# IPL Data Mining

## Table of Contents

Table Of Figures

[illegible]

# 1. Introduction

## 1.1 Background of the Study

Cricket is one of the most popular sports in the world, this complex and intriguing game was first played in 16th century England. (Top-10 List of the World's Most Popular Sports, n.d.) Since its inception, the game has evolved in various ways. During its evolution, different forms and formats of cricket were introduced. Among all the developed formats Test cricket, One Day International Cricket and T20 cricket have emerged to be popular and have stood the test of time. The modernized version of this game T20 has revolutionized and revived cricket. This format of the game introduced a shorter and more engaging format of the game which appealed to a broad audience. The Indian Premier League (IPL) is one of the most influential platforms that has played a major role in globalizing and promoting the game of cricket.

Indian Premier League (IPL) is a T20 cricket league based in India. This tournament currently features ten franchise teams. IPL is the most successful cricket tournament in the whole world. Since its introduction in 2008 IPL has focused on commercialization and entertainment and due to this the league has been a major hit. Cricket is the most popular sport in India and the IPL capitalized on this creating a two-month cricketing carnival. (Majumdar)

Cricket is a data-rich sport, every delivery, run, and dismissal generates some sort of data. Due to this a vast amount of data is generated every time a game of cricket is played. As the game moves to a more fast-paced and dynamic format, data analysis and data-based approaches are getting more popular. In IPL these approaches are getting more popular as teams are moving towards a data-driven approach. Various Patterns and analytics are becoming increasingly valuable.

With a vast amount of IPL data available, this study aims to uncover valuable patterns and insights from IPL datasets by using various data mining techniques.

## 1.2 Problem Statement

Indian Premier league is one of the most data rich events in the world. Despite a large amount the data, most of the data and analytics are gatekept by major IPL teams, analysts and companies. A vast amount of data remains underutilized. The current analytics available tend to analyze basic statistics rather than being focused on exploring hidden data and patterns.

The lack of available data analysis has impacted several key stakeholders. Many **teams** are not able to utilize this vast data to make better team decisions and strategies. **Fans** are also missing out on a lot of data-driven insights that can make the viewing experience more pleasing. Cricket is one of the most heavily betted sports in the entire world, due to the lack of advanced data analytics in cricket, various bettors miss out on multiple patterns and trends that can support evidence-based decisions.

This study aims to bridge the gap between data analysis and analysis availability by applying various data mining techniques to the IPL dataset. The goal of this study is to uncover various hidden patterns, trends, player impact, and suitable combinations, and track how the league has evolved. These insights will benefit the team, fans, and bettors by providing them with various insights.

[illegible]

## 1.3 Objectives

Currently, various cricketing stakeholders such as teams, bettors, and fans are missing out on a lot of advanced cricketing insights. This study aims to apply different data mining techniques to IPL datasets to uncover patterns and insights that are hidden below basic statistics. The primary objectives of this study are:

- To understand and analyze IPL from a data-driven perspective to provide insights and analysis to various stakeholders such as teams, bettors, and fans.
- To analyze the evolution of IPL using time series analysis, based on trends such as run rate, and batting-bowling performances.
- To detect anomalies in player performance with the aim of identifying players who are either too good at a certain role or are too poor at a certain role.
- To group players using unsupervised machine learning models based on their performances over the years and check their impact.
- To find impactful combinations of players, focusing on pairs whose participation in a match has led to a team's success.

[illegible]

This study aims to uncover hidden patterns and analytics from IPL data, enabling teams to make more informed decisions and increase fan engagement using various data mining techniques.

# 2. Literature Review

## 2.1 Summary of Related Studies

The usage of data mining techniques in sports analytics has grown significantly in recent years. Especially for a sport like cricket, which generates data in every small event, usage of various data mining techniques can be applied. In recent years, a large amount of analysis has been done on cricket-related datasets. Due to the data-rich nature of cricket, it has been one of the most explored sports.

"**Indian Premier League and World Cricket**" from The Cambridge Companion to Cricket explores how IPL has transformed cricket in recent years. This chapter examines how IPL has grown in stature with the combination of commercialization and entertainment. This chapter also explains how data-rich sports are and how this data-rich nature of sports has influenced games and fan engagement. (Majumdar)

"**Machine Learning to Cluster Cricket Players**" by Laksmi Ajay demonstrates how clustering can be used to analyze player performance from a data-based perspective. This paper explores how various clustering techniques can be used to group player performance into meaningful categories. This paper explores K-means, hierarchical clustering, and other techniques to cluster players in various clusters based on player performance. This paper also explains how these techniques can help in team selection and team building. (Ajay, 2021)

"**Cricket Players Performance Prediction and Evaluation Using Machine Learning Algorithms**" explores the usage of various machine learning techniques to predict and evaluate players' performance. Different methods such as linear regression, K-means clustering, and random forest classification have been used in this study. Random forest was used to validate the clustering outcomes. This research shows that multiple data mining techniques can be used to produce reliable player performance-based analysis. (Sumathi, Prabu, & Murugesan, 2023)

"**Is Jasprit Bumrah a Genius Bowler? Using AutoEncoders for Anomaly Detection in Cricket**" This article shows how anomaly detection can be used to identify rare and exceptional players and player performances in cricket. This study uses various feature engineering techniques to determine these anomalies. This work highlights how various unsupervised learning and deep learning techniques can offer valuable insights. (Balajisr, 2024)

"**Complex Network Analysis in Cricket: Community Structure, player's Role, and Performance Index**" explores applications of network theory to cricket by modeling various batting metrics and a network. This study shows how various graph mining and network analyses capture various strategic importance of player relationships. This study offers a more nuanced view of a player's impact on the game. (Mukherjee, 2013)

These literatures align with the objectives of this project. These studies demonstrate how various data mining techniques such as clustering, association rule mining, graph mining, and anomaly detection can be applied to cricket-related datasets to extract hidden insights and meaningful patterns.

## 2.2 Proposed methods and techniques

Based on the literature reviewed to extract meaningful insights from the Indian Premier League dataset four data mining techniques have been chosen. Each data mining technique is aimed at fulfilling our objectives. Using this data mining a deeper understanding of the game can be gathered from a data-driven perspective.

The following are the data mining techniques used in this study.

### 2.2.1 Times Series Analysis

Times Series Analysis is a sequence of various observations gathered throughout a particular time frame. This data mining technique is used to identify trends between intervals, to identify seasonal patterns, and changes, and possibly to make future predictions. (Wei)

This data mining technique can be applied to observe how the IPL has evolved since its inception. Using various independent variables and feature engineering techniques metrics such as run rates, averages, and strike rates were produced. These metrics were further analyzed over time. This technique helps us identify how team strategies and gameplay have evolved and differed across different seasons.

### 2.2.2 Association Analysis of Player Pairs

This analysis combines association analysis and frequency itemset mining. This analysis is like market basket analysis, where player pairs appear together in teams' wins are analyzed. This technique explores the relationship between combinations and team success. Rather than using formal rule mining algorithms, this approach is based on participation frequency and outcomes. In team sports, player synergy and partnerships are vital and can significantly influence results.

### 2.2.3 K-Means Clustering

K-Means Clustering is an unsupervised machine learning algorithm in which datasets without any label are grouped into various clusters. In this technique, various clusters are created among data that are like each other. (Adams)

K-means clustering can help identify various player groups based on their performance metrics and statistics. These clusters can reveal various natural groups like consistent performers, strike bowlers, power hitters, etc. This technique can give insights into what kind of a performer a player is.

### 2.2.4 Anomaly Detection (Isolation Forest)

Isolation Forest is an anomaly detection algorithm in which observations that are significantly different from normal data are isolated. This algorithm works by identifying outliers by isolating them in a random forest. (IsolationForest, n.d.) This technique can help us identify exceptional players or poor players. Identifying these anomalies can help identify high or low-impact performers.

## 2.3 Justification for the Methods Selected

Upon reviewing the IPL dataset and related literature these data mining techniques were chosen. The objective of this project is to uncover hidden insights, patterns, and trends related to IPL. Each method has been chosen to extract certain information. The information extracted using these techniques can be relevant to team strategy, player performance identification, and league evolution.

The following are the justifications for each selected method:

### 2.3.1 Justification for Time Series Analysis

IPL is a tournament that has evolved a lot since its inception in 2008. As the years have passed almost all the aspects of the way this tournament is played have changed. Time series analysis is very well suited to studying trends over time. Time series analysis will allow stakeholders to understand how the league progresses and changes in strategies. This analysis will also give our stakeholders ideas about whether the season's prior seasons have been batting or bowling dominated.

### 2.3.2 Justification for Association Analysis of Player Pairs

This method is chosen to explore how team combinations and synergy between pairs affect match outcomes. In cricket, partnerships play a crucial role. Analyzing how frequently specific players appear together in winning matches can give our stakeholders an idea about team selection, and pair synergy and can provide insight into appropriate selection for building a fantasy team. This method is particularly valuable for team selection.

### 2.3.3 Justification for K-Means Clustering

K-Means clustering is chosen due to its ability to identify natural clusters based on various performance statistics. In cricket, there might be various types of roles. This method helps our stakeholders identify the type of players such as hard hitters, strike bowlers, high-performance batters, etc. This method can help our stakeholders understand what kind of players they have and what players they can select to make the team composition even better.

### 2.3.4 Justification for Anomaly Detection (Isolation Forest)

Isolation forest is selected to identify players who have either performed exceptionally well or exceptionally poorly than their counterparts. By detecting these anomalies we can identify instances in which an individual has overperformed or underperformed. These anomalies can help teams identify their key players, rising talents, and underperforming players. Insights generated from this data mining technique can also help our stakeholders to make better decisions or bets. This technique also can influence player retention and team structure.

# 3. Methodology

This section outlines the step-by-step approach that has been used to analyze the Indian premier league dataset to discover meaningful insights and patterns using data mining techniques.

## 3.1 Dataset Description and Source

Based on the literature reviewed and the roadmaps laid for this project, a dataset containing ball-by-ball IPL data and match-based data was required. To meet this requirement **IPL Complete Dataset (2008-2024)** which is available at Kaggle was selected.

This data contains ball-by-ball data and matches-based data of IPL from 2008 to 2024. This dataset includes two files "**deliveries.csv**" and "**matches.csv**".

**deliveries.csv** – This file contains ball-by-ball information of each delivery bowled in IPL from 2008 to 2024. This file includes information such as **match ID, innings, batting team, bowling team, over, ball, batter, bowler, non-striker, extra runs, total runs, extra type, did wicket fall, dismissed player, mode of dismissal, and the fielder involved**. This detailed data is critical for our study as ball-by-ball data is critical for generating player performance metrics such as bowling average, batting average, and strike rates. This data is also important in identifying player patterns and constructing player relationships.

**matches.csv** – This file contains match-level information for each IPL game that has been played from 2008 to 2024. It includes columns such as **match ID, season, city, date, venue, toss winner, match winner, result type, result margin, etc**. This dataset is critical to analyzing match outcomes, match winners, player pair performances, etc.

By a combination of these datasets, both match level and player analysis can be done. This dataset supports various data mining techniques that can be used to uncover hidden patterns, trends, and anomalies.

**Source of this dataset:**

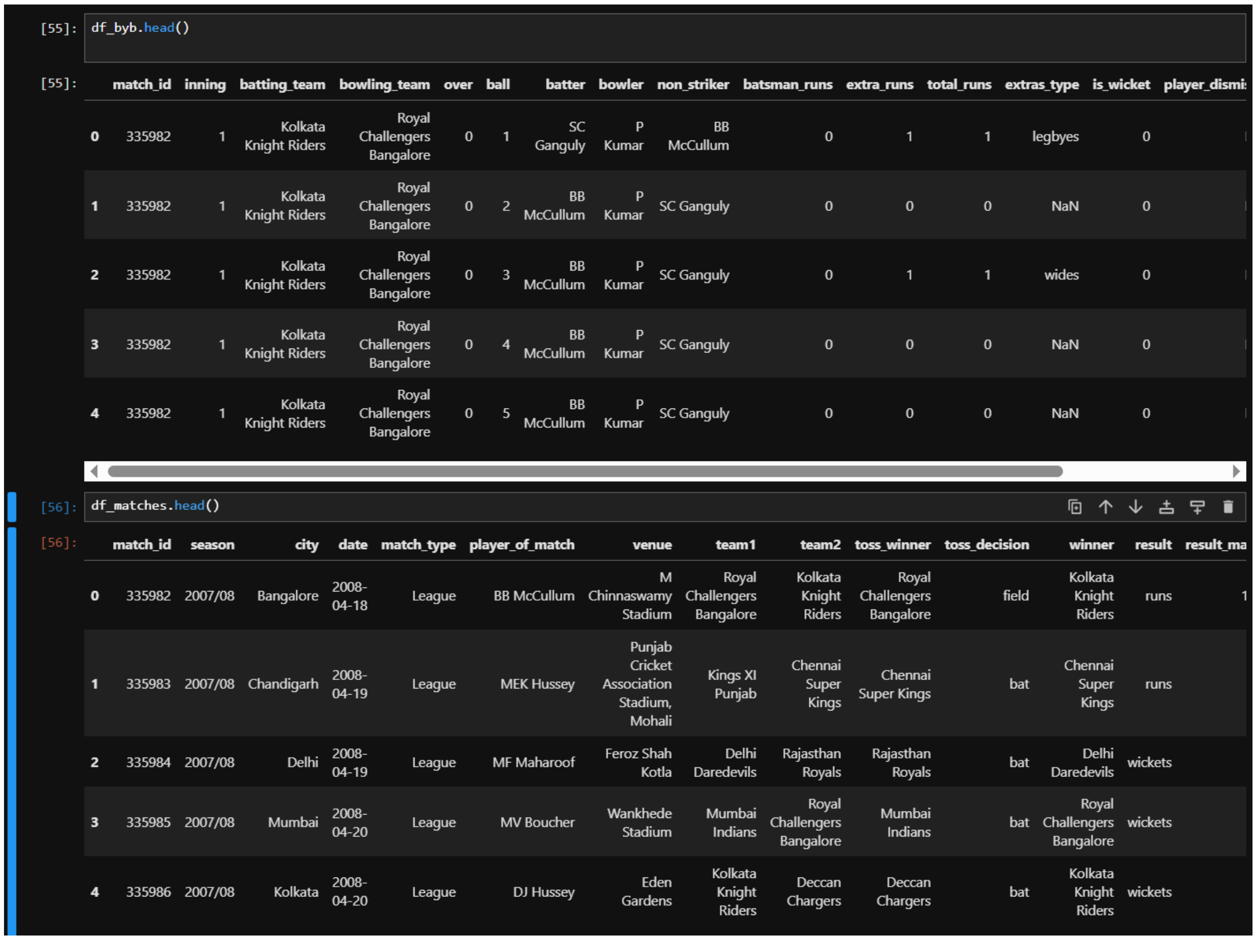
Kaggle: **IPL Complete Dataset (2008-2024)**

## 3.2 Data preprocessing steps

Data preprocessing is a process that can prepare data for further analysis and application of data mining techniques. According to the needs of the project, various data preprocessing techniques can be implemented.

### 3.2.1 Initial Data Exploration

After importing the dataset into our notebook, it is necessary to explore the data at a surface level. This can give us a better understanding of how useful a dataset is and can give us an idea about what data-cleaning techniques need to be employed.



```
[55]: df_byb.head()
```

[55]:

| | match_id | inning | batting_team | bowling_team | over | ball | batter | bowler | non_striker | batsman_runs | extra_runs | total_runs | extras_type | is_wicket | player_dismi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 335982 | 1 | Kolkata Knight Riders | Royal Challengers Bangalore | 0 | 1 | SC Ganguly | P Kumar | BB McCullum | 0 | 1 | 1 | legbyes | 0 | |
| 1 | 335982 | 1 | Kolkata Knight Riders | Royal Challengers Bangalore | 0 | 2 | BB McCullum | P Kumar | SC Ganguly | 0 | 0 | 0 | NaN | 0 | |
| 2 | 335982 | 1 | Kolkata Knight Riders | Royal Challengers Bangalore | 0 | 3 | BB McCullum | P Kumar | SC Ganguly | 0 | 1 | 1 | wides | 0 | |
| 3 | 335982 | 1 | Kolkata Knight Riders | Royal Challengers Bangalore | 0 | 4 | BB McCullum | P Kumar | SC Ganguly | 0 | 0 | 0 | NaN | 0 | |
| 4 | 335982 | 1 | Kolkata Knight Riders | Royal Challengers Bangalore | 0 | 5 | BB McCullum | P Kumar | SC Ganguly | 0 | 0 | 0 | NaN | 0 | |

```
[56]: df_matches.head()
```

[56]:

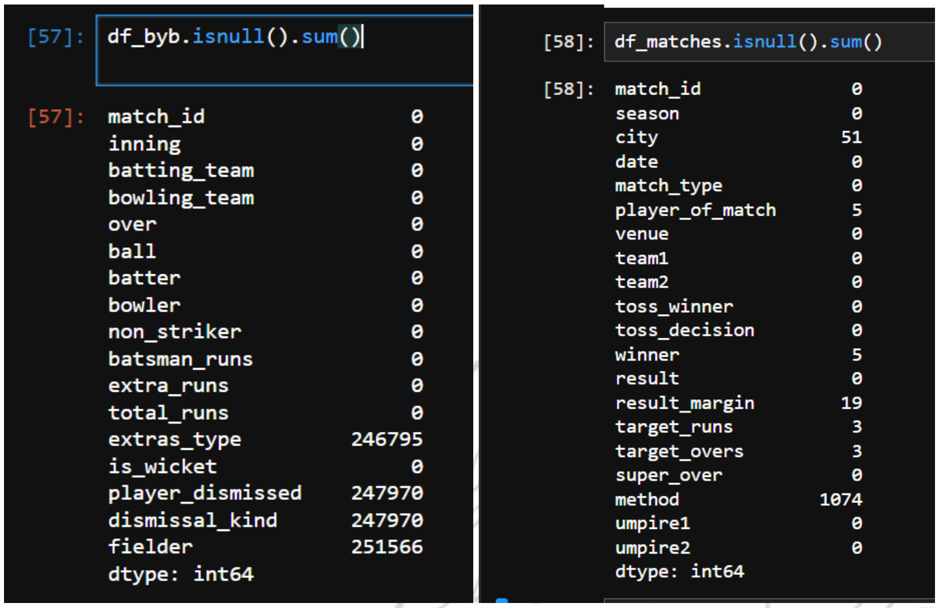
| | match_id | season | city | date | match_type | player_of_match | venue | team1 | team2 | toss_winner | toss_decision | winner | result | result_ma |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 335982 | 2007/08 | Bangalore | 2008-04-18 | League | BB McCullum | M Chinnaswamy Stadium | Royal Challengers Bangalore | Kolkata Knight Riders | Royal Challengers Bangalore | field | Kolkata Knight Riders | runs | 1 |
| 1 | 335983 | 2007/08 | Chandigarh | 2008-04-19 | League | MEK Hussey | Punjab Cricket Association Stadium, Mohali | Kings XI Punjab | Chennai Super Kings | Chennai Super Kings | bat | Chennai Super Kings | runs | |
| 2 | 335984 | 2007/08 | Delhi | 2008-04-19 | League | MF Maharoof | Feroz Shah Kotla | Delhi Daredevils | Rajasthan Royals | Rajasthan Royals | bat | Delhi Daredevils | wickets | |
| 3 | 335985 | 2007/08 | Mumbai | 2008-04-20 | League | MV Boucher | Wankhede Stadium | Mumbai Indians | Royal Challengers Bangalore | Mumbai Indians | bat | Royal Challengers Bangalore | wickets | |
| 4 | 335986 | 2007/08 | Kolkata | 2008-04-20 | League | DJ Hussey | Eden Gardens | Kolkata Knight Riders | Deccan Chargers | Deccan Chargers | bat | Kolkata Knight Riders | wickets | |

*Figure 1: Initial Data Exploration*

## 3.2.2 Data Cleaning

### Checking for Missing Values

One of the major issues with datasets is missing values. These missing values either need to be treated imputed or removed to properly use data mining techniques.



```
[57]: df_byb.isnull().sum()

[57]: match_id                0
      inning                  0
      batting_team            0
      bowling_team            0
      over                    0
      ball                    0
      batter                  0
      bowler                  0
      non_striker             0
      batsman_runs            0
      extra_runs              0
      total_runs              0
      extras_type        246795
      is_wicket               0
      player_dismissed   247970
      dismissal_kind     247970
      fielder            251566
      dtype: int64
```

```
[58]: df_matches.isnull().sum()

[58]: match_id              0
      season                0
      city                 51
      date                  0
      match_type            0
      player_of_match       5
      venue                 0
      team1                 0
      team2                 0
      toss_winner           0
      toss_decision         0
      winner                5
      result                0
      result_margin        19
      target_runs           3
      target_overs          3
      super_over            0
      method             1074
      umpire1               0
      umpire2               0
      dtype: int64
```

*Figure 2: Checking for Null Values*

There are a lot of missing or null values in this dataset. These columns are out of the current scope of this project. There are other columns that are not necessary for the current scope of this project. Due to this, instead of **removing or imputing** various **feature selection** and **feature engineering techniques** will be used to select only those data that are necessary for this project.

