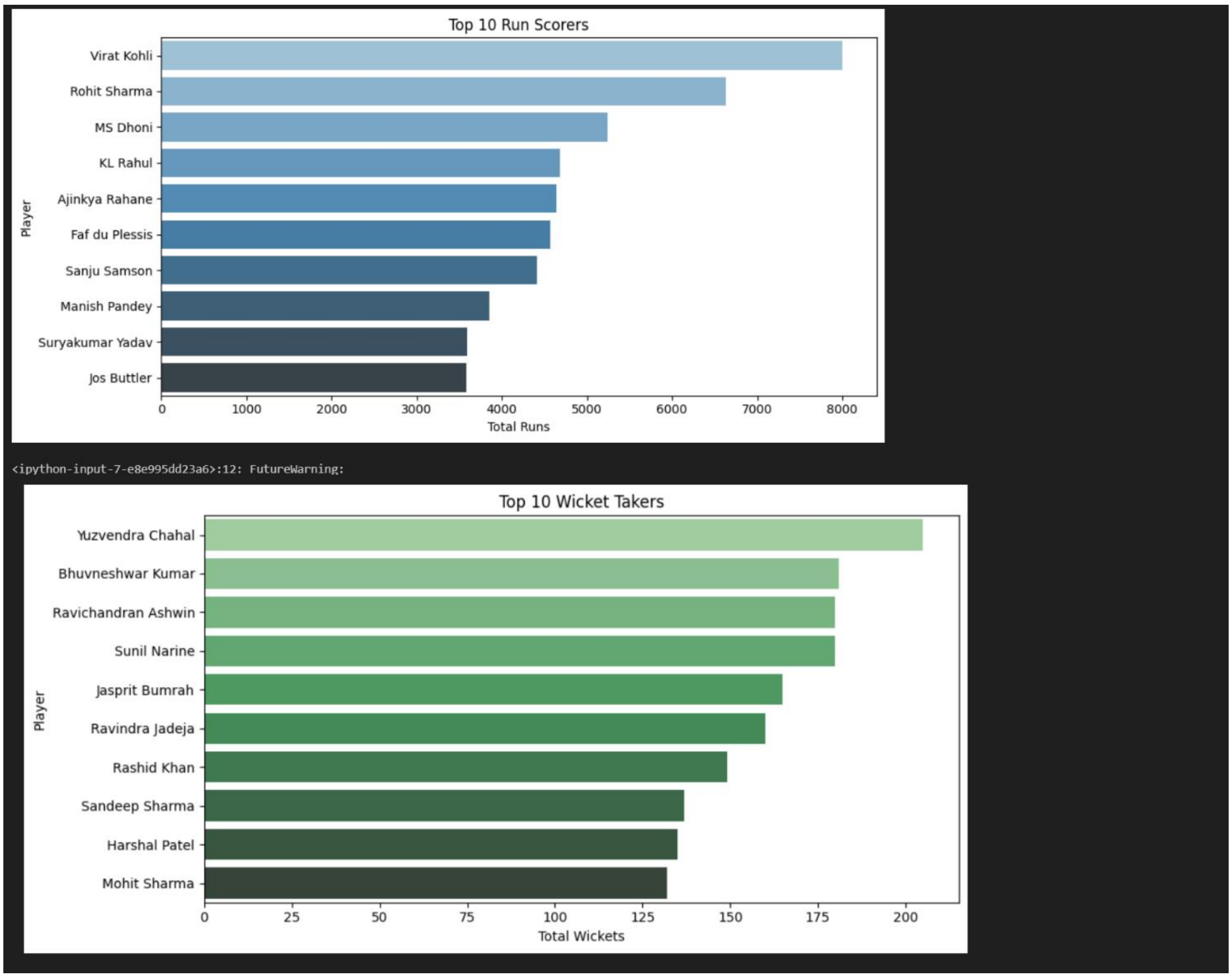
Explanation:

- Groups the dataset by player name and calculates total runs and wickets.
- Sorts the results in descending order and picks the top 10 for each category.
- Prints the top 10 batsmen and bowlers.

