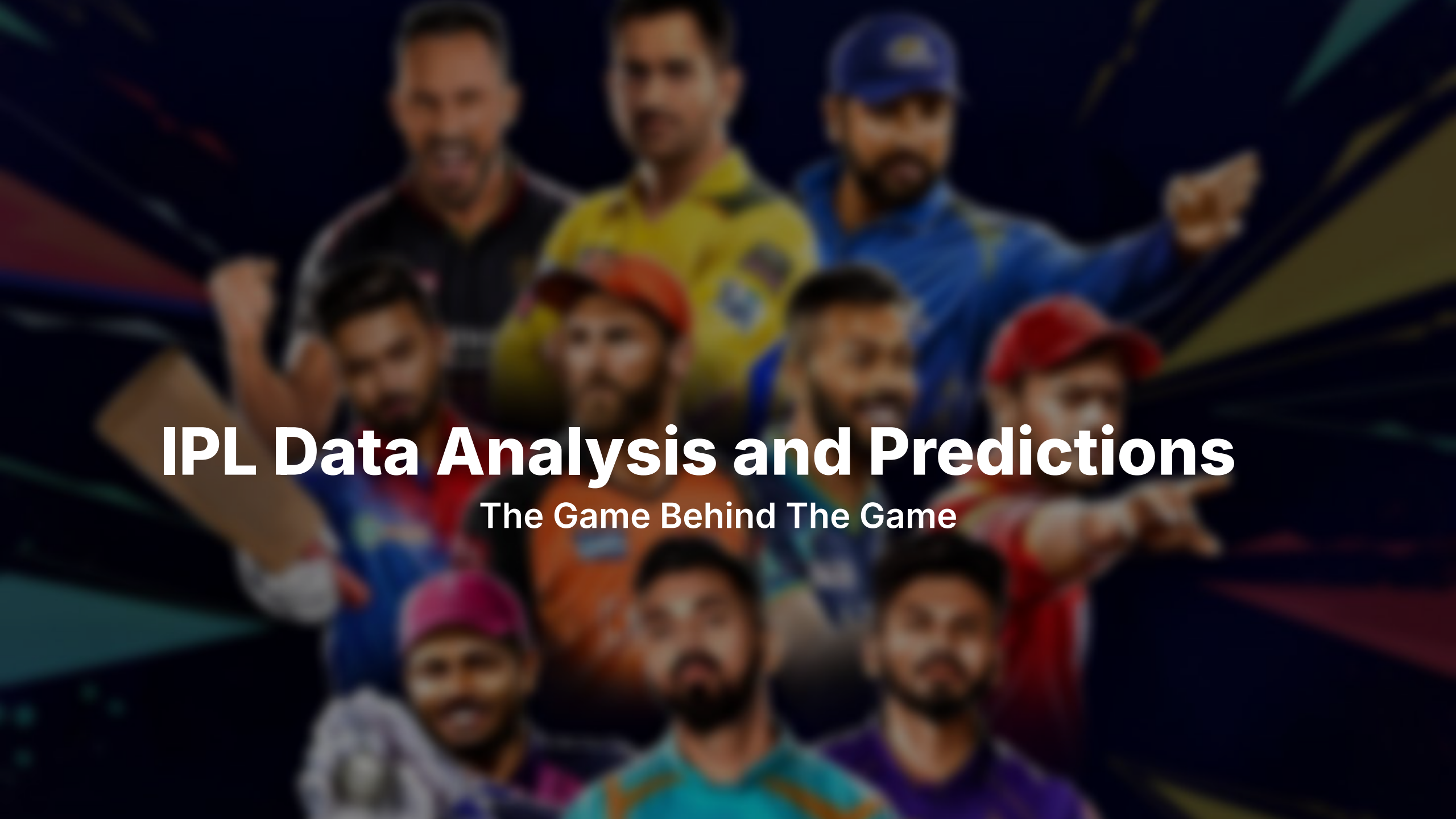