### Outlier Detection and Treatment

As this dataset contains ball-by-ball IPL data and match-related data, outliers present in this dataset are naturally occurring and are critical for further analysis. For example, 5 total runs in a legal delivery are entirely possible, but this will be flagged as an outlier. Due to this advanced and robust analysis, all the outliers in this dataset must be retained.

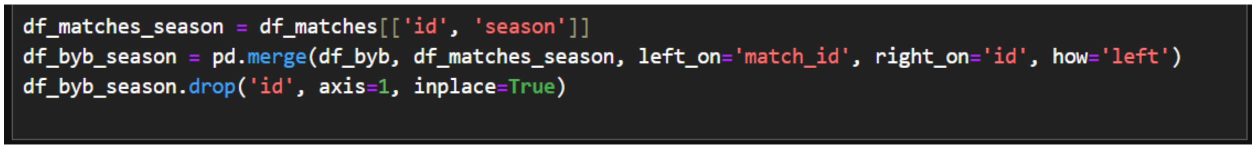
Rather than data cleaning techniques **feature selection** and **feature engineering** will be applied in this project.

### 3.2.3 Feature Selection and Feature Engineering

Feature selection is a step in which relevant features are selected from existing datasets. This is a critical step in data preparation and processing.

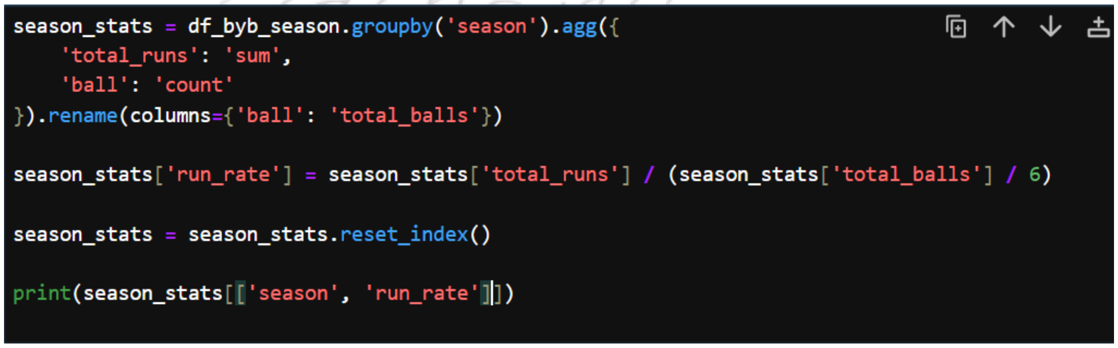
#### Feature selection and Engineering for Time Series Analysis

For Time Series Analysis ball by ball IPL data and matches data is merged using match id as a common column. Season was also merged into ball-by-ball data for this analysis.



```
df_matches_season = df_matches[['id', 'season']]
df_byb_season = pd.merge(df_byb, df_matches_season, left_on='match_id', right_on='id', how='left')
df_byb_season.drop('id', axis=1, inplace=True)
```

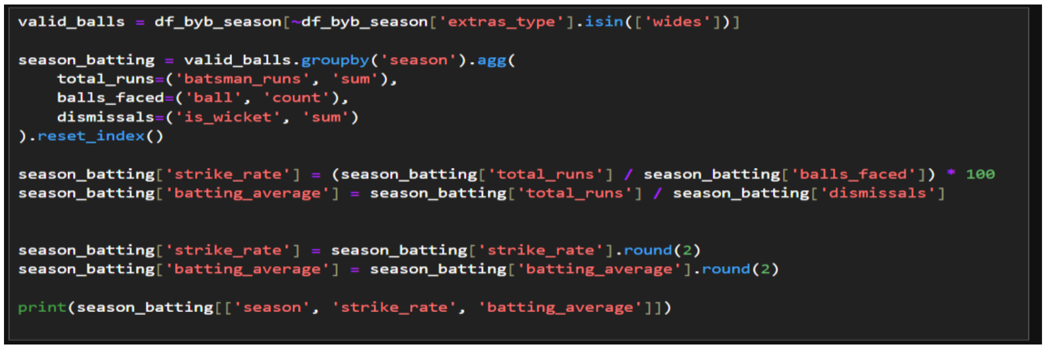
For time series analysis using season, total runs per match, total balls and dividing it by 6 we feature engineered run rate.



```
season_stats = df_byb_season.groupby('season').agg({
    'total_runs': 'sum',
    'ball': 'count'
}).rename(columns={'ball': 'total_balls'})

season_stats['run_rate'] = season_stats['total_runs'] / (season_stats['total_balls'] / 6)

season_stats = season_stats.reset_index()

print(season_stats[['season', 'run_rate']])
```

*Figure 3: Run Rate Calculation*

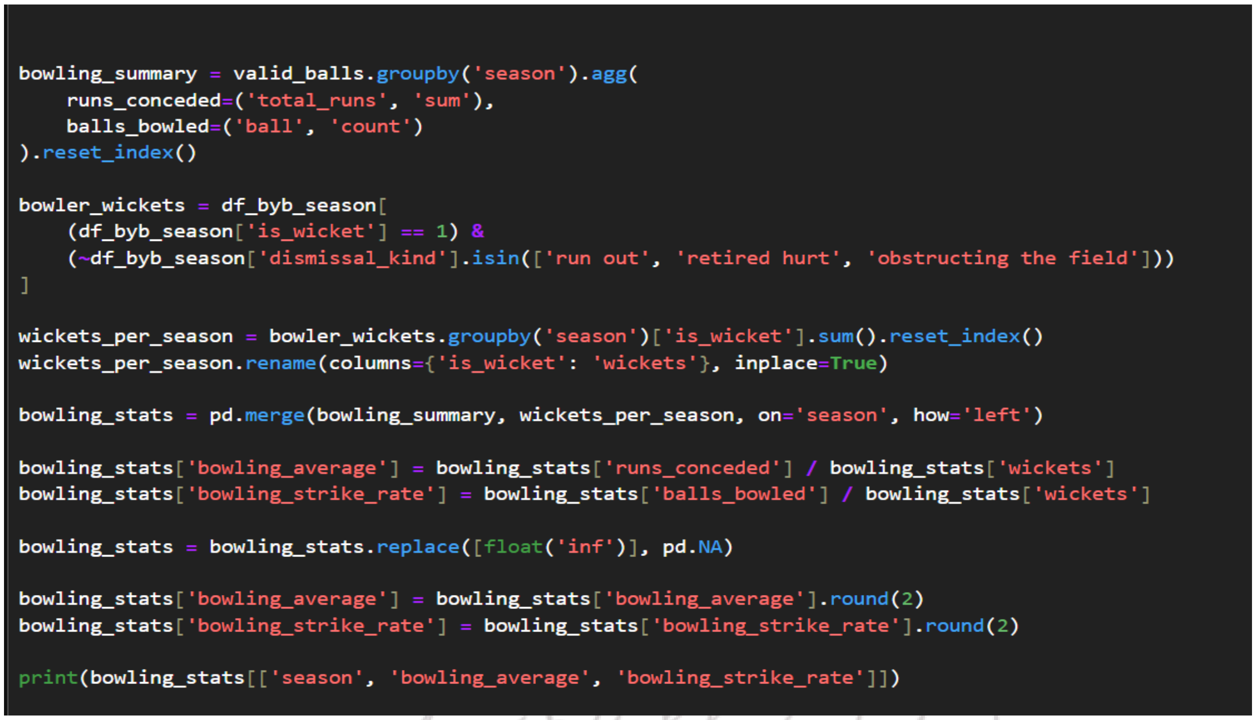
For time series analysis of batting metrics features in dataset like batsman run, ball, is_wicket, batting strike rate and batting average was feature engineered.



```
valid_balls = df_byb_season[~df_byb_season['extras_type'].isin(['wides'])]

season_batting = valid_balls.groupby('season').agg(
    total_runs=('batsman_runs', 'sum'),
    balls_faced=('ball', 'count'),
    dismissals=('is_wicket', 'sum')
).reset_index()

season_batting['strike_rate'] = (season_batting['total_runs'] / season_batting['balls_faced']) * 100
season_batting['batting_average'] = season_batting['total_runs'] / season_batting['dismissals']


season_batting['strike_rate'] = season_batting['strike_rate'].round(2)
season_batting['batting_average'] = season_batting['batting_average'].round(2)

print(season_batting[['season', 'strike_rate', 'batting_average']])
```

*Figure 4:Batting Metrics Calculation*

For time series analysis of bowling metrics total_runs, ball and is_wicket are combined, and then new features such as bowling_average and batting_average are engineered.

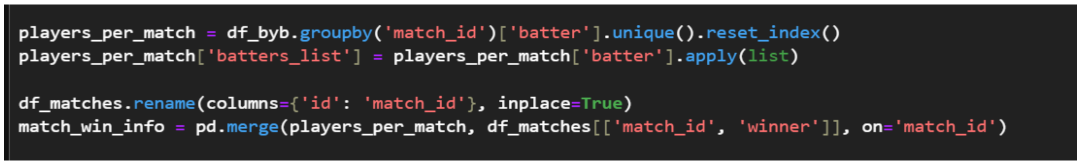


```python
bowling_summary = valid_balls.groupby('season').agg(
    runs_conceded=('total_runs', 'sum'),
    balls_bowled=('ball', 'count')
).reset_index()

bowler_wickets = df_byb_season[
    (df_byb_season['is_wicket'] == 1) &
    (~df_byb_season['dismissal_kind'].isin(['run out', 'retired hurt', 'obstructing the field']))
]

wickets_per_season = bowler_wickets.groupby('season')['is_wicket'].sum().reset_index()
wickets_per_season.rename(columns={'is_wicket': 'wickets'}, inplace=True)

bowling_stats = pd.merge(bowling_summary, wickets_per_season, on='season', how='left')

bowling_stats['bowling_average'] = bowling_stats['runs_conceded'] / bowling_stats['wickets']
bowling_stats['bowling_strike_rate'] = bowling_stats['balls_bowled'] / bowling_stats['wickets']

bowling_stats = bowling_stats.replace([float('inf')], pd.NA)

bowling_stats['bowling_average'] = bowling_stats['bowling_average'].round(2)
bowling_stats['bowling_strike_rate'] = bowling_stats['bowling_strike_rate'].round(2)

print(bowling_stats[['season', 'bowling_average', 'bowling_strike_rate']])
```

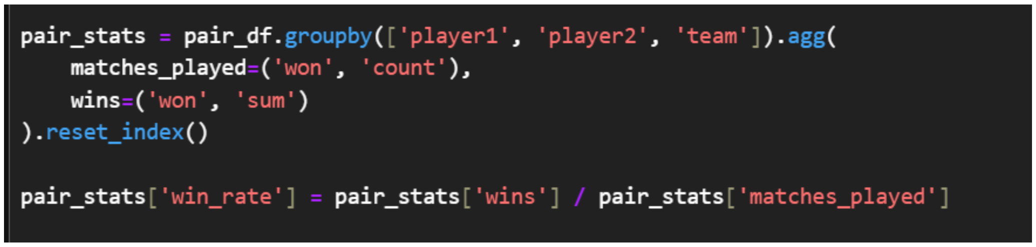
*Figure 5: Bowling Metrics Calculation*

## Feature Selection and Engineering for Association Analysis of Player Pair

Firstly, which batter have batted in matches was selected, then player who have appeared in matches was feature engineered, lastly wins and matches played were selected  and win rate was engineered



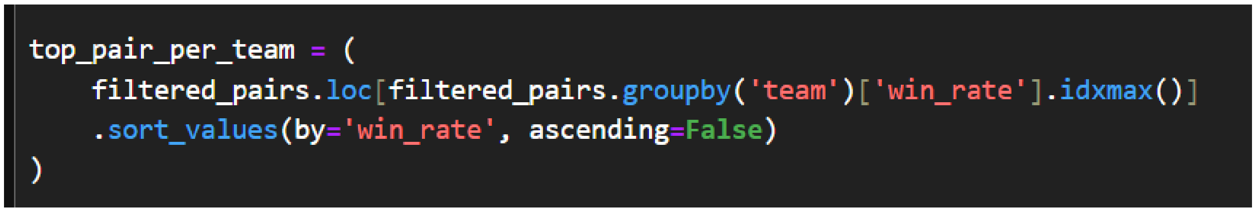
```python
players_per_match = df_byb.groupby('match_id')['batter'].unique().reset_index()
players_per_match['batters_list'] = players_per_match['batter'].apply(list)

df_matches.rename(columns={'id': 'match_id'}, inplace=True)
match_win_info = pd.merge(players_per_match, df_matches[['match_id', 'winner']], on='match_id')
```



```python
pair_stats = pair_df.groupby(['player1', 'player2', 'team']).agg(
    matches_played=('won', 'count'),
    wins=('won', 'sum')
).reset_index()

pair_stats['win_rate'] = pair_stats['wins'] / pair_stats['matches_played']
```

*Figure 6: Win Rate Calculation*

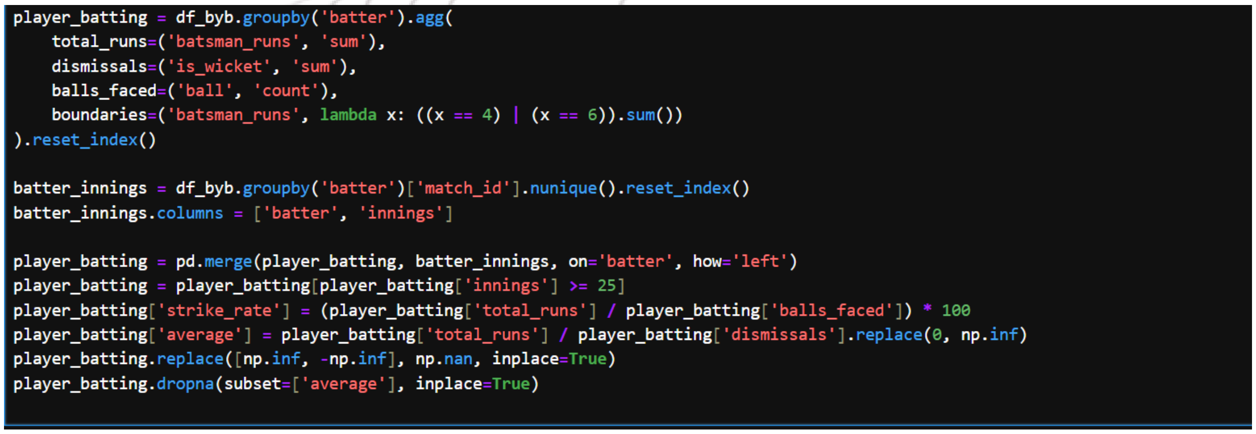
Then features "team" and "win_rate" were selected to find out top winning combinations for a team.



```
top_pair_per_team = (
    filtered_pairs.loc[filtered_pairs.groupby('team')['win_rate'].idxmax()]
    .sort_values(by='win_rate', ascending=False)
)
```

## Feature Selection and Engineering for K-Means Clustering

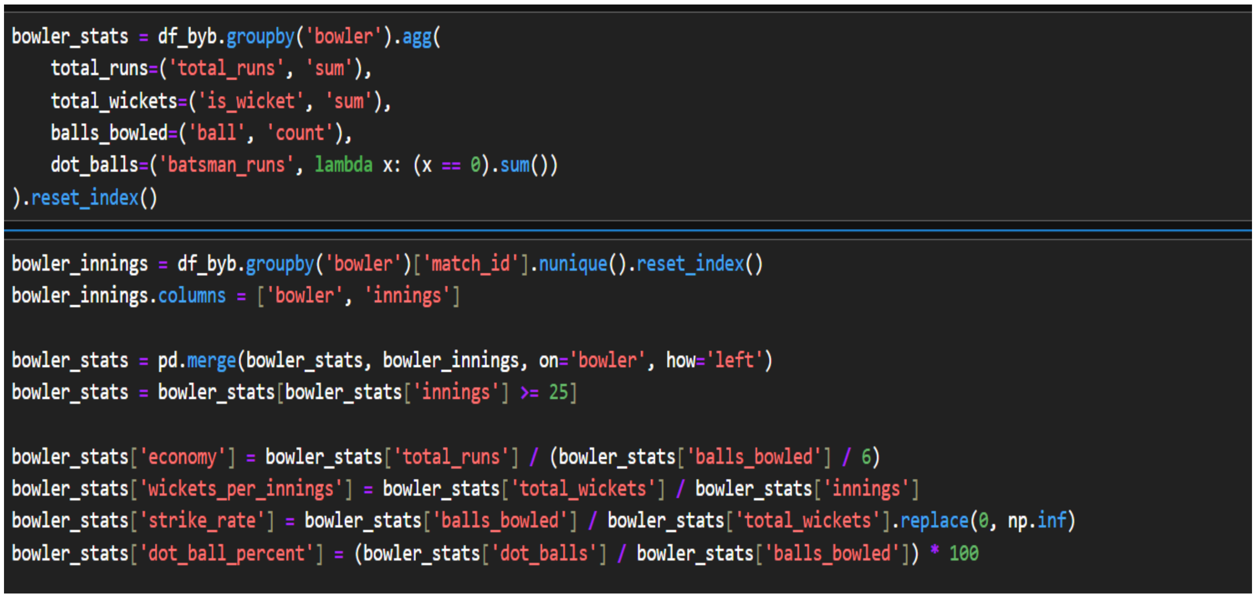
For K-means Clustering feature engineering was performed by combining batting statistics such as total runs, dismissals, balls faced, and boundaries from the ball-by-ball dataset. Then various batting metrics were engineered. Furthermore, players with at least 25 innings or more were selected to ensure for a less biased analysis.



```
player_batting = df_byb.groupby('batter').agg(
    total_runs=('batsman_runs', 'sum'),
    dismissals=('is_wicket', 'sum'),
    balls_faced=('ball', 'count'),
    boundaries=('batsman_runs', lambda x: ((x == 4) | (x == 6)).sum())
).reset_index()

batter_innings = df_byb.groupby('batter')['match_id'].nunique().reset_index()
batter_innings.columns = ['batter', 'innings']

player_batting = pd.merge(player_batting, batter_innings, on='batter', how='left')
player_batting = player_batting[player_batting['innings'] >= 25]
player_batting['strike_rate'] = (player_batting['total_runs'] / player_batting['balls_faced']) * 100
player_batting['average'] = player_batting['total_runs'] / player_batting['dismissals'].replace(0, np.inf)
player_batting.replace([np.inf, -np.inf], np.nan, inplace=True)
player_batting.dropna(subset=['average'], inplace=True)
```

*Figure 7: Feature Engineering for Batter K-Means Clustering*

For bowlers feature engineering was performed by combining bowling statistics such as total runs, is_wicket, ball, and batsman runs from the ball-by-ball dataset. Then various batting metrics were engineered. Furthermore, players with at least 25 innings or more were selected to ensure for a less biased analysis.