◇ Visualization of Top Performers



```
# Top Run Scorers Plot
plt.figure(figsize=(10, 5))
sns.barplot(x=top_batsmen.values, y=top_batsmen.index, palette="Blues_d")
plt.title("Top 10 Run Scorers")
plt.xlabel("Total Runs")
plt.ylabel("Player")
plt.tight_layout()
plt.show()

# Top Wicket Takers Plot
plt.figure(figsize=(10, 5))
sns.barplot(x=top_bowlers.values, y=top_bowlers.index, palette="Greens_d")
plt.title("Top 10 Wicket Takers")
plt.xlabel("Total Wickets")
plt.ylabel("Player")
plt.tight_layout()
plt.show()
```



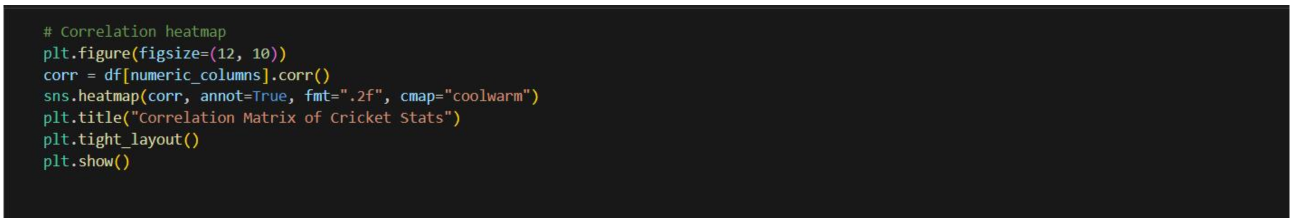
```
<ipython-input-7-e8e995dd23a6>:12: FutureWarning:
```

Explanation:

- Plots horizontal bar charts for:
  - Top 10 run scorers using a blue palette.
  - Top 10 wicket takers using a green palette.
- Uses seaborn's barplot for clean visuals.
- Adjusts figure size and layout for better readability.

Result & Visualization:-

Correlation Heatmap

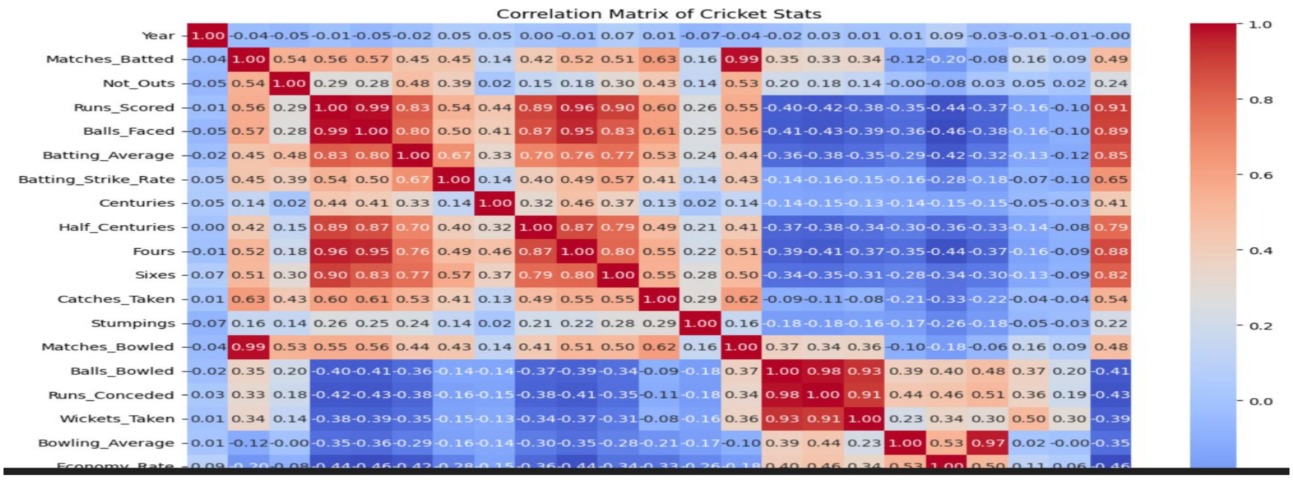


```
# Correlation heatmap
plt.figure(figsize=(12, 10))
corr = df[numeric_columns].corr()
sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Matrix of Cricket Stats")
plt.tight_layout()
plt.show()
```