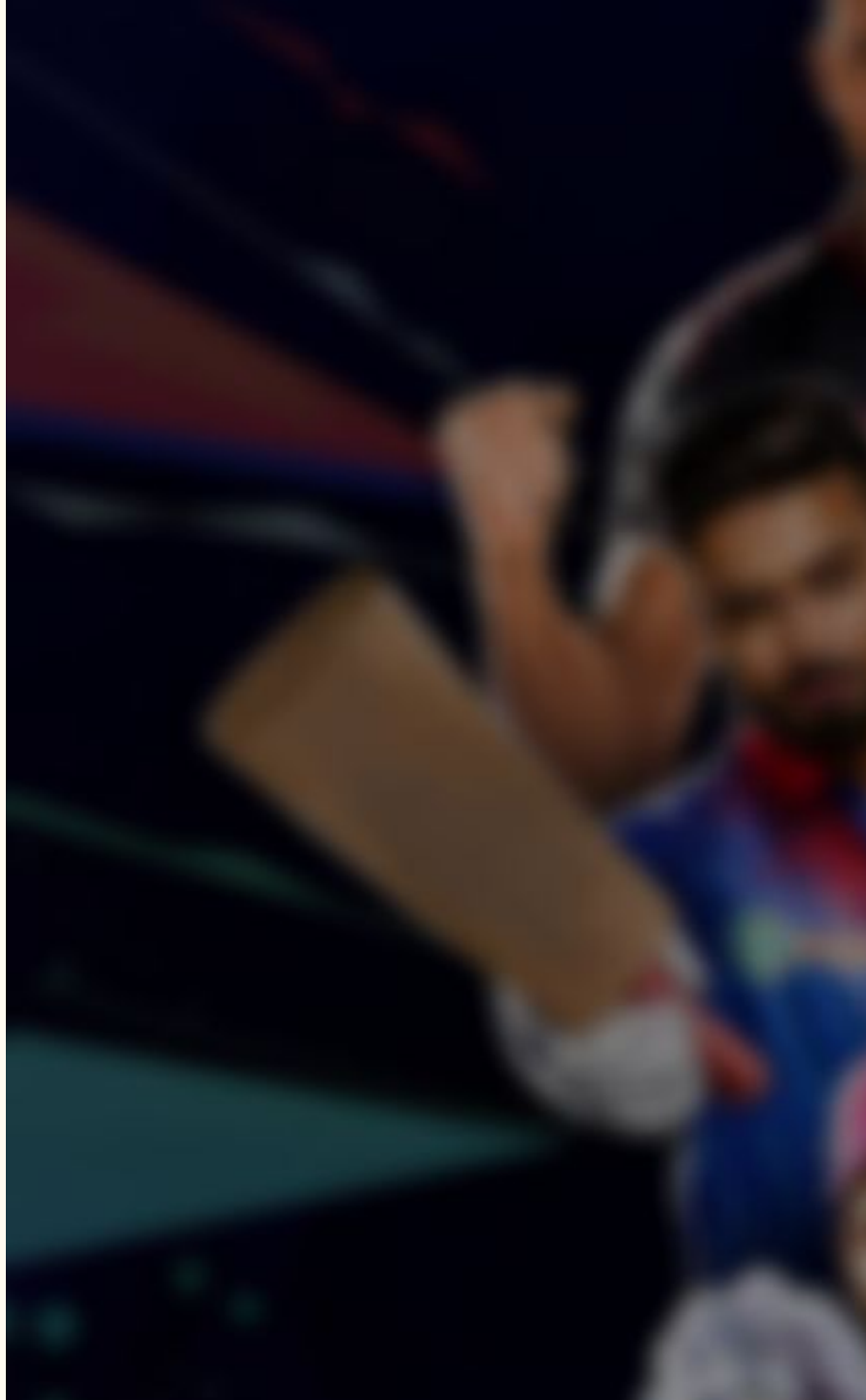