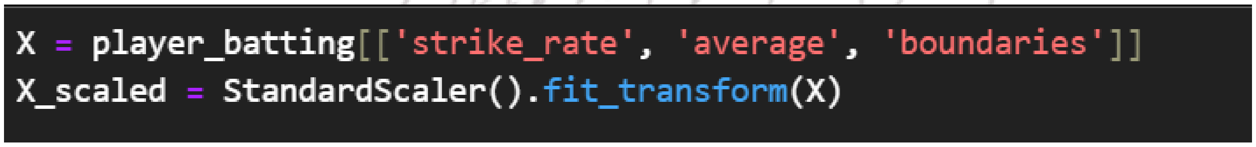
```python
bowler_stats = df_byb.groupby('bowler').agg(
    total_runs=('total_runs', 'sum'),
    total_wickets=('is_wicket', 'sum'),
    balls_bowled=('ball', 'count'),
    dot_balls=('batsman_runs', lambda x: (x == 0).sum())
).reset_index()
```

```python
bowler_innings = df_byb.groupby('bowler')['match_id'].nunique().reset_index()
bowler_innings.columns = ['bowler', 'innings']

bowler_stats = pd.merge(bowler_stats, bowler_innings, on='bowler', how='left')
bowler_stats = bowler_stats[bowler_stats['innings'] >= 25]

bowler_stats['economy'] = bowler_stats['total_runs'] / (bowler_stats['balls_bowled'] / 6)
bowler_stats['wickets_per_innings'] = bowler_stats['total_wickets'] / bowler_stats['innings']
bowler_stats['strike_rate'] = bowler_stats['balls_bowled'] / bowler_stats['total_wickets'].replace(0, np.inf)
bowler_stats['dot_ball_percent'] = (bowler_stats['dot_balls'] / bowler_stats['balls_bowled']) * 100
```

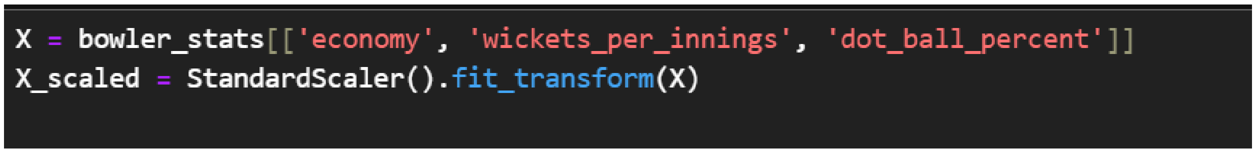
*Figure 8: Feature Engineering for Bowlers K-Mean Clustering*

## Feature Selection and Engineering for Isolation Forest

Batting metrics and bowling metrics were selected to perform anomaly detection using random forest.



```python
X = player_batting[['strike_rate', 'average', 'boundaries']]
X_scaled = StandardScaler().fit_transform(X)
```



```python
X = bowler_stats[['economy', 'wickets_per_innings', 'dot_ball_percent']]
X_scaled = StandardScaler().fit_transform(X)
```

*Figure 9: Feature Selection for Isolation Forest*

## 3.3 Technique Implementation

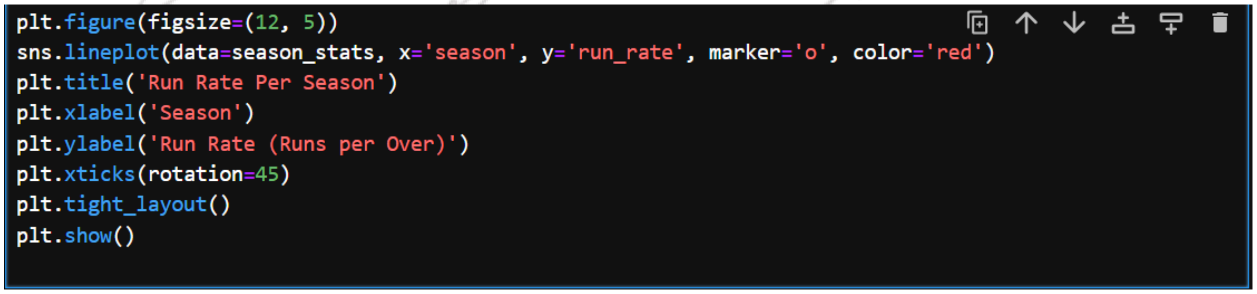
Proposed techniques were implemented using various data mining techniques to the IPL dataset. Various libraries such as pandas, numpy, matpotlib, seaborn, sklearn etc were used to Implement the proposed techniques.

### 3.3.1 Time Series Analysis

Metrics such as run rate, batting average, bowling average, batting strike rate and bowling strike rate were computed. After computing these metrics seasonally, they were visualized using line plots.

**Run Rate Time Series Analysis**

As shown in the feature selection and engineering section run rate per season was calculated and a line graph was produced. From this we can visualize changes in batting tactics over the years.

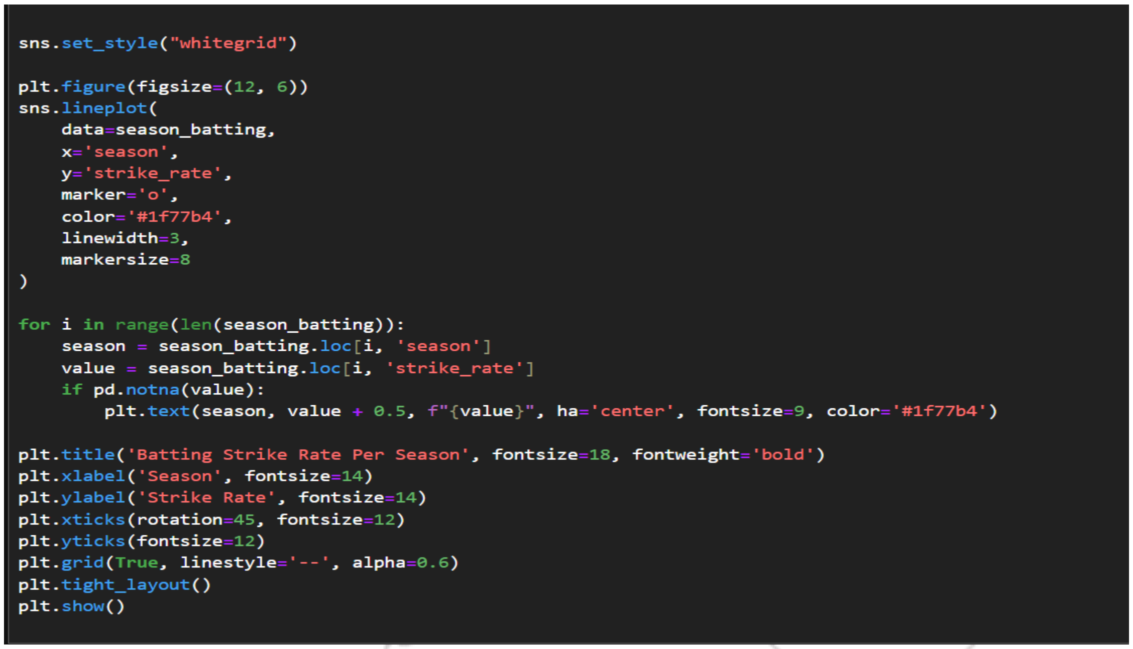


```
plt.figure(figsize=(12, 5))
sns.lineplot(data=season_stats, x='season', y='run_rate', marker='o', color='red')
plt.title('Run Rate Per Season')
plt.xlabel('Season')
plt.ylabel('Run Rate (Runs per Over)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

*Figure 10: Run Rate time Series Analysis Implementation*

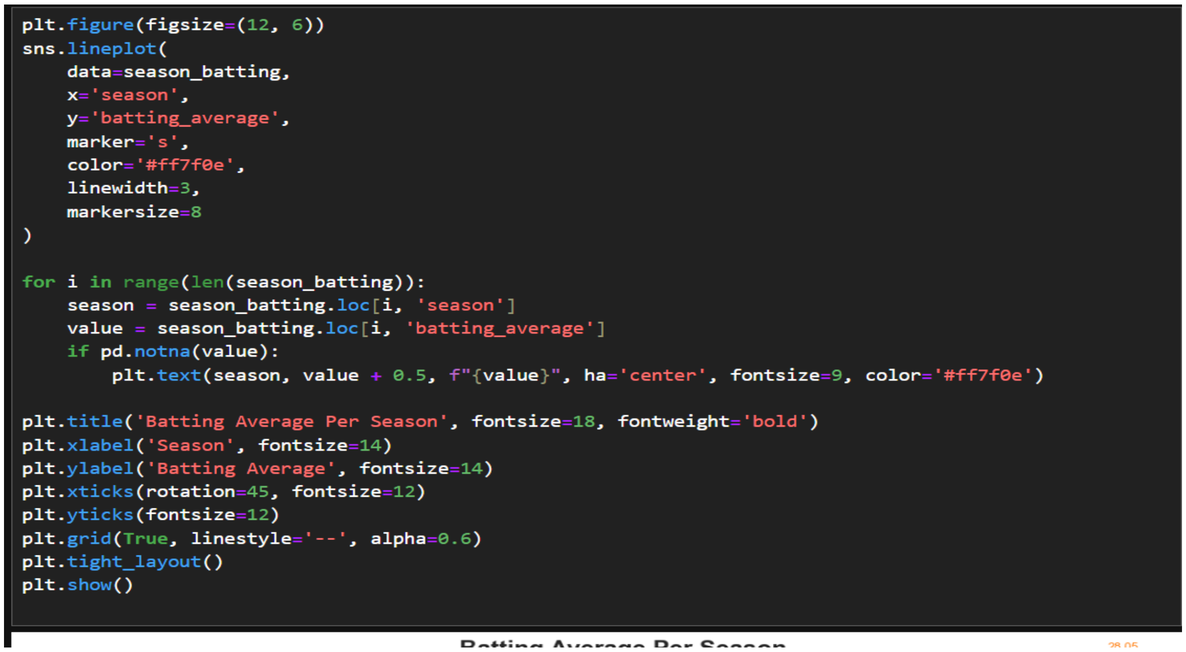
**Batting Average and Batting Strike rate Time Series Analysis**

Batting Average and strike rate were calculated using various feature engineering techniques then based on the season they were plotted into to line graph. From this we can visualize batting performances evolution over the years.



```python
sns.set_style("whitegrid")

plt.figure(figsize=(12, 6))
sns.lineplot(
    data=season_batting,
    x='season',
    y='strike_rate',
    marker='o',
    color='#1f77b4',
    linewidth=3,
    markersize=8
)

for i in range(len(season_batting)):
    season = season_batting.loc[i, 'season']
    value = season_batting.loc[i, 'strike_rate']
    if pd.notna(value):
        plt.text(season, value + 0.5, f"{value}", ha='center', fontsize=9, color='#1f77b4')

plt.title('Batting Strike Rate Per Season', fontsize=18, fontweight='bold')
plt.xlabel('Season', fontsize=14)
plt.ylabel('Strike Rate', fontsize=14)
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

*Figure 11: Batting Strike Rate Time Series Implementation*

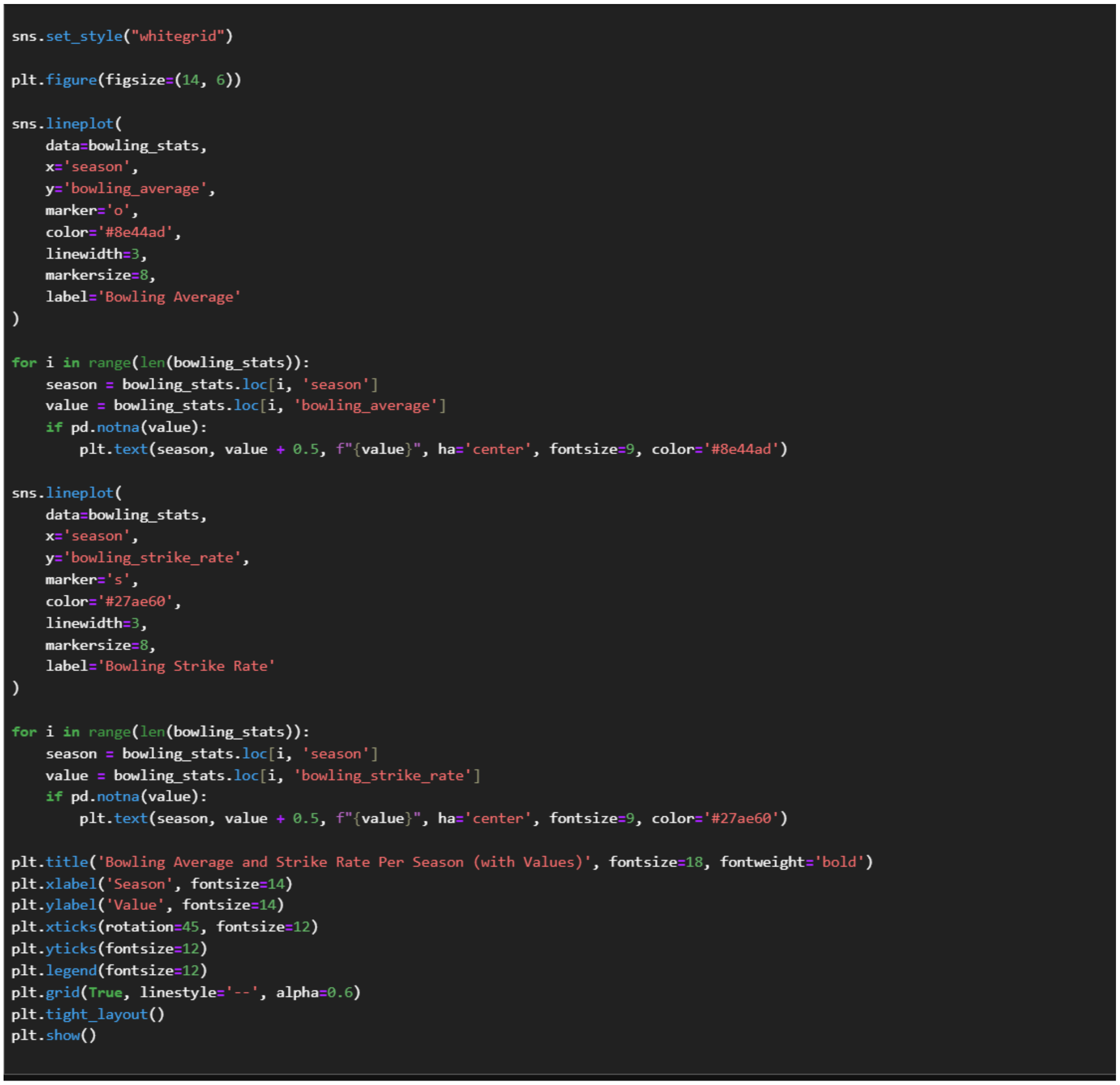


```python
plt.figure(figsize=(12, 6))
sns.lineplot(
    data=season_batting,
    x='season',
    y='batting_average',
    marker='s',
    color='#ff7f0e',
    linewidth=3,
    markersize=8
)

for i in range(len(season_batting)):
    season = season_batting.loc[i, 'season']
    value = season_batting.loc[i, 'batting_average']
    if pd.notna(value):
        plt.text(season, value + 0.5, f"{value}", ha='center', fontsize=9, color='#ff7f0e')

plt.title('Batting Average Per Season', fontsize=18, fontweight='bold')
plt.xlabel('Season', fontsize=14)
plt.ylabel('Batting Average', fontsize=14)
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

*Figure 12: Batting Average Time Series Implementation*

## Bowling Average and Bowling Strike rate Time Series Analysis

Bowling Average and strike rate were calculated using various feature engineering techniques then based on the season they were plotted into to line graph. From this we can visualize bowling performance evolution over the years.

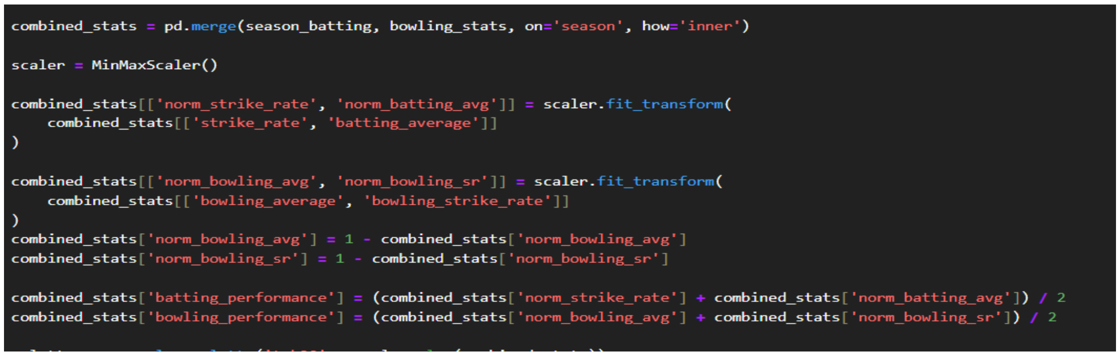