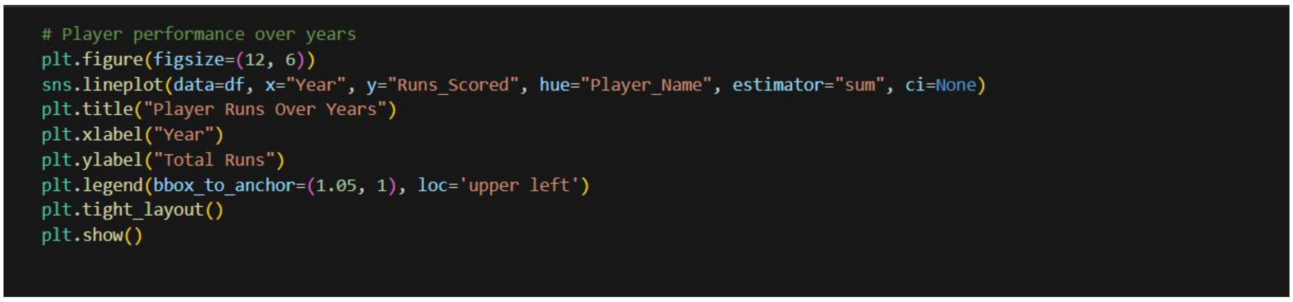
Explanation:

- **Purpose:** Visualize the pairwise correlations between numerical features in the dataset.
- **Steps:**
  - **Compute Correlation Matrix:** df[numeric_columns].corr() calculates the Pearson correlation coefficients between pairs of numeric columns.
  - **Plot Heatmap:** sns.heatmap() creates a heatmap of the correlation matrix.
    - annot=True: Displays the correlation coefficients on the heatmap.
    - fmt=".2f": Formats the annotations to two decimal places.
    - cmap="coolwarm": Uses a diverging color palette to highlight positive and negative correlations.
  - **Adjust Layout:** plt.tight_layout() ensures that labels and titles fit within the figure area.

Expected Output:



◇ Player Performance Over Years

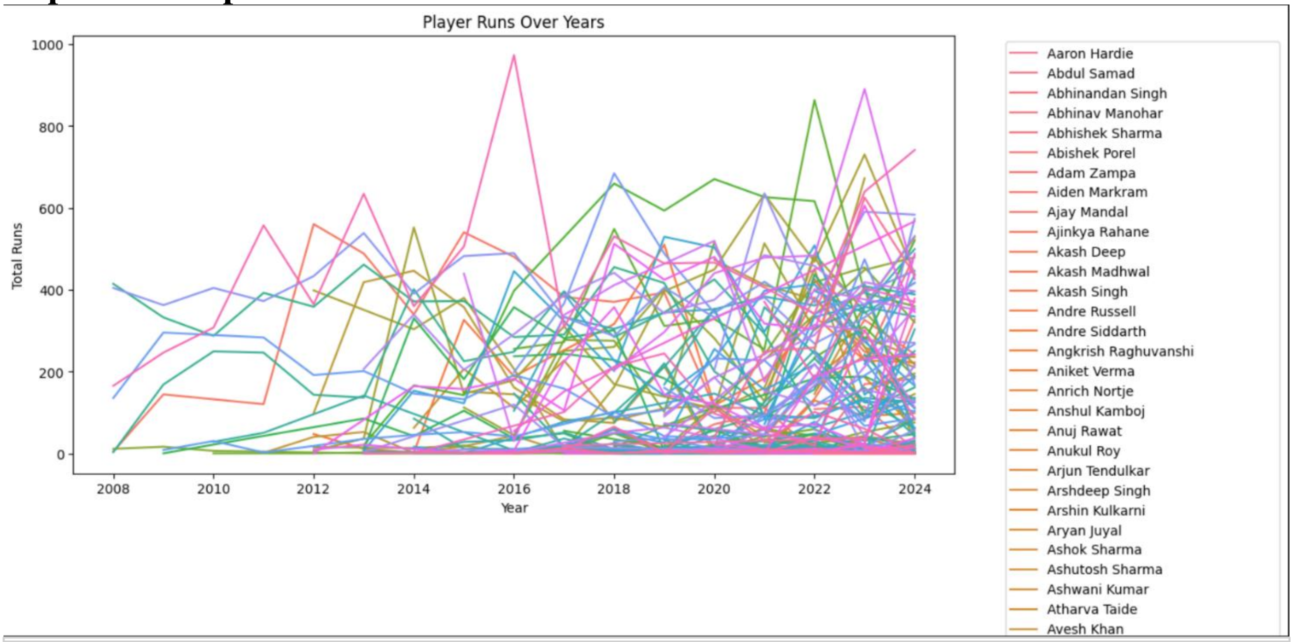


```
# Player performance over years
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x="Year", y="Runs_Scored", hue="Player_Name", estimator="sum", ci=None)
plt.title("Player Runs Over Years")
plt.xlabel("Year")
plt.ylabel("Total Runs")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

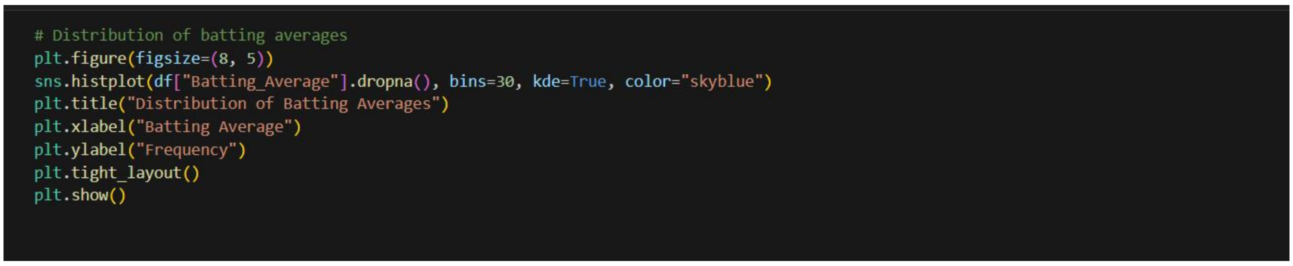
Explanation:

- **Purpose:** Analyze how players' run totals have changed over the years.
- **Steps:**
  - **Line Plot:** sns.lineplot() plots the total runs scored by each player per year.
    - x="Year": X-axis represents the year.
    - y="Runs_Scored": Y-axis represents the runs scored.
    - hue="Player_Name": Different lines for each player.
    - estimator="sum": Aggregates runs per player per year.
    - ci=None: No confidence interval shading.
  - **Customize Plot:** Titles and labels are added for clarity.
  - **Legend Placement:** Positioned outside the plot area to prevent overlap.

Expected Output:



◇ Distribution of Batting Averages

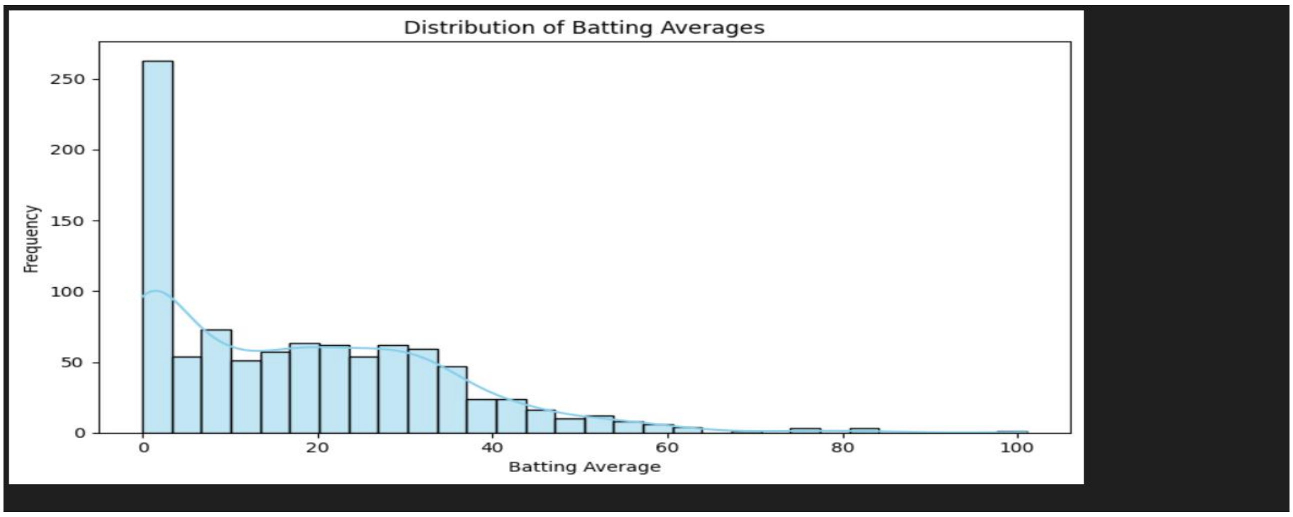