IPL Data Analysis and Predictions

The Game Behind The Game

Table of Contents

1. Introduction
2. Exploratory Data Analysis (EDA)
3. Dataset Pre-processing
4. Model Architecture and Training Methodology
5. Performance Evaluation
6. Results and Discussion
7. Conclusion
8. References



Introduction

IPL, India's most famous cricket tournament is set have its 18th season in 2025. Our aim is build a model to predict the outcome of future IPL matches using past data by performing a comprehensive analysis of data since first season of IPL in 2008 to make predictions in the 2025 season by uncovering key insights, trends, and patterns using data collection, preprocessing, and exploratory data analysis (EDA).

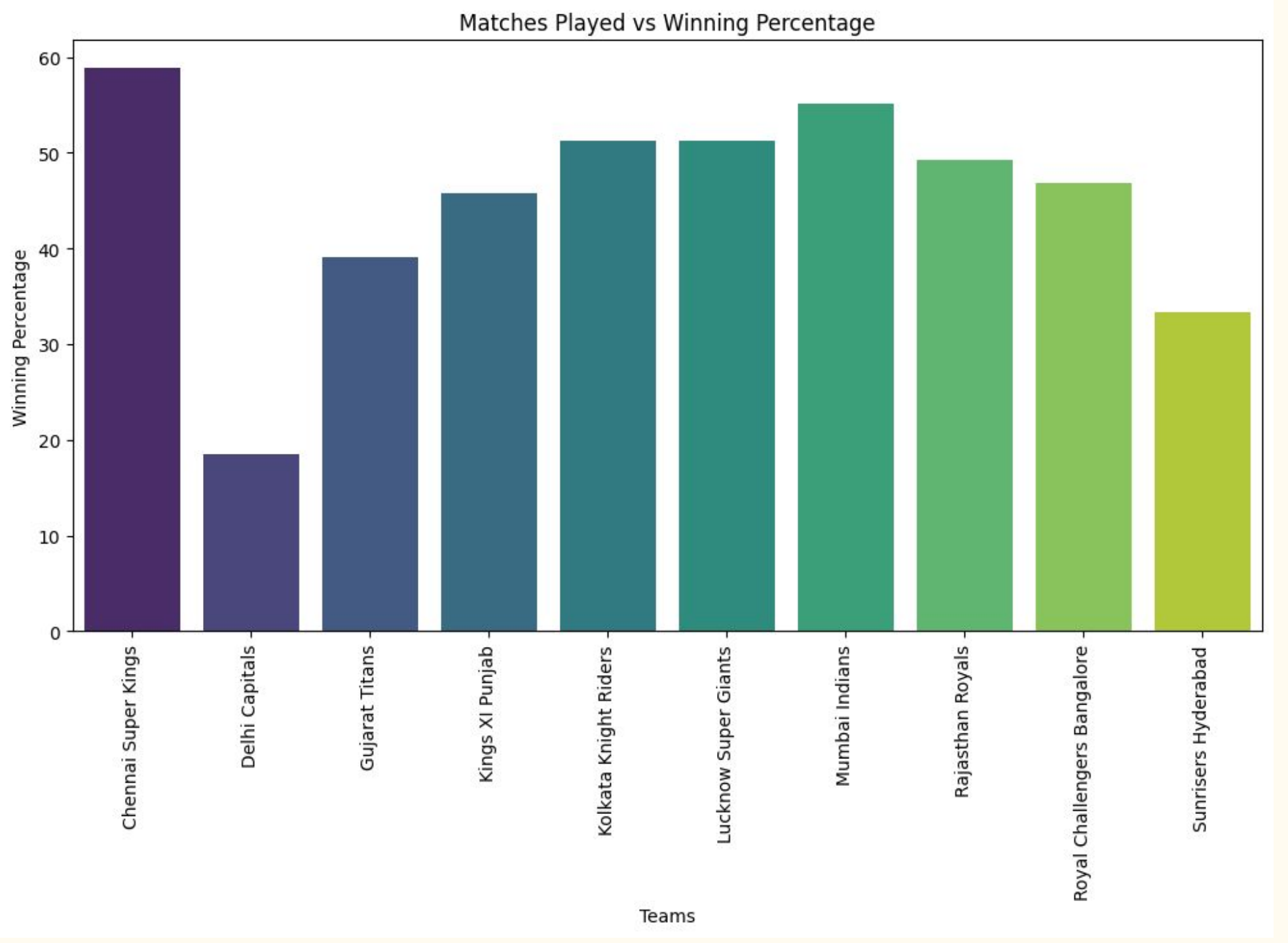
Before diving into our report check out our