```python
sns.set_style("whitegrid")

plt.figure(figsize=(14, 6))

sns.lineplot(
    data=bowling_stats,
    x='season',
    y='bowling_average',
    marker='o',
    color='#8e44ad',
    linewidth=3,
    markersize=8,
    label='Bowling Average'
)

for i in range(len(bowling_stats)):
    season = bowling_stats.loc[i, 'season']
    value = bowling_stats.loc[i, 'bowling_average']
    if pd.notna(value):
        plt.text(season, value + 0.5, f"{value}", ha='center', fontsize=9, color='#8e44ad')

sns.lineplot(
    data=bowling_stats,
    x='season',
    y='bowling_strike_rate',
    marker='s',
    color='#27ae60',
    linewidth=3,
    markersize=8,
    label='Bowling Strike Rate'
)

for i in range(len(bowling_stats)):
    season = bowling_stats.loc[i, 'season']
    value = bowling_stats.loc[i, 'bowling_strike_rate']
    if pd.notna(value):
        plt.text(season, value + 0.5, f"{value}", ha='center', fontsize=9, color='#27ae60')

plt.title('Bowling Average and Strike Rate Per Season (with Values)', fontsize=18, fontweight='bold')
plt.xlabel('Season', fontsize=14)
plt.ylabel('Value', fontsize=14)
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

*Figure 13: Bowling Metrics Time Series Analysis Implementation*

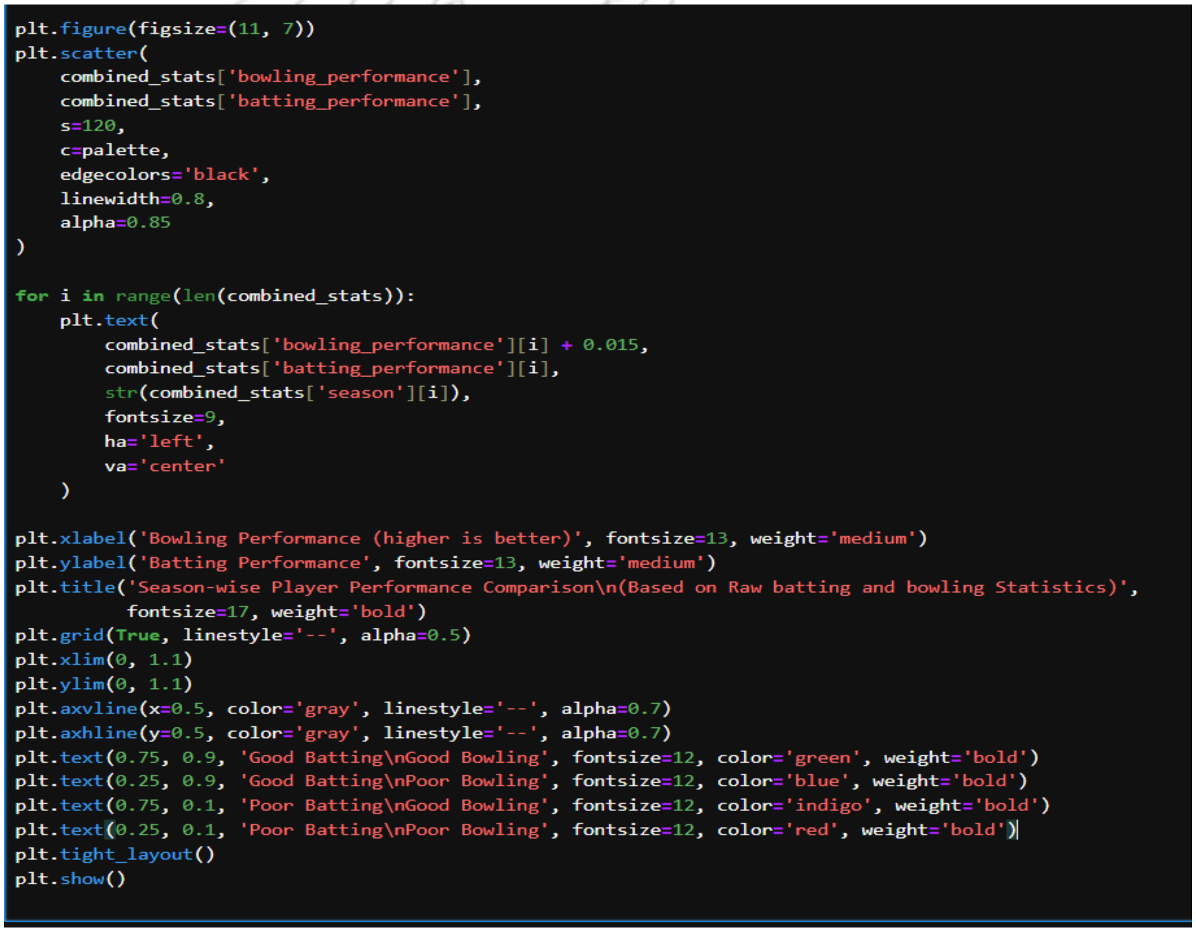
## Season-wise Player Performance Comparison

Based on batting metrics i.e., batting average and strike rate and bowling metrics i.e., bowling average and strike rate season were compared. Normalization was used to combine these metrices. Using this technique Better batting and better bowling season can be estimated.



```python
combined_stats = pd.merge(season_batting, bowling_stats, on='season', how='inner')

scaler = MinMaxScaler()

combined_stats[['norm_strike_rate', 'norm_batting_avg']] = scaler.fit_transform(
    combined_stats[['strike_rate', 'batting_average']]
)

combined_stats[['norm_bowling_avg', 'norm_bowling_sr']] = scaler.fit_transform(
    combined_stats[['bowling_average', 'bowling_strike_rate']]
)
combined_stats['norm_bowling_avg'] = 1 - combined_stats['norm_bowling_avg']
combined_stats['norm_bowling_sr'] = 1 - combined_stats['norm_bowling_sr']

combined_stats['batting_performance'] = (combined_stats['norm_strike_rate'] + combined_stats['norm_batting_avg']) / 2
combined_stats['bowling_performance'] = (combined_stats['norm_bowling_avg'] + combined_stats['norm_bowling_sr']) / 2
```

*Figure 14: Batting and Bowling Metrics Normalization*



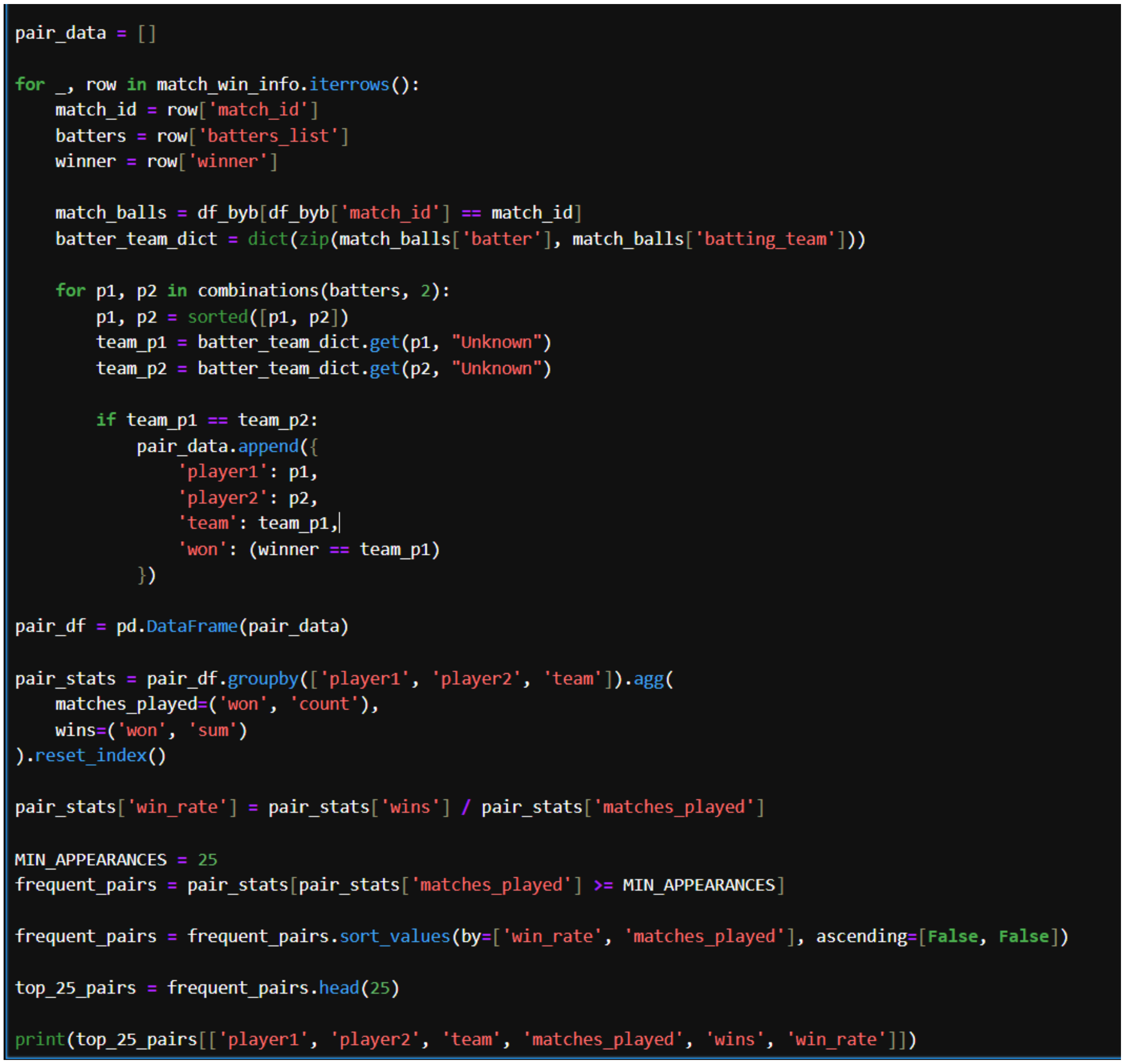
```python
plt.figure(figsize=(11, 7))
plt.scatter(
    combined_stats['bowling_performance'],
    combined_stats['batting_performance'],
    s=120,
    c=palette,
    edgecolors='black',
    linewidth=0.8,
    alpha=0.85
)

for i in range(len(combined_stats)):
    plt.text(
        combined_stats['bowling_performance'][i] + 0.015,
        combined_stats['batting_performance'][i],
        str(combined_stats['season'][i]),
        fontsize=9,
        ha='left',
        va='center'
    )

plt.xlabel('Bowling Performance (higher is better)', fontsize=13, weight='medium')
plt.ylabel('Batting Performance', fontsize=13, weight='medium')
plt.title('Season-wise Player Performance Comparison\n(Based on Raw batting and bowling Statistics)',
          fontsize=17, weight='bold')
plt.grid(True, linestyle='--', alpha=0.5)
plt.xlim(0, 1.1)
plt.ylim(0, 1.1)
plt.axvline(x=0.5, color='gray', linestyle='--', alpha=0.7)
plt.axhline(y=0.5, color='gray', linestyle='--', alpha=0.7)
plt.text(0.75, 0.9, 'Good Batting\nGood Bowling', fontsize=12, color='green', weight='bold')
plt.text(0.25, 0.9, 'Good Batting\nPoor Bowling', fontsize=12, color='blue', weight='bold')
plt.text(0.75, 0.1, 'Poor Batting\nGood Bowling', fontsize=12, color='indigo', weight='bold')
plt.text(0.25, 0.1, 'Poor Batting\nPoor Bowling', fontsize=12, color='red', weight='bold')
plt.tight_layout()
plt.show()
```

*Figure 15: Season-wise Player Performance Comparison Time Series Analysis*

## 3.3.2 Association Analysis of Player Pairs

### Top 25 Batting Pairs

Using this code pairs of batters from the same team are selected. After selection, they are compared against how often their team wins. After this, the top 25 pairs with the highest win rate and 25 or more matches are selected. Using this method the best-performing batting combinations can be identified.

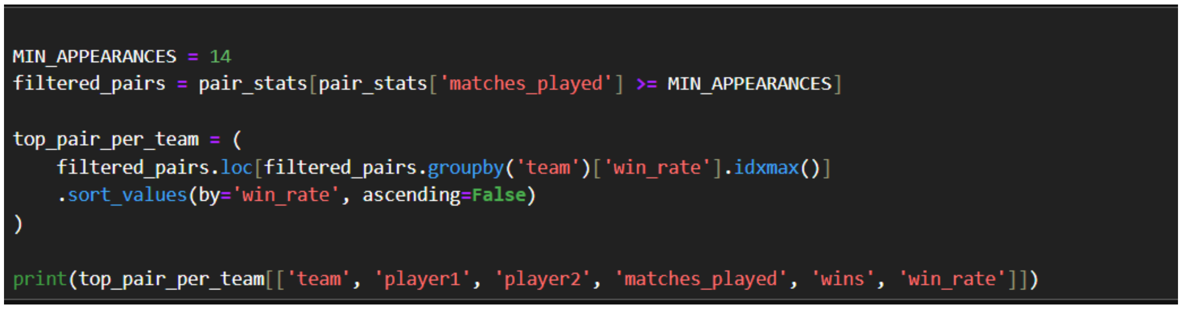


```
pair_data = []

for _, row in match_win_info.iterrows():
    match_id = row['match_id']
    batters = row['batters_list']
    winner = row['winner']

    match_balls = df_byb[df_byb['match_id'] == match_id]
    batter_team_dict = dict(zip(match_balls['batter'], match_balls['batting_team']))

    for p1, p2 in combinations(batters, 2):
        p1, p2 = sorted([p1, p2])
        team_p1 = batter_team_dict.get(p1, "Unknown")
        team_p2 = batter_team_dict.get(p2, "Unknown")

        if team_p1 == team_p2:
            pair_data.append({
                'player1': p1,
                'player2': p2,
                'team': team_p1,
                'won': (winner == team_p1)
            })

pair_df = pd.DataFrame(pair_data)

pair_stats = pair_df.groupby(['player1', 'player2', 'team']).agg(
    matches_played=('won', 'count'),
    wins=('won', 'sum')
).reset_index()

pair_stats['win_rate'] = pair_stats['wins'] / pair_stats['matches_played']

MIN_APPEARANCES = 25
frequent_pairs = pair_stats[pair_stats['matches_played'] >= MIN_APPEARANCES]

frequent_pairs = frequent_pairs.sort_values(by=['win_rate', 'matches_played'], ascending=[False, False])

top_25_pairs = frequent_pairs.head(25)

print(top_25_pairs[['player1', 'player2', 'team', 'matches_played', 'wins', 'win_rate']])
```

*Figure 16: Association Analysis of Top 25 Batting Pairs*

Best Batting Players per team

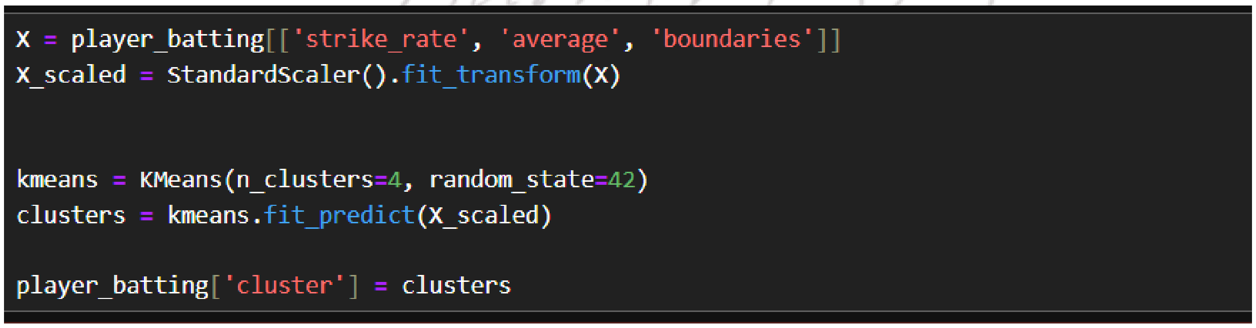
Using this code pairs of batters from the same team are selected. After selection, they are compared against how often their team wins. After this, every time the batting pair with the top win rate is selected. Using this technique best batting pairs from each team can be identified.



```
MIN_APPEARANCES = 14
filtered_pairs = pair_stats[pair_stats['matches_played'] >= MIN_APPEARANCES]

top_pair_per_team = (
    filtered_pairs.loc[filtered_pairs.groupby('team')['win_rate'].idxmax()]
    .sort_values(by='win_rate', ascending=False)
)

print(top_pair_per_team[['team', 'player1', 'player2', 'matches_played', 'wins', 'win_rate']])
```

## 3.3.3 K-Means Clustering

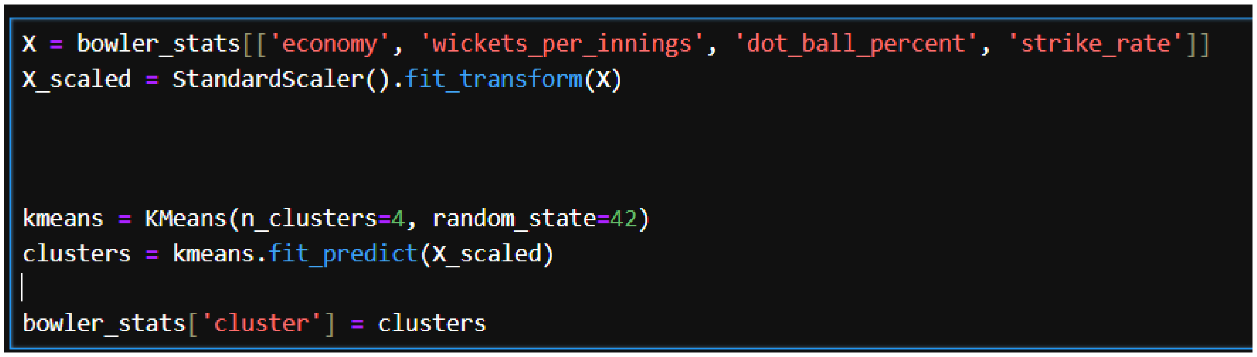
Using batting metrices such as strike rate, average and boundaries K-Means clustering is applied to find clusters of batter. This method can help identify different categories of batters.



```
X = player_batting[['strike_rate', 'average', 'boundaries']]
X_scaled = StandardScaler().fit_transform(X)


kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

player_batting['cluster'] = clusters
```

*Figure 17: Batters K-Means Clustering Implementation*

Using bowling metrices such as strike rate, economy, wicket per inning and dot ball percentage K-Means clustering is applied to find clusters of batter. This method can help identify different categories of bowlers.