```
# Distribution of batting averages
plt.figure(figsize=(8, 5))
sns.histplot(df["Batting_Average"].dropna(), bins=30, kde=True, color="skyblue")
plt.title("Distribution of Batting Averages")
plt.xlabel("Batting Average")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```

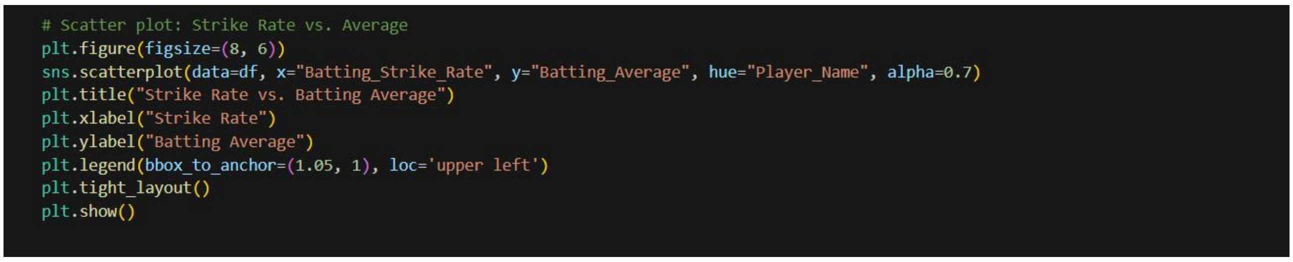
Explanation:

- **Purpose:** Understand the distribution of batting averages among players.
- **Steps:**
    - **Histogram:** sns.histplot() creates a histogram of batting averages.
        - dropna(): Excludes missing values.
        - bins=30: Divides the range into 30 intervals.
        - kde=True: Adds a Kernel Density Estimate curve to show the distribution shape.
        - color="skyblue": Sets the color of the bars.
    - **Customize Plot:** Titles and labels are added for context.

Expected Output:



◇ Code Cell 8: Scatter Plot of Strike Rate vs. Average

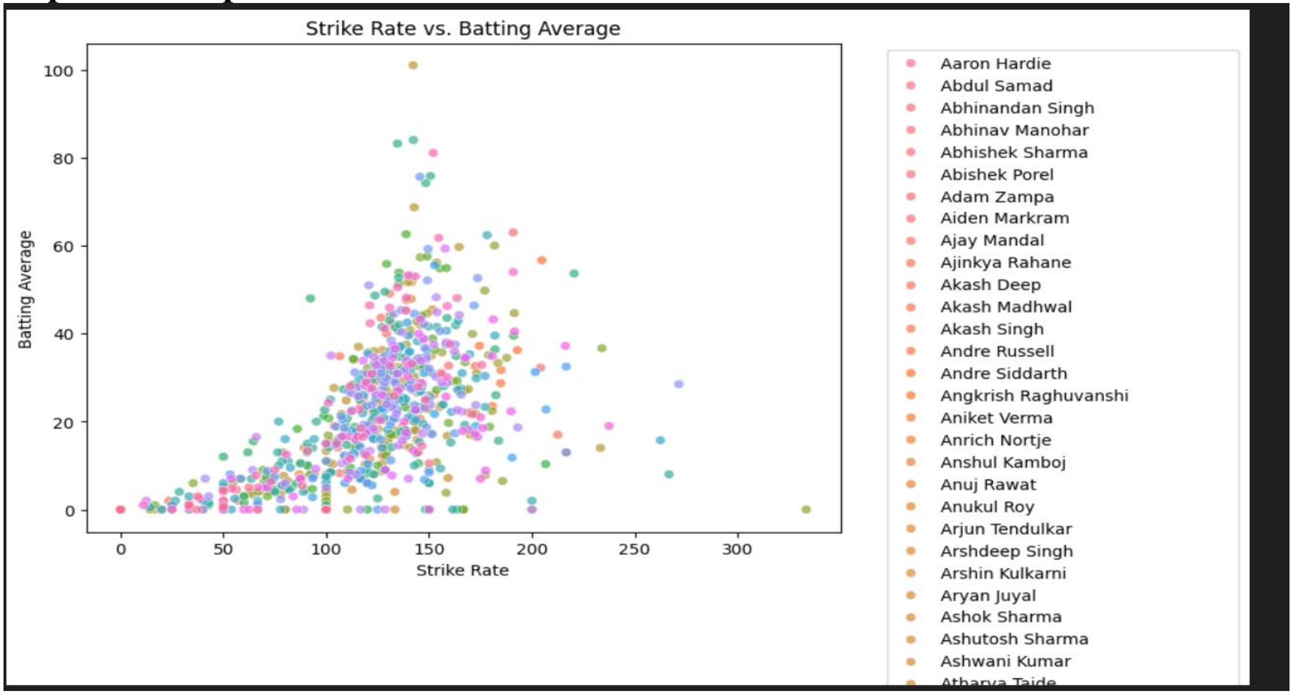


```
# Scatter plot: Strike Rate vs. Average
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x="Batting_Strike_Rate", y="Batting_Average", hue="Player_Name", alpha=0.7)
plt.title("Strike Rate vs. Batting Average")
plt.xlabel("Strike Rate")
plt.ylabel("Batting Average")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

Explanation:

- **Purpose:** Explore the relationship between batting strike rate and batting average.
- **Steps:**
  - **Scatter Plot:** sns.scatterplot() plots each player's strike rate against their batting average.
    - x="Batting_Strike_Rate": X-axis represents the strike rate.
    - y="Batting_Average": Y-axis represents the batting average.
    - hue="Player_Name": Different colors for each player.
    - alpha=0.7: Sets the transparency of the points.
  - **Customize Plot:** Titles and labels are added for clarity.
  - **Legend Placement:** Positioned outside the plot area to prevent overlap.

Expected Output:



🚀 Future Scope

1. Year-wise Performance Trends

- Analyze how individual players' performance has changed over the years.
- Visualize trends using line plots or heatmaps.

2. Player Clustering

- Use clustering algorithms (like KMeans) to group players based on similar performance metrics (e.g., all-rounders vs. specialists).

3. Performance Prediction

- Use machine learning models to predict future performance based on historical data.
- Example: Predict runs in the next season based on average, strike rate, and consistency.

4. Compare Across Teams/Leagues

- If team or league data is added, compare player performance across different contexts (IPL teams, international series, etc.).

5. Interactive Dashboards

- Use tools like Plotly Dash or Power BI to build interactive dashboards for live filtering and exploration of player stats.

6. Incorporate Contextual Stats

- Add match conditions like venue, pitch type, and opposition to analyze performance in different scenarios.

☑ Conclusion

This analysis provided insightful information about player performances in recent cricket seasons. By cleaning and aggregating the data:

- We identified **top run scorers** and **leading wicket-takers**, highlighting consistent individual contributions across multiple years.
- Visualizations revealed standout players who dominate in batting and bowling metrics.
- The dataset, once cleaned, offered a reliable foundation for trend analysis, performance benchmarking, and talent scouting.

Such insights can be crucial for teams, selectors, analysts, and fans to make data-driven decisions around player selection, team formation, and performance evaluation.