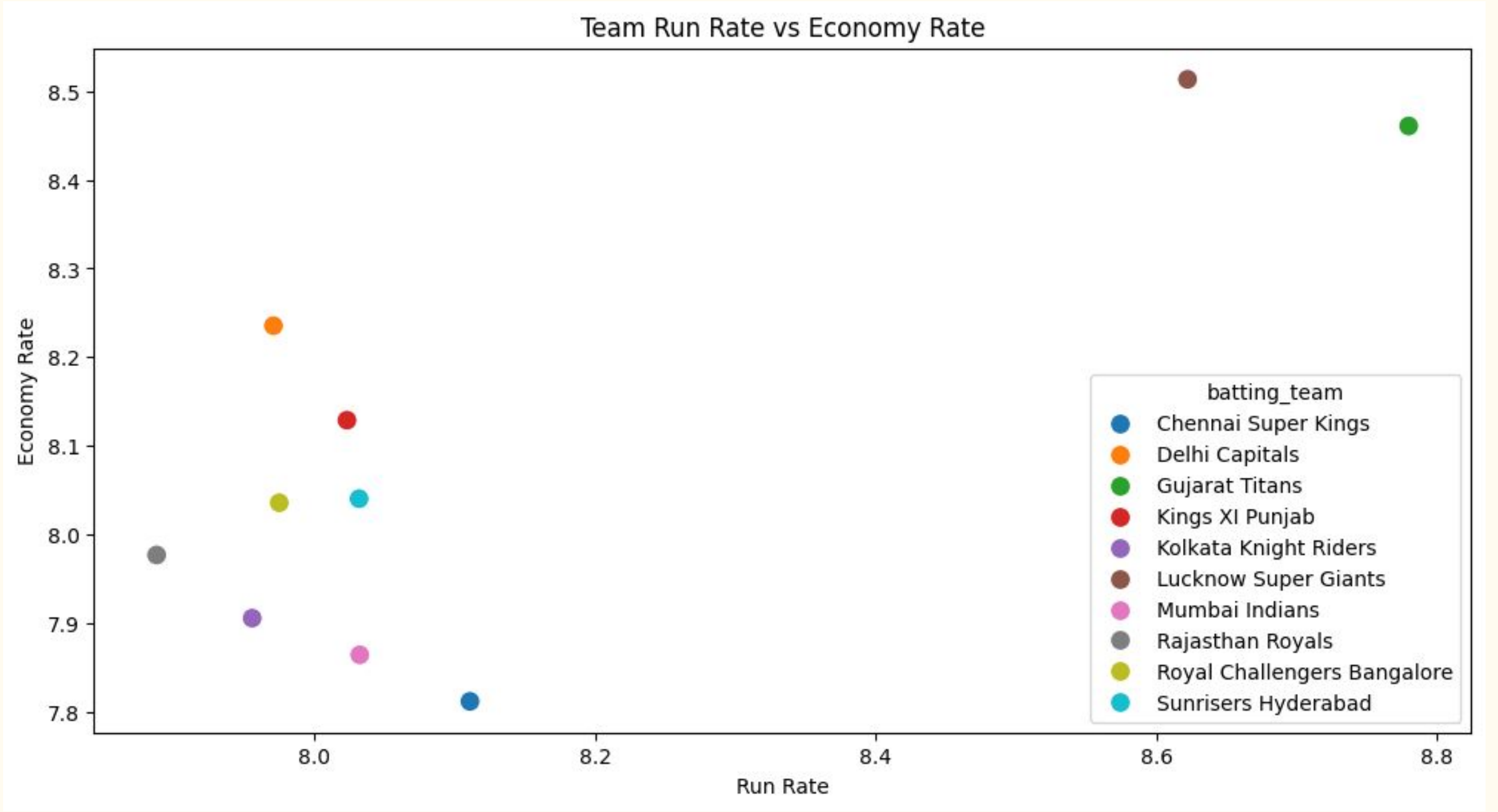
IPL Winner Predictor! (Scan QR code or access link below)