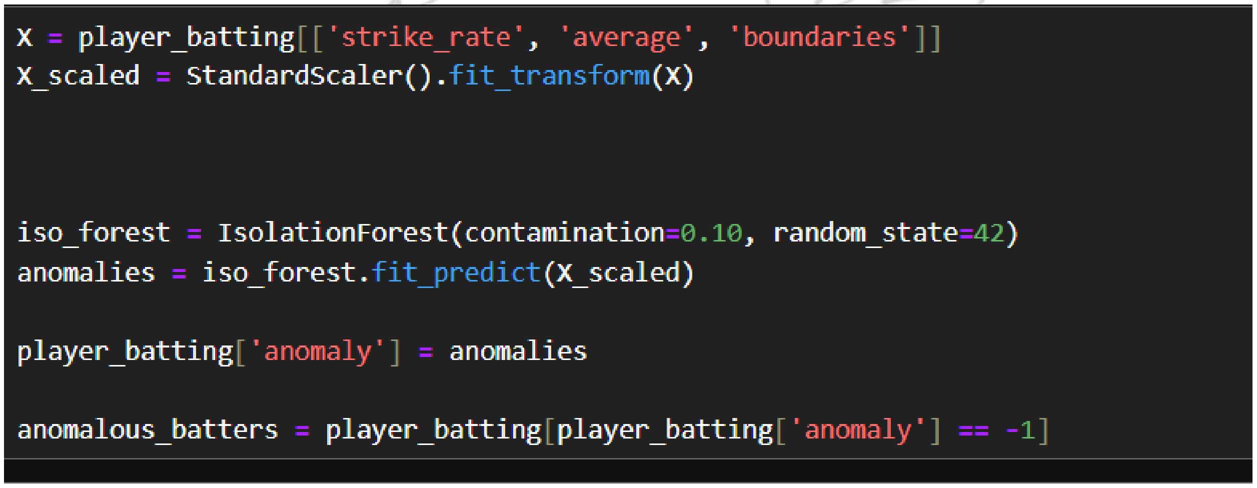


```
X = bowler_stats[['economy', 'wickets_per_innings', 'dot_ball_percent', 'strike_rate']]
X_scaled = StandardScaler().fit_transform(X)



kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

bowler_stats['cluster'] = clusters
```

*Figure 18: Bowlers K-Means Clustering Implementation*

### 3.3.4 Isolation Forest

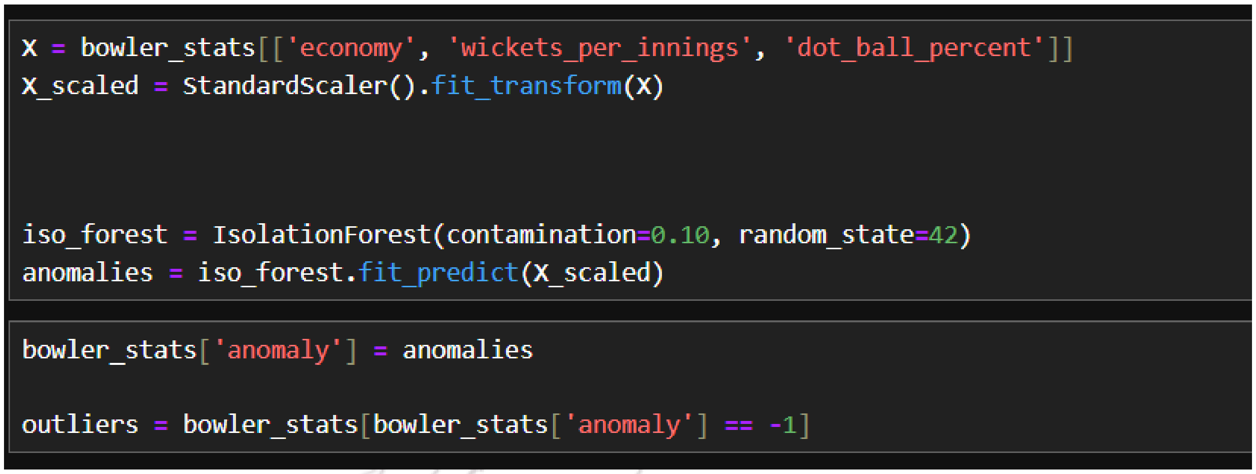
Using batting metrices such as strike rate, average and boundaries isolation Forest is applied to find anomalies in batters. This method can help identify various underachievers and overachievers.



```
X = player_batting[['strike_rate', 'average', 'boundaries']]
X_scaled = StandardScaler().fit_transform(X)



iso_forest = IsolationForest(contamination=0.10, random_state=42)
anomalies = iso_forest.fit_predict(X_scaled)

player_batting['anomaly'] = anomalies

anomalous_batters = player_batting[player_batting['anomaly'] == -1]
```

*Figure 19: Batting Isolation Forest Implementation*

Using bowling metrices such as economy, wicket per inning and dot ball percentage isolation Forest is applied to find anomalies in bowlers. This method can help identify various underachievers and overachievers.



```
X = bowler_stats[['economy', 'wickets_per_innings', 'dot_ball_percent']]
X_scaled = StandardScaler().fit_transform(X)



iso_forest = IsolationForest(contamination=0.10, random_state=42)
anomalies = iso_forest.fit_predict(X_scaled)
```

```
bowler_stats['anomaly'] = anomalies

outliers = bowler_stats[bowler_stats['anomaly'] == -1]
```

*Figure 20: Bowlers Isolation Forest Implementation*

[illegible]

# 4. Results and Interpretation

After implementing various data mining techniques to the IPL dataset various results and findings have been gathered. These results focus on league trends, player performances, clustering, pair combination, and rare players.

## 4.1 Time Series Analysis Result And Interpretation

### Run Rate Time series Analysis

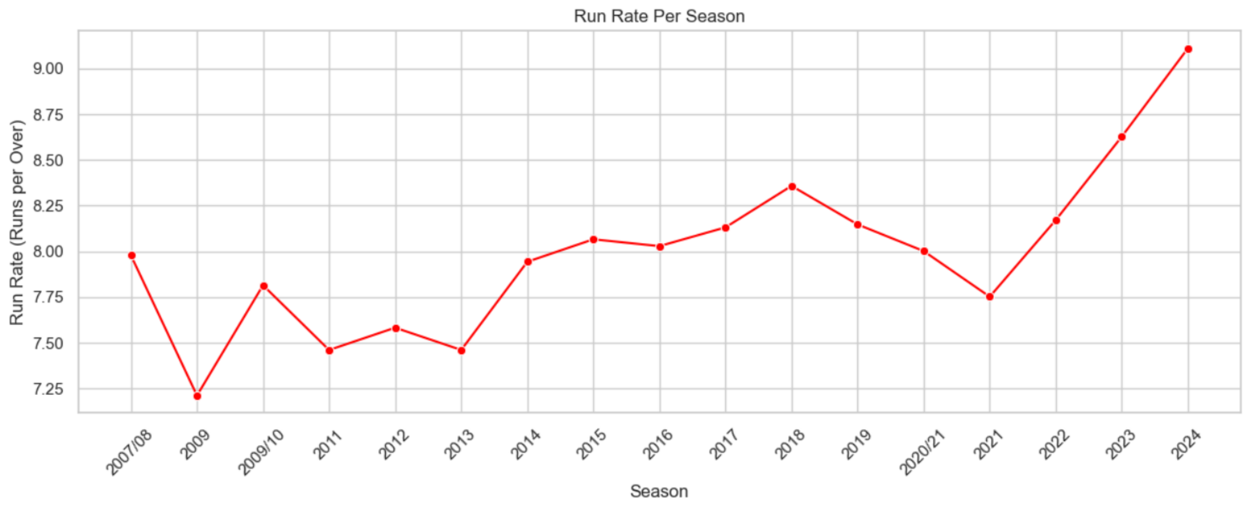


*Figure 21: Run Rate Time series Analysis*

**Interpretation**: The scoring rate starts off strong with 8.0 in the first year of IPL it takes a massive deep in 2009 which may be because the IPL was hosted in South Africa a bowling-friendly country. Slight fluctuations keep occurring from 2010 to 2013. After that we can see a steep upward trend in scoring rate till 2018. From 2019 to 2021 there was a dip in scoring rates which may be due to COVID-19 After 2021 there was a sharp rise in the scoring rate which indicates a shift towards a more aggressive batting style and changes in the game that favor the batsman.

## Batting Average Time Series Analysis

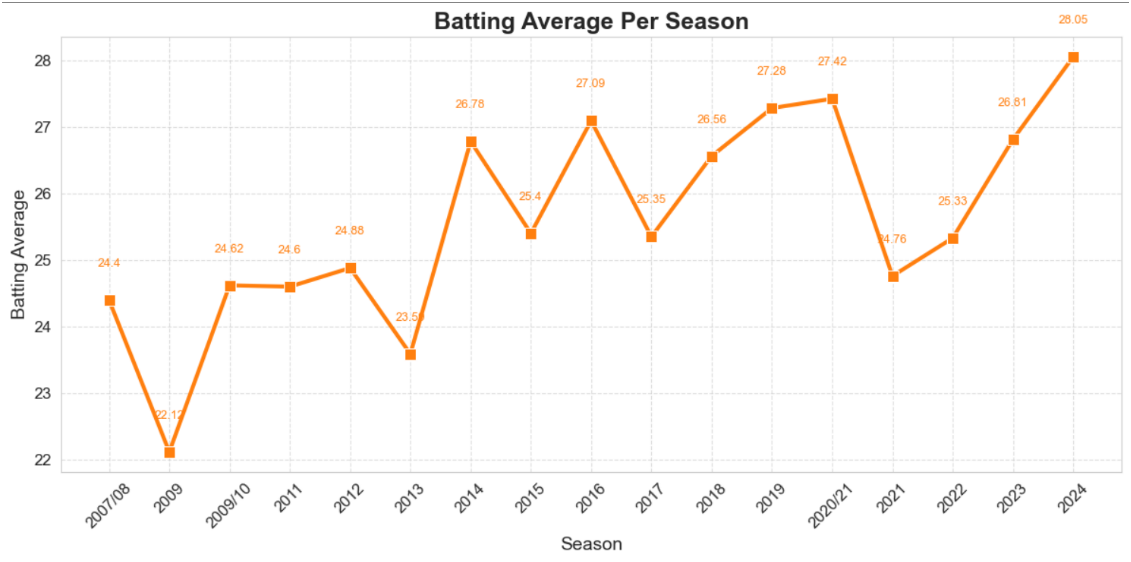


*Figure 22: Batting Average Time Series Analysis*

**Interpretation:** The batting average starts off strong at around 24.4 in the first season, then there is a massive dip in batting average which may be the result of the IPL being hosted in South Africa, we can see a steady graph until the dip occurring in 2013. From 2014 to 2019 there are constant fluctuations in batting averages. After 2019 we can see a downward trend in batting averages which may be the result of COVID-19. After 2021 the batting averages shot up showcasing strong seasons for batting post 2021.

## Batting Strike Rate Time Series Analysis

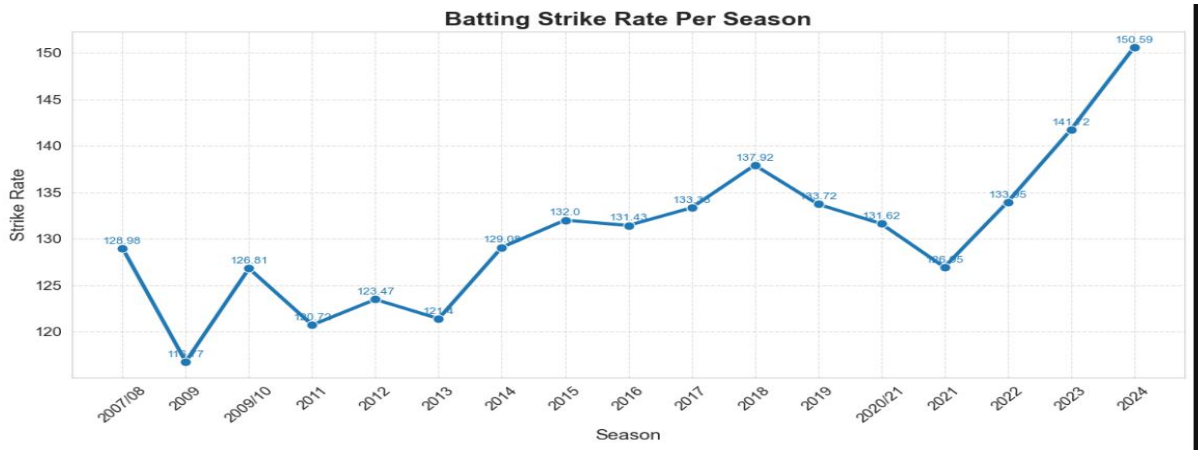


*Figure 23: Batting Strike Rate Time Series Analysis*

**Interpretation**: The batting Strike rate follows a similar trend to the run rate, both starting strong and dropping off in 2009 and during the COVID-19 periods. Since 2021 the batting strike rate has seen a massive increase. This showcases changes in batting strategies and might also possibly showcase an influx of new talents who play their game fearlessly. IPL in India is also faster scoring when compared to being hosted in other nations.

## Batting Average and Strike Rate Time Series Analysis

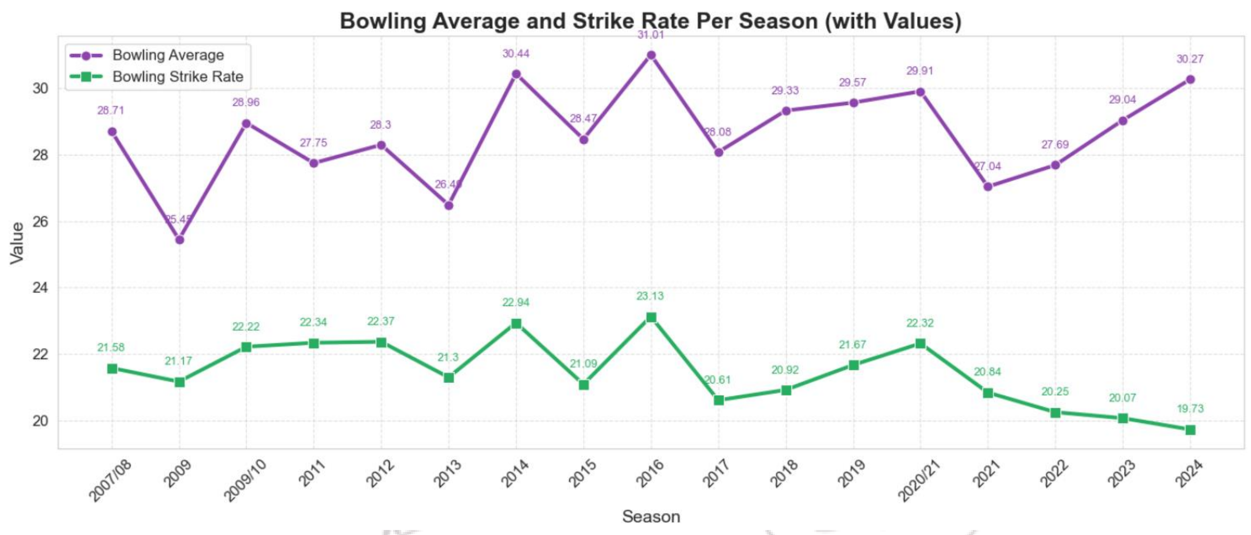


*Figure 24: Batting Average and Strike Rate Time Series Analysis*

**Interpretation**: Bowling Average and Strike rate started at 28.7 and 21.55 in 2008 respectively after that there was a massive dip in bowling average showcasing a poor season for the batter as the dip is not that large in bowling strike, it can interpreted that it was harder to score runs in 2009  after some fluctuations bowling average and strike rate seem to follow similar trends until 2021. After 2021 the bowling average shot up and the bowling strike rate is going down. This might be the result of a more aggressive approach among batters and a shift in fast-scoring strategies.

## Season-wise Player Performance Comparison

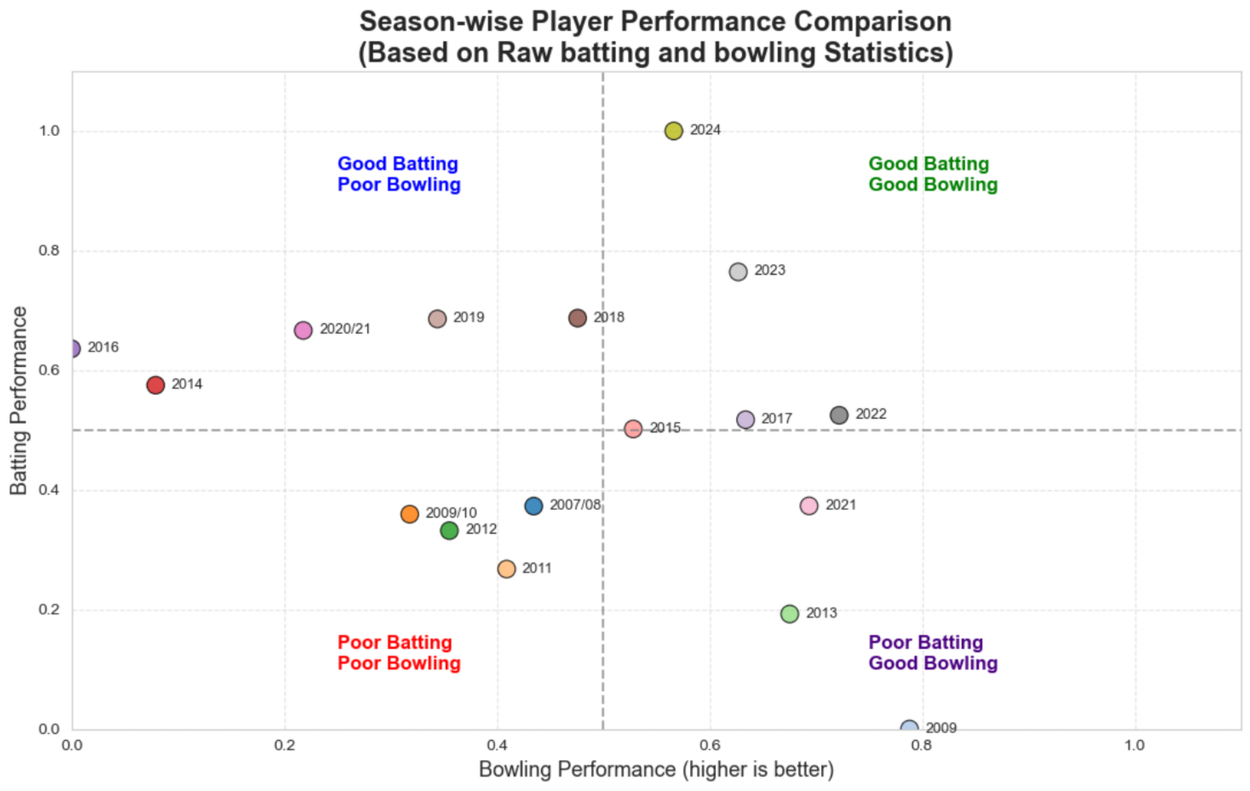


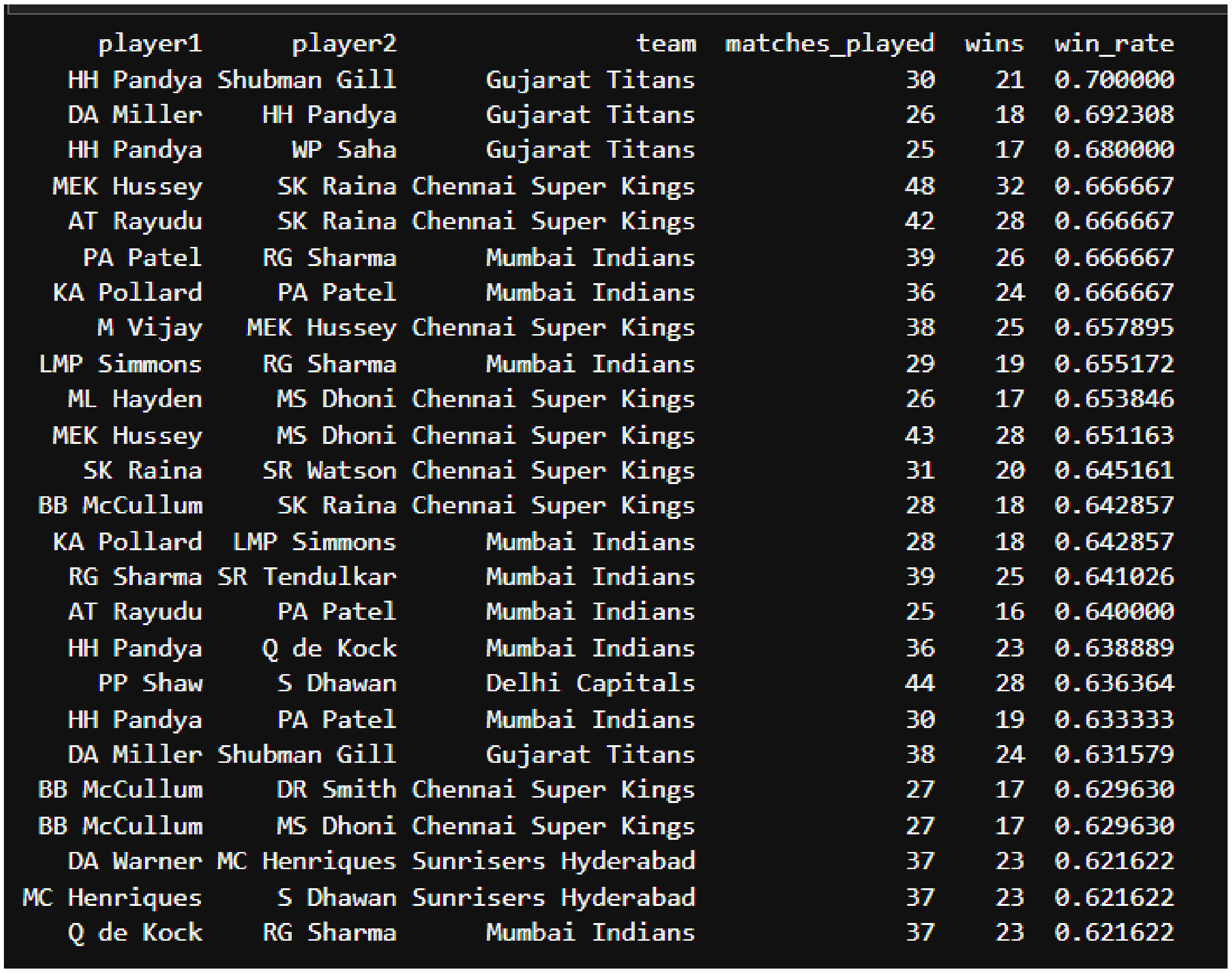
*Figure 25: Season-wise Player Performance Comparison*

This plot helps us identify the season in which batting or bowling has dominated. Normalized batting and bowling performance metrics are divided into various quadrants. By interpreting this graph, we can identify what kind of season a certain season was.

From the Time Series Analysis of historical IPL data, we can see that there are significant dips in 2009 possibly due to South Africa hosting it. This analysis also showcases the impact of COVID-19 on cricket as during those years batting performances went down. Since 2021 batters have been dominating bowlers, and batting strike rate and batting average have been going up. Despite batting metrics being on the rise the bowling strike rate has also been falling which might indicate a change in batting strategies with more focus on an aggressive style.

## 4.2 Association Analysis of Player Pairs Results And Interpretation

Top 25 Batting Pairs



```
      player1      player2                team  matches_played  wins  win_rate
    HH Pandya Shubman Gill      Gujarat Titans              30    21  0.700000
    DA Miller     HH Pandya      Gujarat Titans              26    18  0.692308
    HH Pandya       WP Saha      Gujarat Titans              25    17  0.680000
   MEK Hussey      SK Raina Chennai Super Kings              48    32  0.666667
    AT Rayudu      SK Raina Chennai Super Kings              42    28  0.666667
     PA Patel     RG Sharma      Mumbai Indians              39    26  0.666667
   KA Pollard      PA Patel      Mumbai Indians              36    24  0.666667
      M Vijay    MEK Hussey Chennai Super Kings              38    25  0.657895
  LMP Simmons     RG Sharma      Mumbai Indians              29    19  0.655172
    ML Hayden      MS Dhoni Chennai Super Kings              26    17  0.653846
   MEK Hussey      MS Dhoni Chennai Super Kings              43    28  0.651163
     SK Raina     SR Watson Chennai Super Kings              31    20  0.645161
  BB McCullum      SK Raina Chennai Super Kings              28    18  0.642857
   KA Pollard   LMP Simmons      Mumbai Indians              28    18  0.642857
    RG Sharma SR Tendulkar      Mumbai Indians              39    25  0.641026
    AT Rayudu      PA Patel      Mumbai Indians              25    16  0.640000
    HH Pandya     Q de Kock      Mumbai Indians              36    23  0.638889
      PP Shaw      S Dhawan      Delhi Capitals              44    28  0.636364
    HH Pandya      PA Patel      Mumbai Indians              30    19  0.633333
    DA Miller Shubman Gill      Gujarat Titans              38    24  0.631579
  BB McCullum      DR Smith Chennai Super Kings              27    17  0.629630
  BB McCullum      MS Dhoni Chennai Super Kings              27    17  0.629630
    DA Warner MC Henriques Sunrisers Hyderabad              37    23  0.621622
 MC Henriques      S Dhawan Sunrisers Hyderabad              37    23  0.621622
    Q de Kock     RG Sharma      Mumbai Indians              37    23  0.621622
```

**Interpretation**: This list shows the best batting pairs of IPL history based on win rate. This list is dominated by Chennai Super Kings and Mumbai Indians. This indicates that these two teams have a history of having strong batting lineups and winning very often. A news team Gujrat Titans has also appeared on this list multiple times. This showcases that despite being a new team Gujrat Titans have a very well-built team. Several pairs such as MEK Hussey and SK Raina from Chennai Super Kings have played a lot of matches together. This showcases that the Chennai Super Kings were heavily reliant on them when they were playing together.

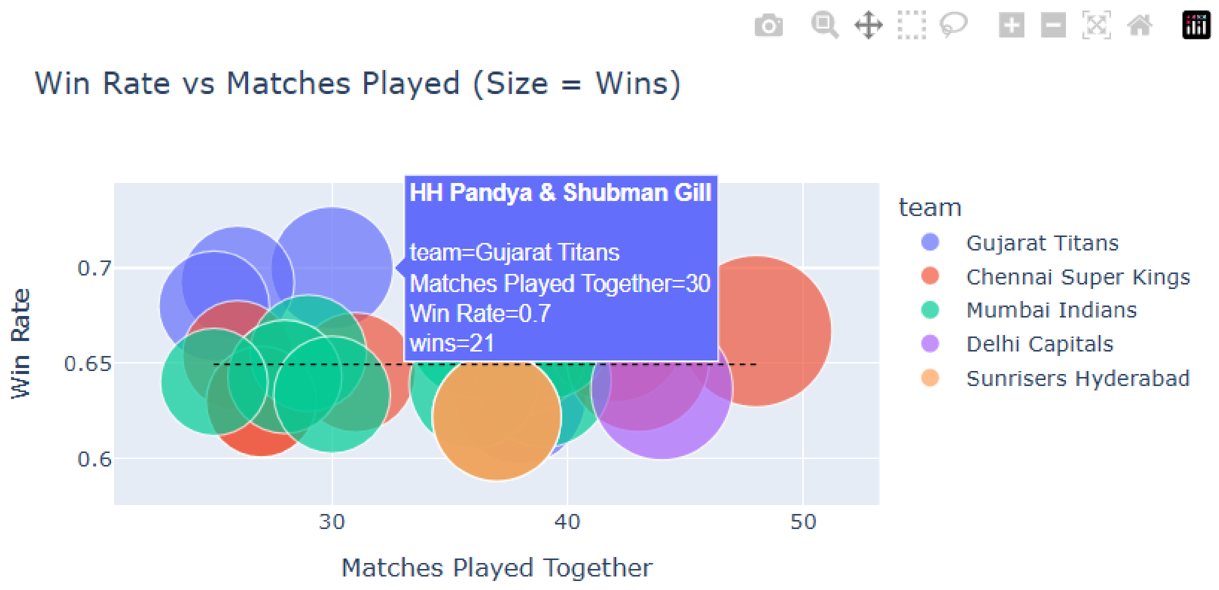
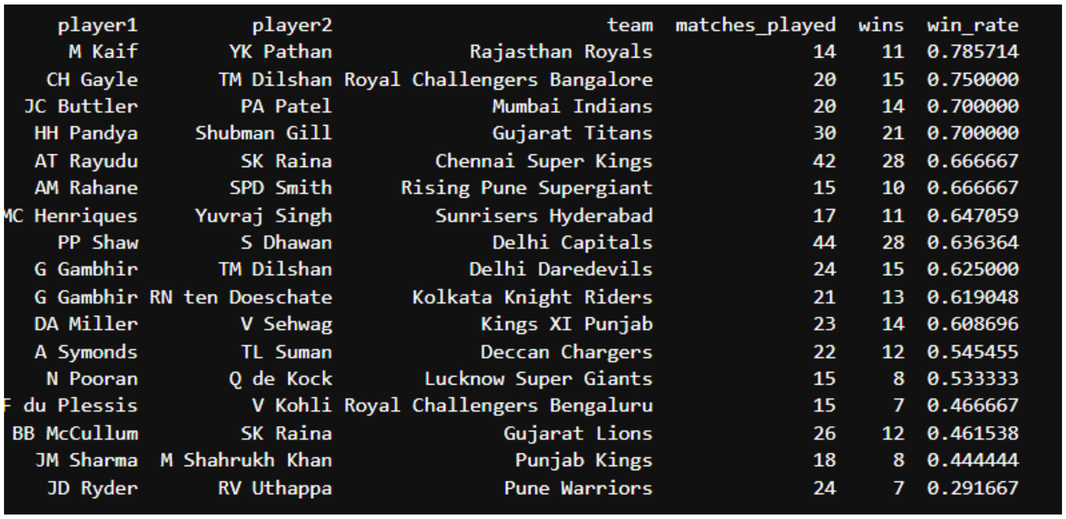


*Figure 26: Top 25 Batting Pair Bubble Plot*

The data showcased above is represented in a bubble plot. The following interpretations can also be obtained from this plot. Upon hovering over these bubbles, more details regarding these pairs are visible.

## Best Pair of Each Teams



```
     player1          player2                        team  matches_played  wins  win_rate
      M Kaif        YK Pathan            Rajasthan Royals              14    11  0.785714
    CH Gayle       TM Dilshan Royal Challengers Bangalore              20    15  0.750000
  JC Buttler         PA Patel              Mumbai Indians              20    14  0.700000
   HH Pandya     Shubman Gill              Gujarat Titans              30    21  0.700000
   AT Rayudu         SK Raina         Chennai Super Kings              42    28  0.666667
   AM Rahane        SPD Smith      Rising Pune Supergiant              15    10  0.666667
MC Henriques     Yuvraj Singh         Sunrisers Hyderabad              17    11  0.647059
     PP Shaw         S Dhawan              Delhi Capitals              44    28  0.636364
   G Gambhir       TM Dilshan            Delhi Daredevils              24    15  0.625000
   G Gambhir RN ten Doeschate       Kolkata Knight Riders              21    13  0.619048
   DA Miller         V Sehwag             Kings XI Punjab              23    14  0.608696
   A Symonds          TL Suman             Deccan Chargers              22    12  0.545455
    N Pooran        Q de Kock        Lucknow Super Giants              15     8  0.533333
F du Plessis          V Kohli Royal Challengers Bengaluru              15     7  0.466667
 BB McCullum         SK Raina                Gujarat Lions              26    12  0.461538
   JM Sharma  M Shahrukh Khan                Punjab Kings              18     8  0.444444
    JD Ryder       RV Uthappa               Pune Warriors              24     7  0.291667
```

**Interpretation**: The best pair for each team shows a variety of successful combinations, this indicates that different players have relied on different players for success. Some teams have a high win rate while some teams have a very low win rate which showcases how building a strong pair or partnership is crucial in cricket. This data also showcases the importance of having a strong and reliable batting pair in cricket.

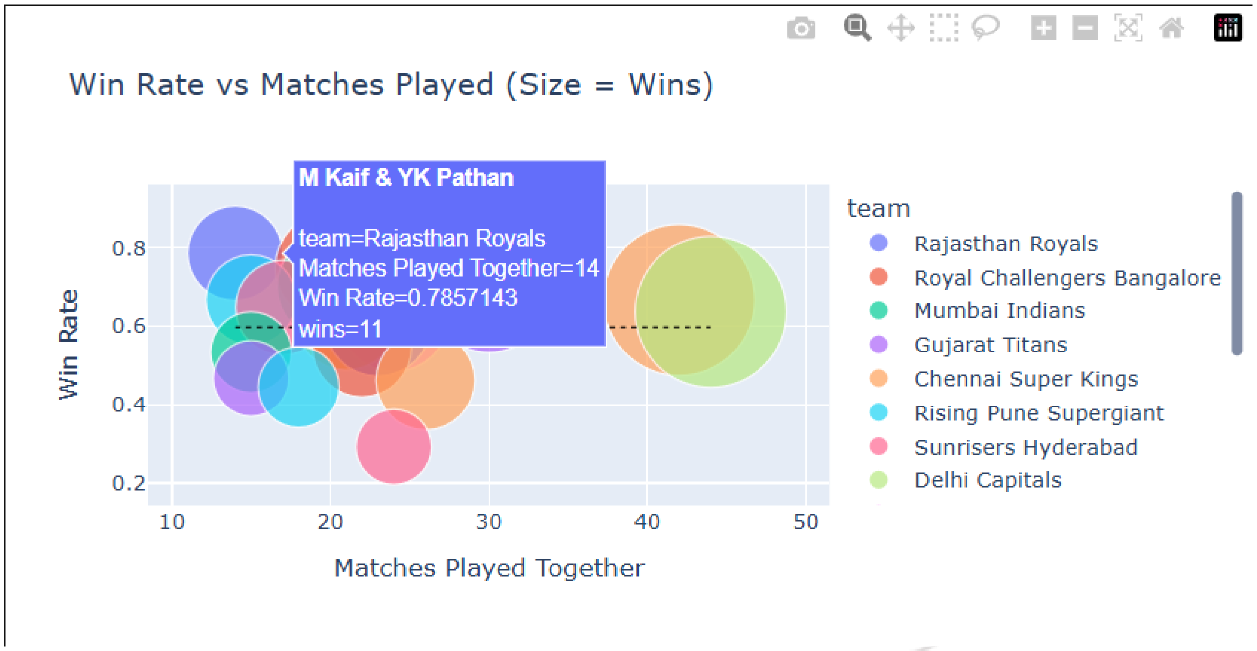


*Figure 27:Best Pair of Each Teams Bubble Plot*

The data showcased above is represented in a bubble plot. The following interpretations can also be obtained from this plot. Upon hovering over these bubbles, more details regarding these pairs are visible.

## 4.3 K-Means Clustering Results and Interpretation

### Batters Clustering

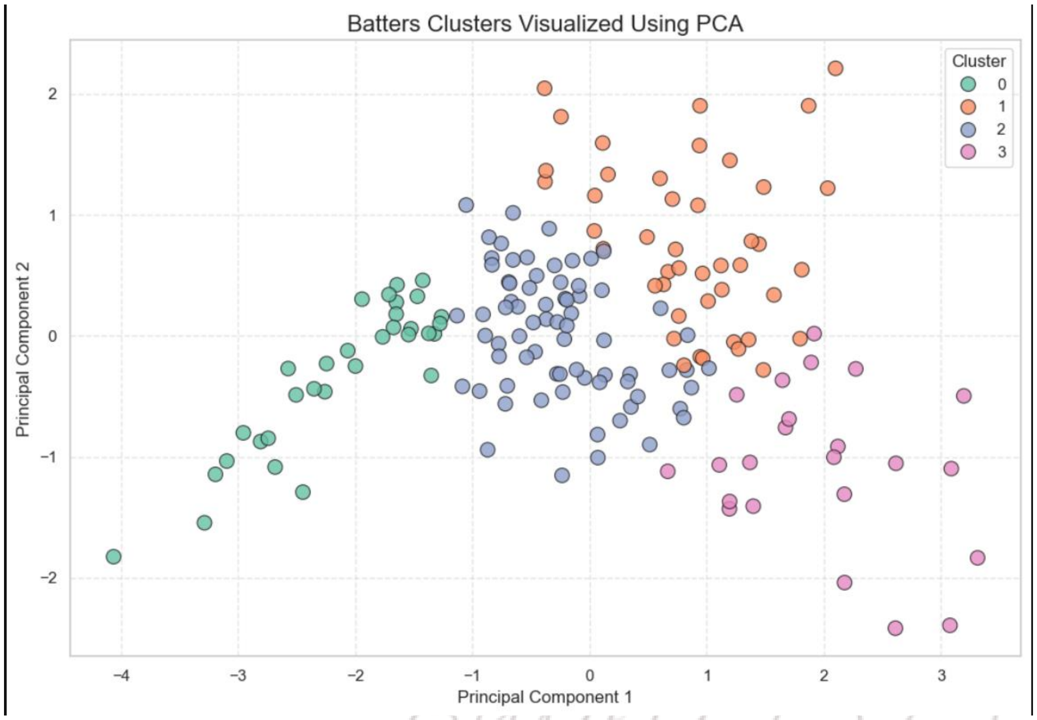


*Figure 28: K-Means Clustering for Batters*

PCA has been Applied to this clustering. A scatter plot has been visualized using PCA. In the visualization we can see groups of batters based on their principal components. Four clusters are visible.