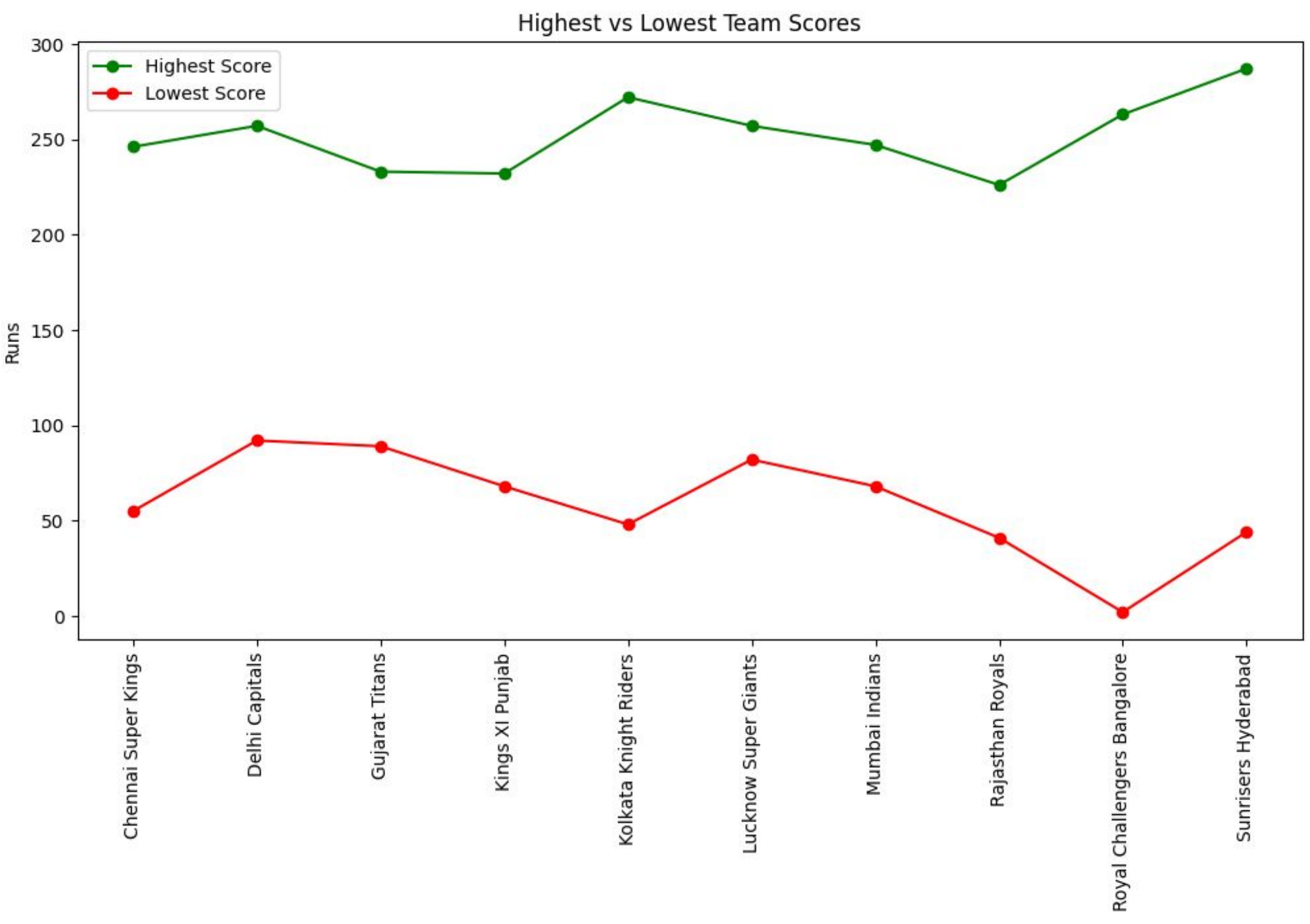
https://ipl-win-predictor-6jfx.onrender.com/