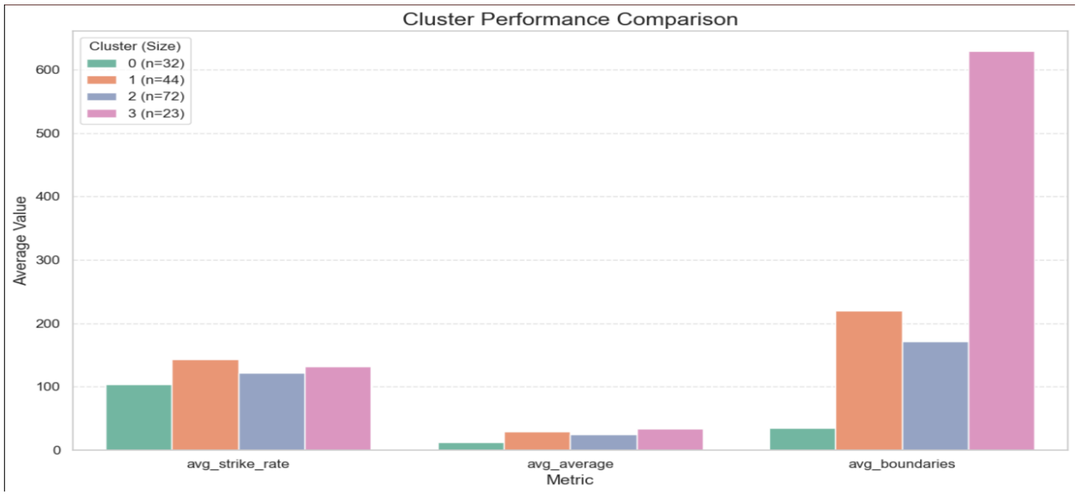
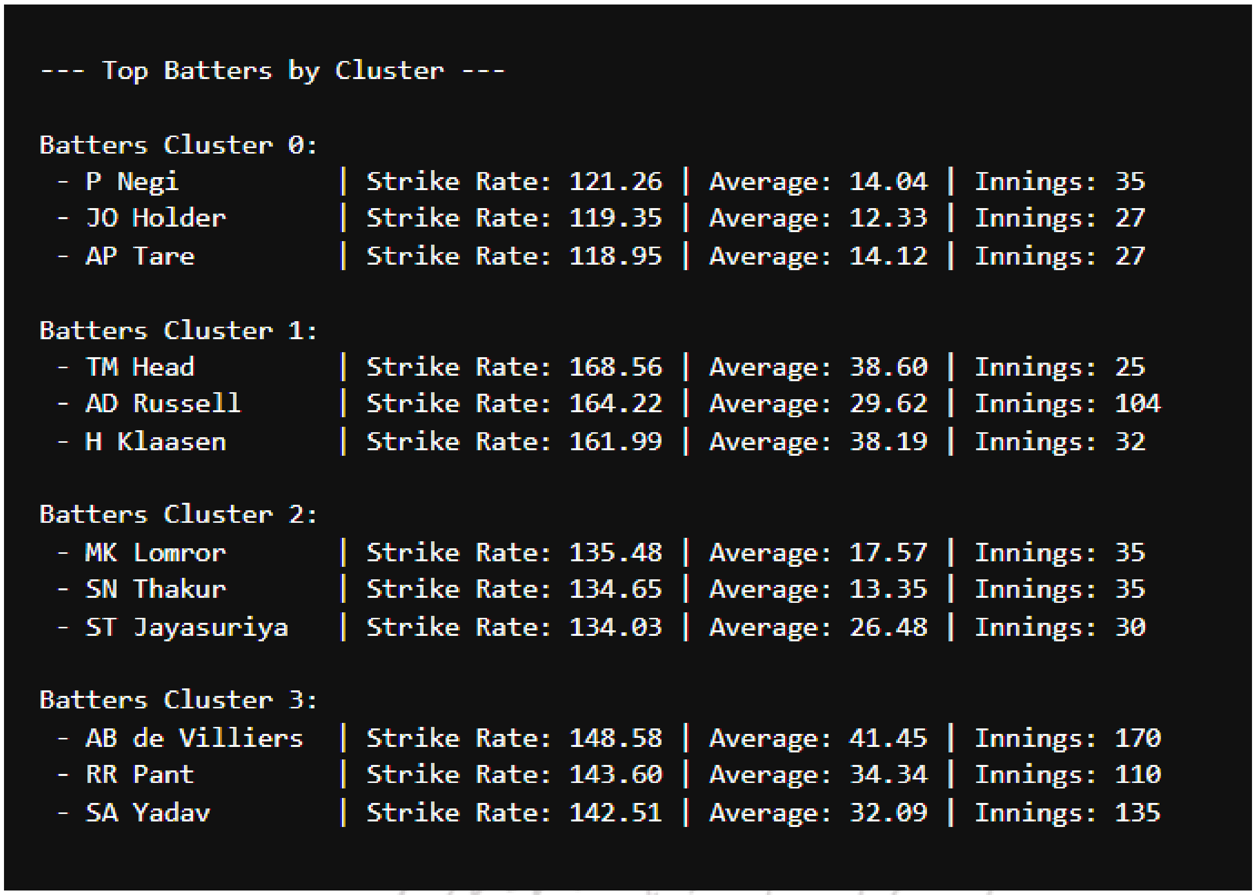


*Figure 29: Batters Cluster Performance Comparison*



```
--- Top Batters by Cluster ---

Batters Cluster 0:
  - P Negi          | Strike Rate: 121.26 | Average: 14.04 | Innings: 35
  - JO Holder       | Strike Rate: 119.35 | Average: 12.33 | Innings: 27
  - AP Tare         | Strike Rate: 118.95 | Average: 14.12 | Innings: 27

Batters Cluster 1:
  - TM Head         | Strike Rate: 168.56 | Average: 38.60 | Innings: 25
  - AD Russell      | Strike Rate: 164.22 | Average: 29.62 | Innings: 104
  - H Klaasen       | Strike Rate: 161.99 | Average: 38.19 | Innings: 32

Batters Cluster 2:
  - MK Lomror       | Strike Rate: 135.48 | Average: 17.57 | Innings: 35
  - SN Thakur       | Strike Rate: 134.65 | Average: 13.35 | Innings: 35
  - ST Jayasuriya   | Strike Rate: 134.03 | Average: 26.48 | Innings: 30

Batters Cluster 3:
  - AB de Villiers  | Strike Rate: 148.58 | Average: 41.45 | Innings: 170
  - RR Pant         | Strike Rate: 143.60 | Average: 34.34 | Innings: 110
  - SA Yadav        | Strike Rate: 142.51 | Average: 32.09 | Innings: 135
```

**Interpretation**: K-means clustering has divided into 4 different groups. Many anomalies and outliers are also visible in this cluster. Each player in a cluster has different characteristics:

**Cluster 0** – These players have a low strike rate and low average with a relatively high number of innings. This suggests that either they are underperforming lower-order batters or batting is not their primary skill.

**Cluster 1** – These players have high strike rates and decent averages; this indicates they are aggressive batsmen who are capable of scoring big runs constantly. Players of this cluster being middle-order or lower-middle-order batsmen could also be a reason that they have high averages.

**Cluster 2** – These players have decent strike rates and low averages, this indicates either they are lower-order batters, or they have not been able to score at a consistent rate. This player might be all-rounders and upcoming batters.

**Cluster 3** – These players have the highest averages and second highest strike rate showcasing they are versatile players who mostly bat at the top-order. These batters also stay highly consistent and aggressive. They are likely to be the best performers of a team.

## Bowlers Clustering

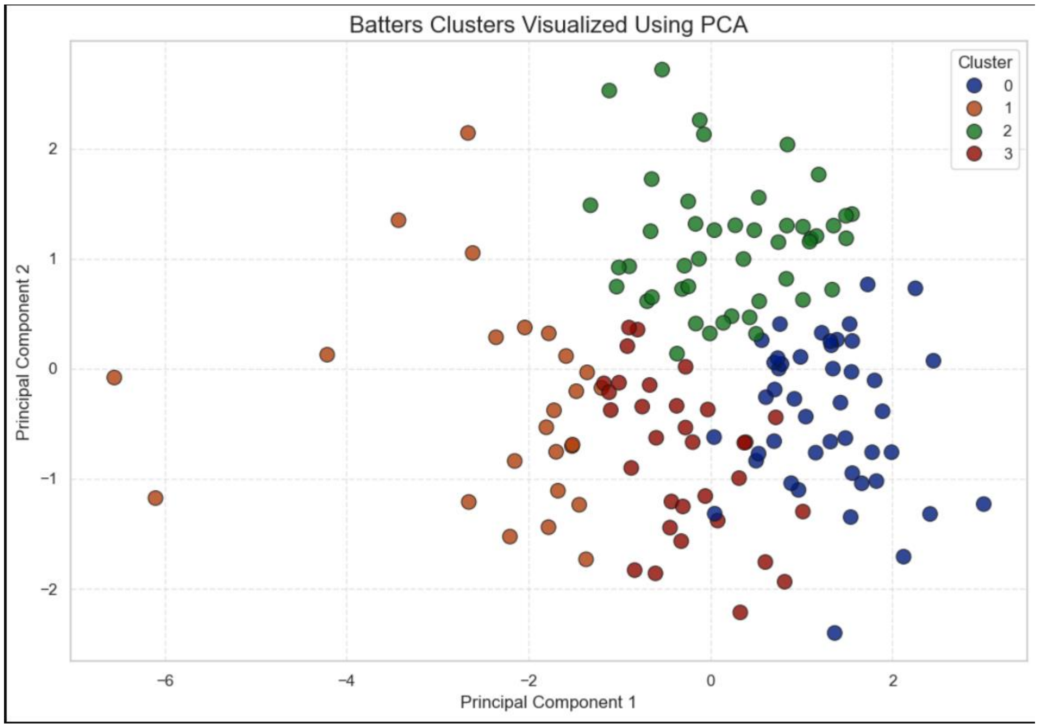


*Figure 30: Bowlers Clustering Using PCA*

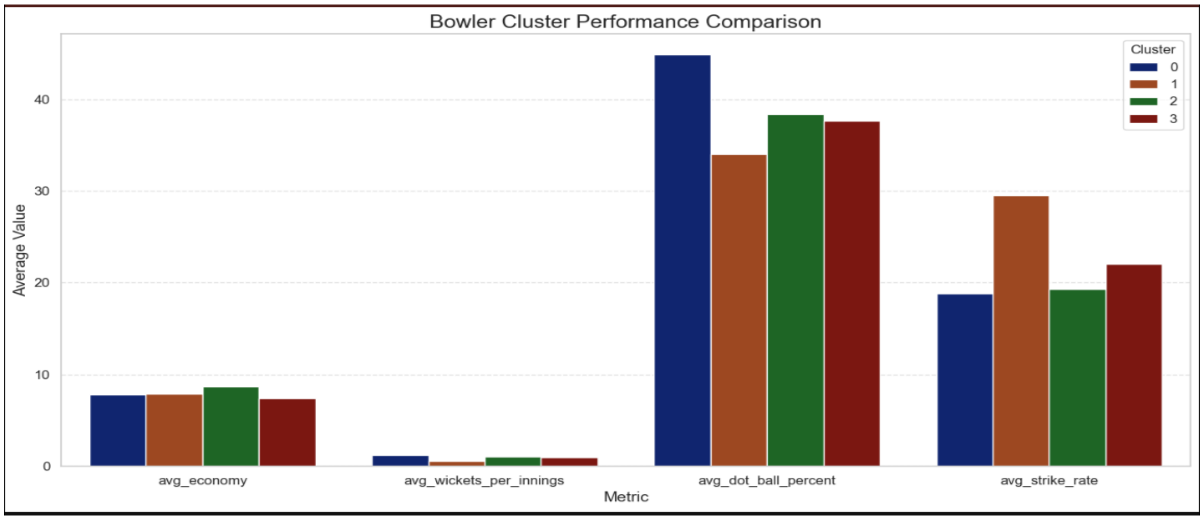
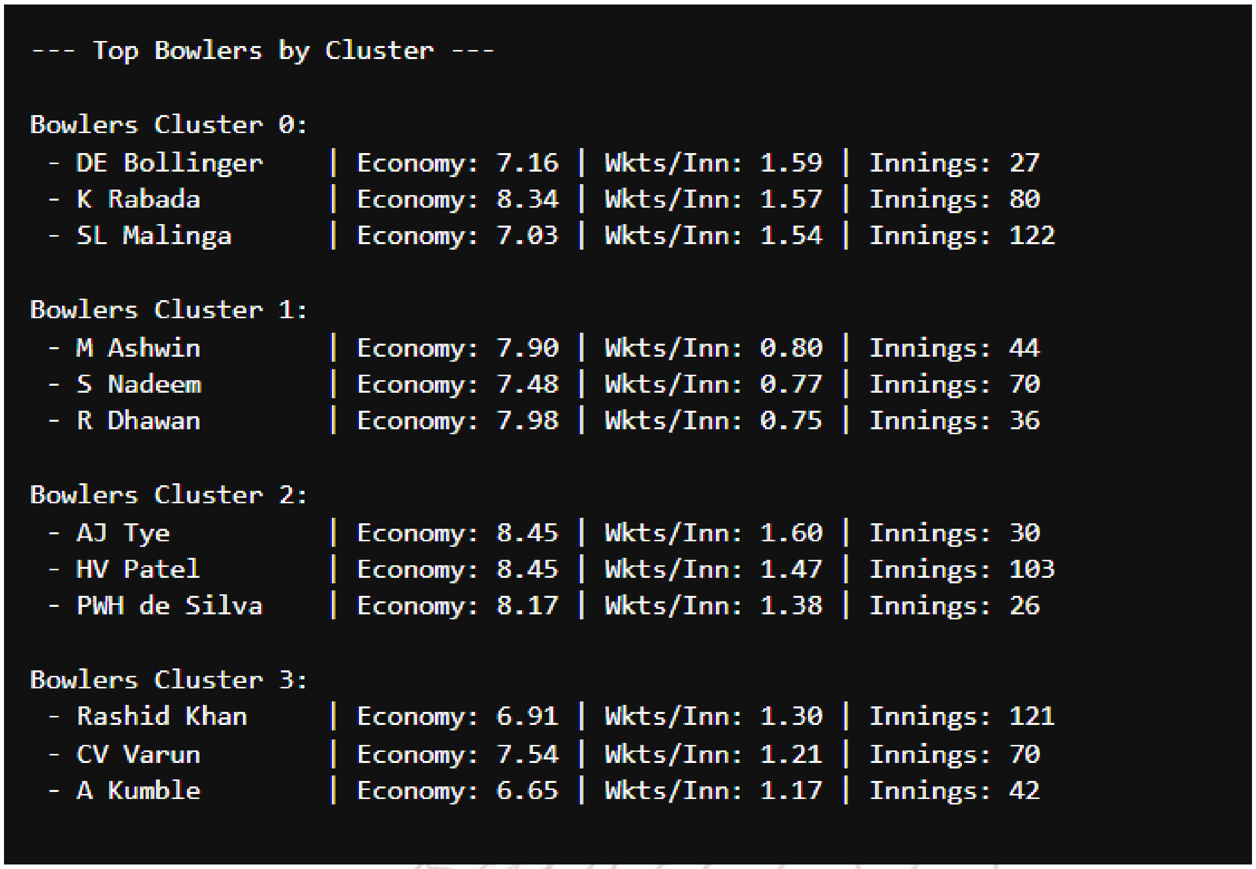


*Figure 31: Bowlers Clustering Comparison Chart*



```
--- Top Bowlers by Cluster ---

Bowlers Cluster 0:
 - DE Bollinger     | Economy: 7.16 | Wkts/Inn: 1.59 | Innings: 27
 - K Rabada         | Economy: 8.34 | Wkts/Inn: 1.57 | Innings: 80
 - SL Malinga       | Economy: 7.03 | Wkts/Inn: 1.54 | Innings: 122

Bowlers Cluster 1:
 - M Ashwin         | Economy: 7.90 | Wkts/Inn: 0.80 | Innings: 44
 - S Nadeem         | Economy: 7.48 | Wkts/Inn: 0.77 | Innings: 70
 - R Dhawan         | Economy: 7.98 | Wkts/Inn: 0.75 | Innings: 36

Bowlers Cluster 2:
 - AJ Tye           | Economy: 8.45 | Wkts/Inn: 1.60 | Innings: 30
 - HV Patel         | Economy: 8.45 | Wkts/Inn: 1.47 | Innings: 103
 - PWH de Silva     | Economy: 8.17 | Wkts/Inn: 1.38 | Innings: 26

Bowlers Cluster 3:
 - Rashid Khan      | Economy: 6.91 | Wkts/Inn: 1.30 | Innings: 121
 - CV Varun         | Economy: 7.54 | Wkts/Inn: 1.21 | Innings: 70
 - A Kumble         | Economy: 6.65 | Wkts/Inn: 1.17 | Innings: 42
```

**Interpretation**: K-means clustering has divided into 4 different groups. Many anomalies and outliers are also visible in this cluster. Each player in a cluster has different characteristics:

**Cluster 0** – These players have moderate to low economy and high wickets per match. This suggests that these players contribute consistently to the team's performance and are very reliable.

**Cluster 1** – These players have a moderate economy rate, but they have lower wickets per innings this indicates that either these players are low-performing defensive bowlers or bowlers who bowl less overs.

**Cluster 2** – These players have high wickets per innings and have high economy rates. This suggests that these bowlers could be death bowlers.

**Cluster 3** – These players have a low economy with a high wicket per innings ratio this suggests that these players are economical bowlers who are also great at taking wickets.