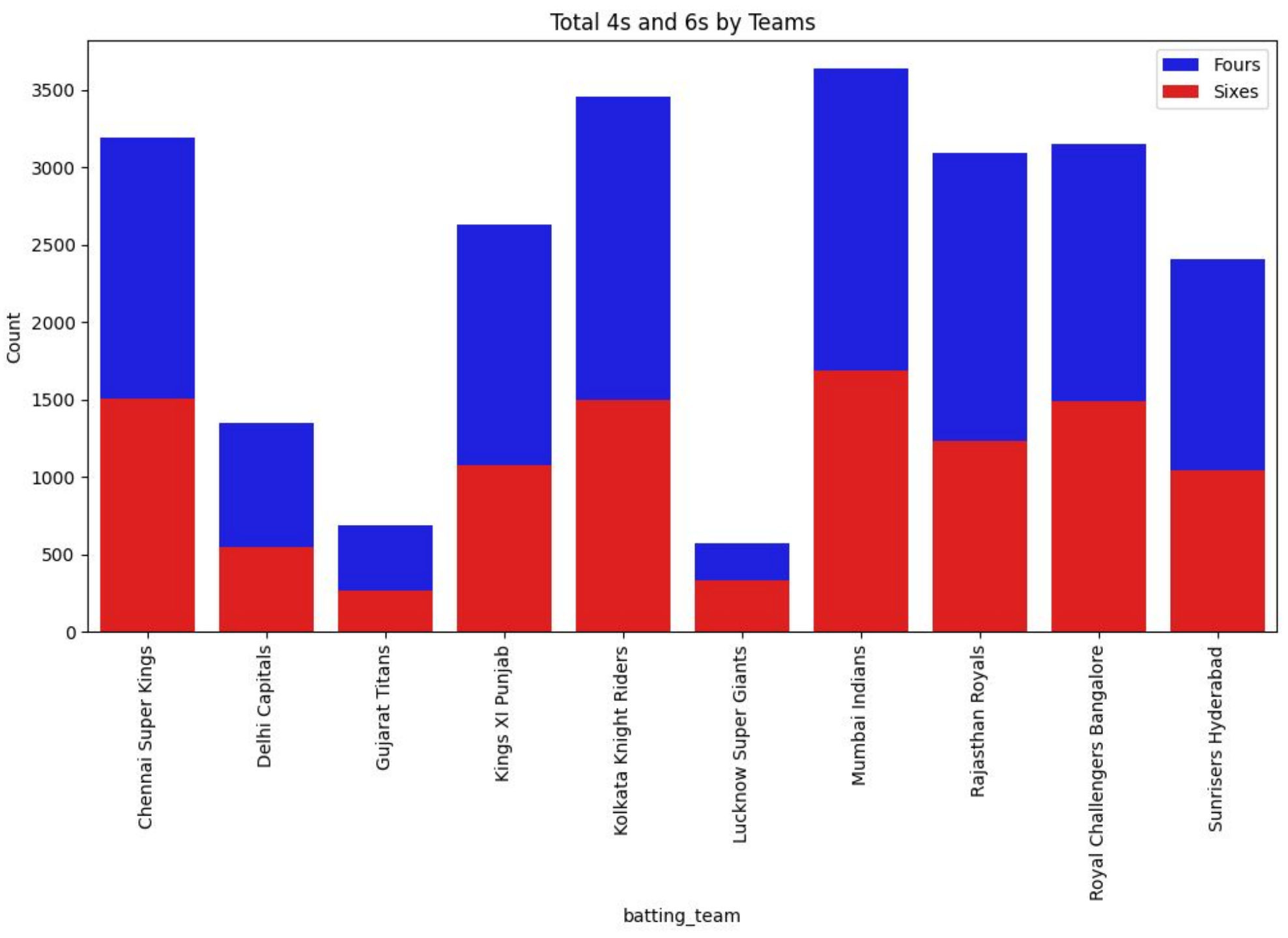
Team Performance Metrics

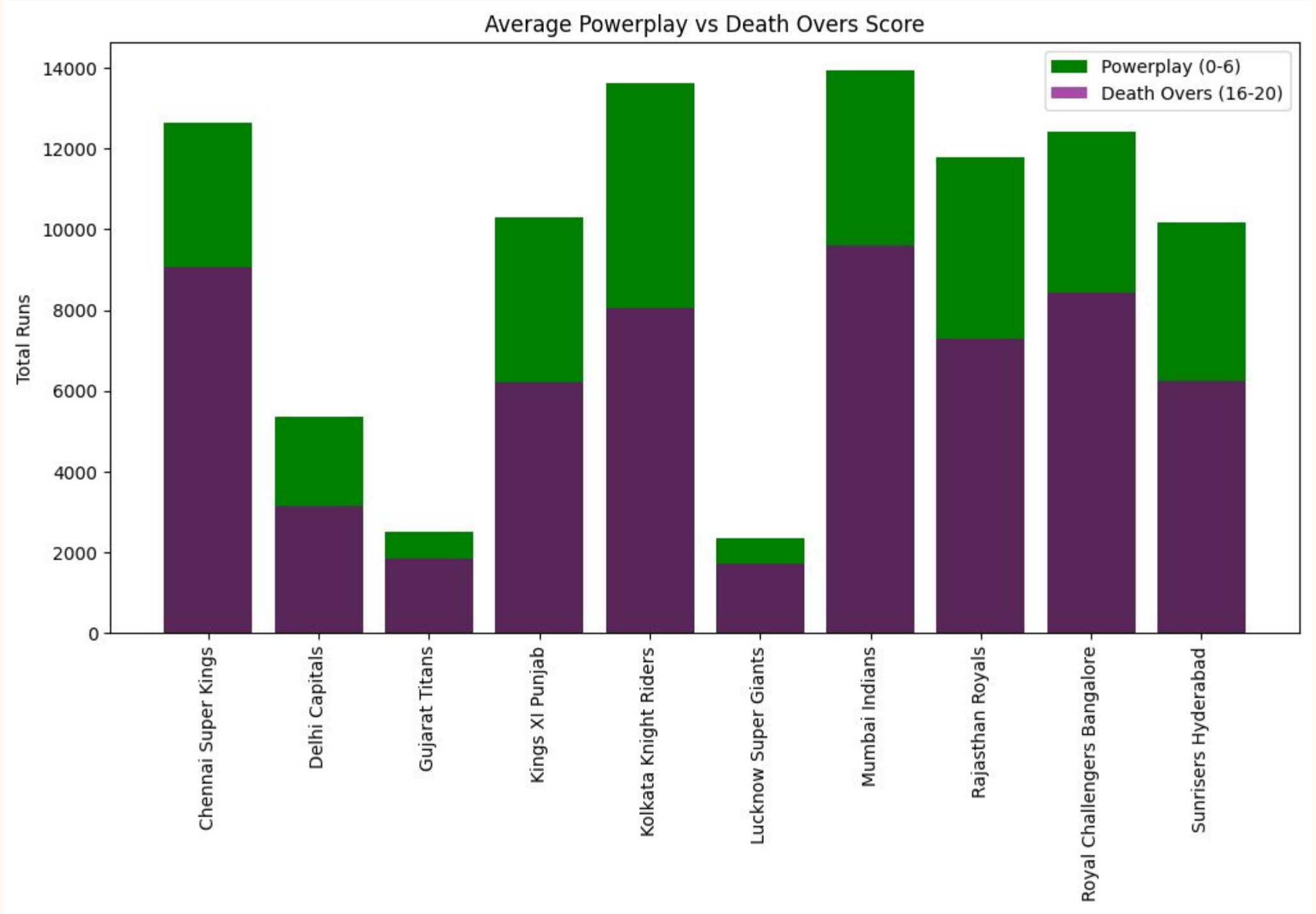
- Plotted Matches Played vs Winning Percentages.
- Plotted Run Rate and Economy Rate as a bowling side.
- Plotted the highest and lowest scores.
- Plotted the total number of 4s and 6s.
- Plotted the average scores in Powerplay and Death Overs.


Matches Played vs Winning Percentage
Winning Percentage
0
10
20
30
40
50
60
Chennai Super Kings
Delhi Capitals
Gujarat Titans
Kings XI Punjab
Kolkata Knight Riders
Lucknow Super Giants
Mumbai Indians
Rajasthan Royals
Royal Challengers Bangalore
Sunrisers Hyderabad
Teams


Team Run Rate vs Economy Rate
Economy Rate
Run Rate
8.5
8.4
8.3
8.2
8.1
8.0
7.9
7.8
8.0
8.2
8.4
8.6
8.8
batting_team
Chennai Super Kings
Delhi Capitals
Gujarat Titans
Kings XI Punjab
Kolkata Knight Riders
Lucknow Super Giants
Mumbai Indians
Rajasthan Royals
Royal Challengers Bangalore
Sunrisers Hyderabad