## 4.4 Isolation Forest Results and Implementation

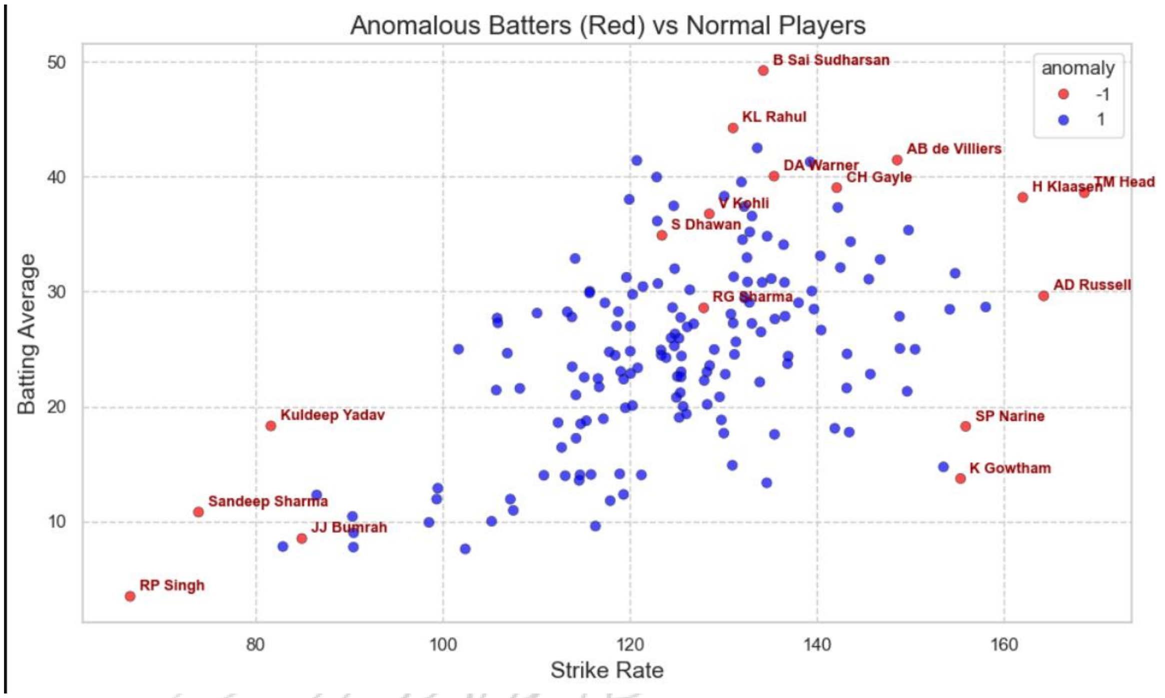
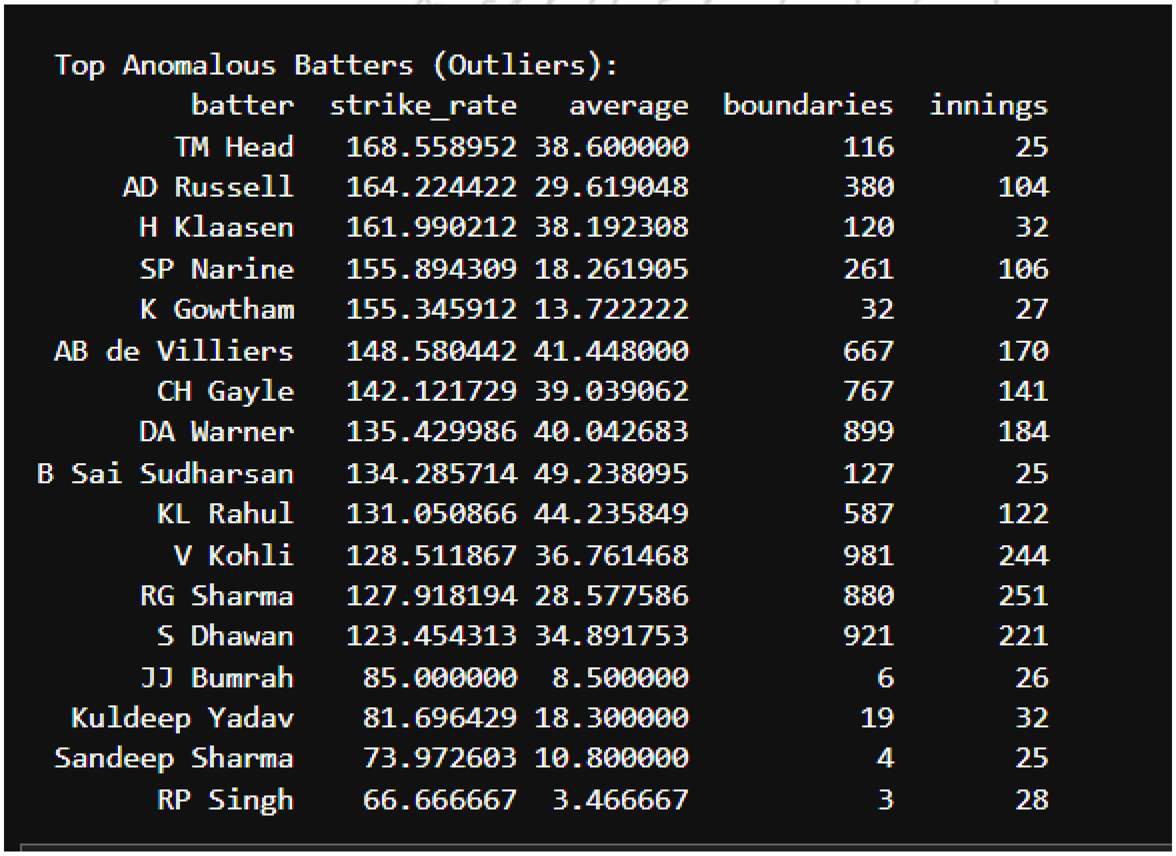


Figure 32: Isolation Forest for Batters


```
Top Anomalous Batters (Outliers):
          batter  strike_rate    average  boundaries  innings
         TM Head   168.558952  38.600000         116       25
      AD Russell   164.224422  29.619048         380      104
       H Klaasen   161.990212  38.192308         120       32
       SP Narine   155.894309  18.261905         261      106
       K Gowtham   155.345912  13.722222          32       27
 AB de Villiers   148.580442  41.448000         667      170
        CH Gayle   142.121729  39.039062         767      141
       DA Warner   135.429986  40.042683         899      184
B Sai Sudharsan   134.285714  49.238095         127       25
        KL Rahul   131.050866  44.235849         587      122
         V Kohli   128.511867  36.761468         981      244
       RG Sharma   127.918194  28.577586         880      251
        S Dhawan   123.454313  34.891753         921      221
       JJ Bumrah    85.000000   8.500000           6       26
   Kuldeep Yadav    81.696429  18.300000          19       32
  Sandeep Sharma    73.972603  10.800000           4       25
        RP Singh    66.666667   3.466667           3       28
```

**Interpretation**: This isolation forest shows 10% of Anomalous Batters from the entire dataset. In this, we can see that some of the best batters in the history of IPL are listed. This list is dominated by players with abnormally high strike rates and averages. There are few bowlers in this list too, this showcases that these bowlers can't bat well. Some players have high strike rate and low average which shows that they are either lower order hitters or pinch hitters

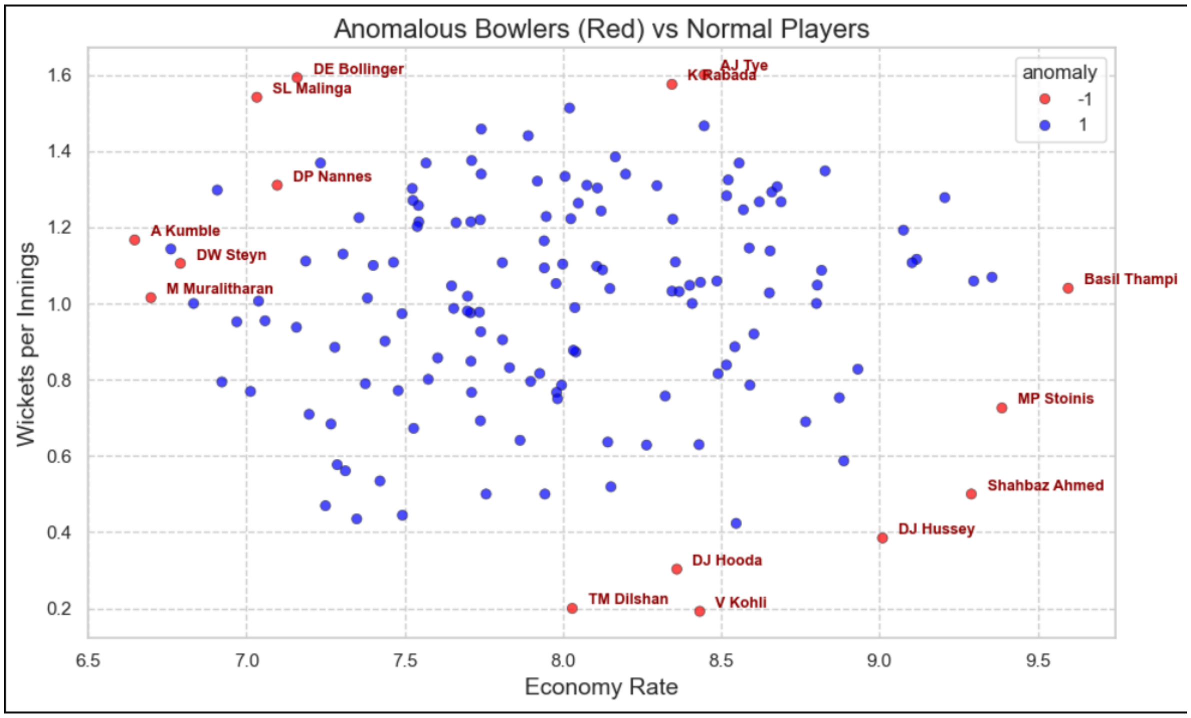
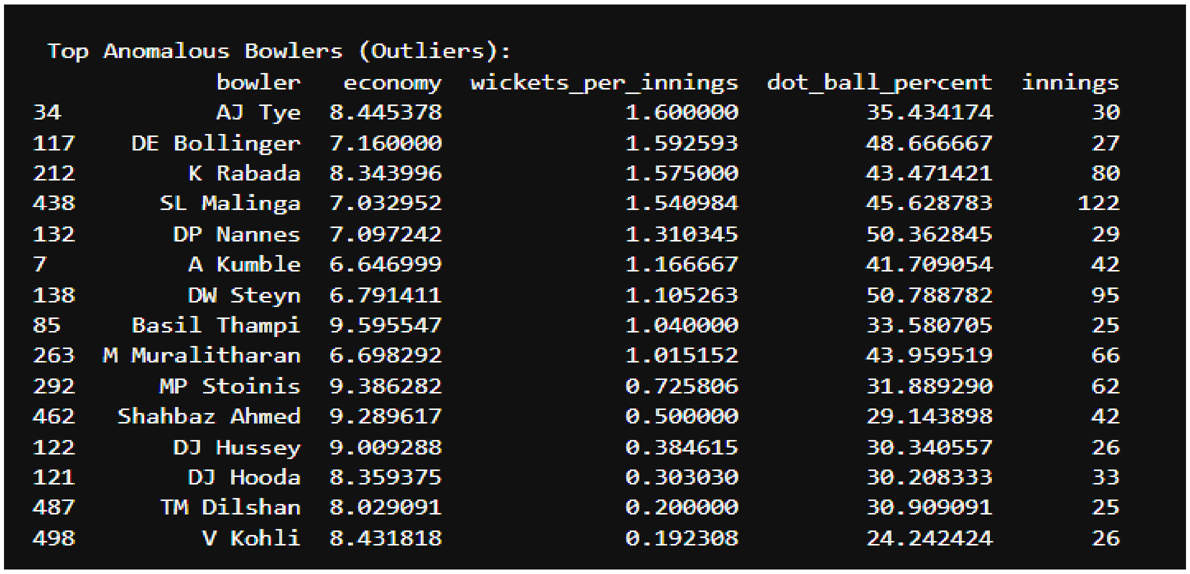


*Figure 33: Isolation Forest for Bowlers*


```
 Top Anomalous Bowlers (Outliers):
             bowler   economy  wickets_per_innings  dot_ball_percent  innings
34           AJ Tye  8.445378             1.600000         35.434174       30
117     DE Bollinger  7.160000             1.592593         48.666667       27
212         K Rabada  8.343996             1.575000         43.471421       80
438       SL Malinga  7.032952             1.540984         45.628783      122
132        DP Nannes  7.097242             1.310345         50.362845       29
7           A Kumble  6.646999             1.166667         41.709054       42
138         DW Steyn  6.791411             1.105263         50.788782       95
85      Basil Thampi  9.595547             1.040000         33.580705       25
263   M Muralitharan  6.698292             1.015152         43.959519       66
292       MP Stoinis  9.386282             0.725806         31.889290       62
462    Shahbaz Ahmed  9.289617             0.500000         29.143898       42
122        DJ Hussey  9.009288             0.384615         30.340557       26
121         DJ Hooda  8.359375             0.303030         30.208333       33
487       TM Dilshan  8.029091             0.200000         30.909091       25
498          V Kohli  8.431818             0.192308         24.242424       26
```

**Interpretation**: This list of anomalous bowlers is dominated by players who have low economy rate, high wicket per innings, and dot ball percentage. These players are the top performing bowlers of the IPL. Basil Thampi is a unique bowler as he has a high economy, low dot ball percentage, and low wickets per innings. The rest of the players in this list are either batting all-rounders or part-timers.

# 5. Insights and Recommendations

## 5.1 Insights

After applying multiple data mining techniques to IPL dataset many actionable insights have been generated. Some of the major insights important to our stakeholders are:

- **IPL is Becoming More Aggressive**

  There has been a major shift in how the game is played. Teams and players have started to opt for more aggressive versions of battings. Due to this reason the season run rate and batter strike rate have been going up. IPL teams should prioritize more aggressive batters with high strike rates. Bowlers who are strong during death overs have also been invaluable due to the recent shift in batting approach.

- **Consistent Players Pairs Correlate with Higher Win Rate**

  This analysis showed that batting pairs with high win rates came from franchises like Chennai Super Kings and Mumbai Indians. This means that maintaining core partnerships in a lineup can positively impact outcomes.

- **Distinct Roles can be Identified using Performance based Clustering**

  K-Means clustering showed that players can be divided into distinct roles such as power hitters, finishers, strike bowlers, economical bowlers, etc. This method can assist teams build better teams and can also help fantasy sports players make better decisions.

- **Scoring Rates Depends on Host nation**

  From this analysis, it was seen that whenever IPL has been played outside of idea the scoring rate has fluctuated. This showcases the strong impact of conditions on cricket.

- **Implications of Outlier detection**

Anomaly detection successfully highlights both high-performing and low-performing players based on simple cricketing statistics. These insights can help in player scouting and risk assessment.

These insights can help teams make better decisions regarding team selection, bettors and fantasy sports players can pick better teams based on these insights.

## 5.2 Recommendations

From my experience of doing the project, I recommend the following recommendations:

- Teams should adopt a more data-driven approach, batters should be backed for a certain time before making changes. Teams should actively use insights to ensure they have a balanced lineup to maximize their strengths. Teams should also consider buying players with well-defined roles such as power hitters, strike bowlers, etc.
- Coach and team managers should use anomaly detection techniques to identify rising talents and underperforming players. This can help teams better understand their own players and build better match-winning combinations.
- Bettors and Fantasy League players should keep up with new insights to make better decisions. Anomalies can indicate an exceptional player that they would want on their fantasy team. These stakeholders should also focus on the scoring trends to better predict and bet on various metrics.
- Understanding these trends, partnerships, and anomalies, can help fans better understand and relate to the game of cricket. These insights allow for a more meaningful conversation around sports.
- Any data analyst working on IPL data should also consider other variables rather than just plain statistics. Various factors such as match conditions, venue, match importance, etc. can be considered while doing such analytics which I was not able to do in this project. Anomaly detection successfully highlights both high-performing and low-performing players based on simple cricketing statistics. These insights can help in player scouting and risk assessment.

## 6.0 Ethical Considerations

For this project, a publicly available IPL dataset was used and does not include any identifiable information. This dataset has been web-scrapped from the ESPN Cricinfo website. In this project, as an analyst, I have tried to produce the fairest and biasless insights.  All data used in this project is publicly available and does not contain any personal information. Such information and wrongly sourced data can cause harm to a company or a person. Any analyst working on any project should respect players and ensure confidentiality.

In this project, I have tried to remove bias as much as possible by creating filters such as minimum matches to qualify for a metric. These filters make sure that the analytics are fairer. Players with fewer matches might appear on anomaly detection which does not represent the data properly leading to misleading conclusions. In this project, I have tried to remove these biases as much as possible.

The findings of this project should just not be used to make any kind of decision, whether it's regarding a player, a bet, or a fantasy team. While data-driven insights are valuable alone they cannot be considered while making such important decisions.

All the visualizations, data mining techniques, and interpretations are intended for educational purposes. This project does not aim to defame any player or team. Insights from this project may support coaches, selection, or performance evaluation but these insights should not be the only metric. Also, these insights should not be used for any publicly negative narrative that may impact players' reputations or the betting market.

## **7.0** Conclusion

This project is aimed at applying various data mining techniques to the Indian Premier League dataset to identify various patterns, trends and insights. Various techniques such as Time Series Analysis, K-means Clustering, Association Analysis, and anomaly detection were used in this project. Using this technique a high-level understanding of leagues evolution, strategic partnerships, and player performance was achieved.

Time series analysis showed how the league has transformed into a more aggressive and high-scoring tournament. This analysis also showed how a change in venue or host country changed the way cricket was played. K- means clustering was used to differentiate players into various natural categories. Using this technique players who play a specific role were identified. Association analysis of player pairs shows how consistent team formation impacts team performance. Anomaly detection further enabled us to identify overperformers and underperformers in IPL.

These findings have direct applications across our stakeholders. Teams can pick teams better, produce better scouting results, and can make better retention strategies. For bettors and fantasy players, this project showcased that data-driven patterns can improve prediction accuracy and help them make better decisions. For fans, these analytics enable them to have a view of this game from a different perspective.

In conclusion, this study shows that data mining techniques can be used to generate valuable and actionable insights in the space of sports analytics. Despite all the technical and time-related limitations, various valuable insights were generated. Integration of other more advanced data mining techniques and Artificial Intelligence can help make this project more insightful.

## References

Adams, R. P. (n.d.). K-Means Clustering and Related Algorithms. *Elements of Machine Learning*.

Ajay, L. (2021, June 6). *Machine Learning to Cluster Cricket Players*. Retrieved from Towards Data Science: https://towardsdatascience.com/machine-learning-to-cluster-cricket-players-1d53beeb69b4/

Balajisr. (2024, June 26). *Is Jasprit Bumrah a Genius Bowler? Using AutoEncoders for Anomaly Detection in Cricket*. Retrieved from Analytics vidhya: https://www.analyticsvidhya.com/blog/2024/06/anomaly-detection-with-autoencoders-in-cricket/

*IsolationForest*. (n.d.). Retrieved from Scikit Learn : https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Majumdar, B. (n.d.). The Indian Premier League. In *The Cambridge Companion to Cricket* (pp. 173–186). Cambridge University Press.

Mukherjee, S. (2013). Complex Network Analysis in Cricket : Community structure, player's role and performance index. *Advances in Complex Systems*.

Sumathi, M., Prabu, S., & Murugesan, R. (2023). Cricket Players Performance Prediction and Evaluation Using Machine Learning Algorithms. *Conference: 2023 International Conference on Networking and Communications (ICNWC)*, 1-6.

*Top-10 List of the World's Most Popular Sports*. (n.d.). Retrieved from Top End Sports : https://www.topendsports.com/world/lists/popular-sport/fans.htm

Wei, W. W. (n.d.). *Time Series Analusis Univariate and Multivariate Methods.* Pearson Education, Inc.

[illegible]