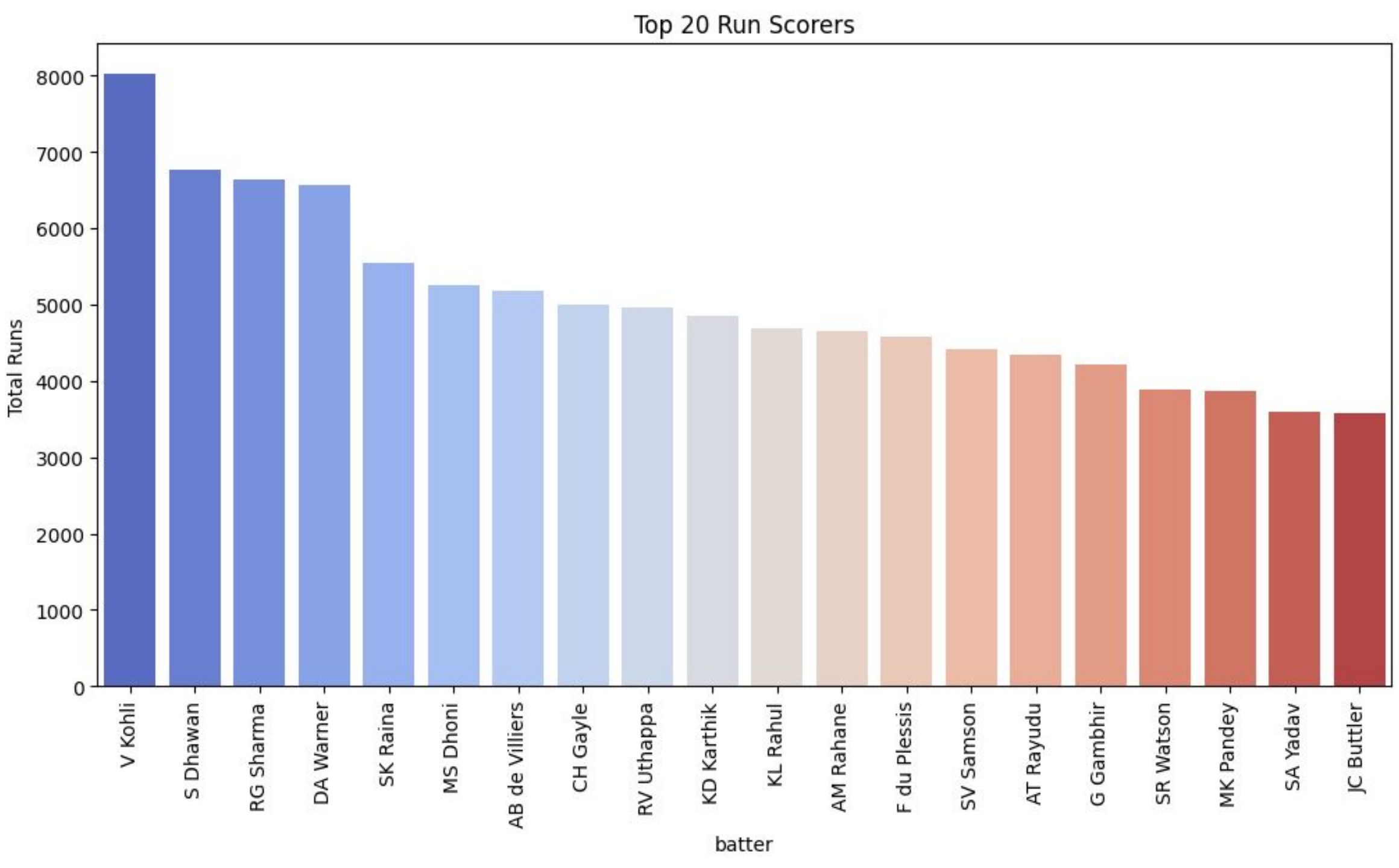

Highest vs Lowest Team Scores
Highest Score
Lowest Score
Runs
300
250
200
150
100
50
0
Chennai Super Kings
Delhi Capitals
Gujarat Titans
Kings XI Punjab
Kolkata Knight Riders
Lucknow Super Giants
Mumbai Indians
Rajasthan Royals
Royal Challengers Bangalore
Sunrisers Hyderabad

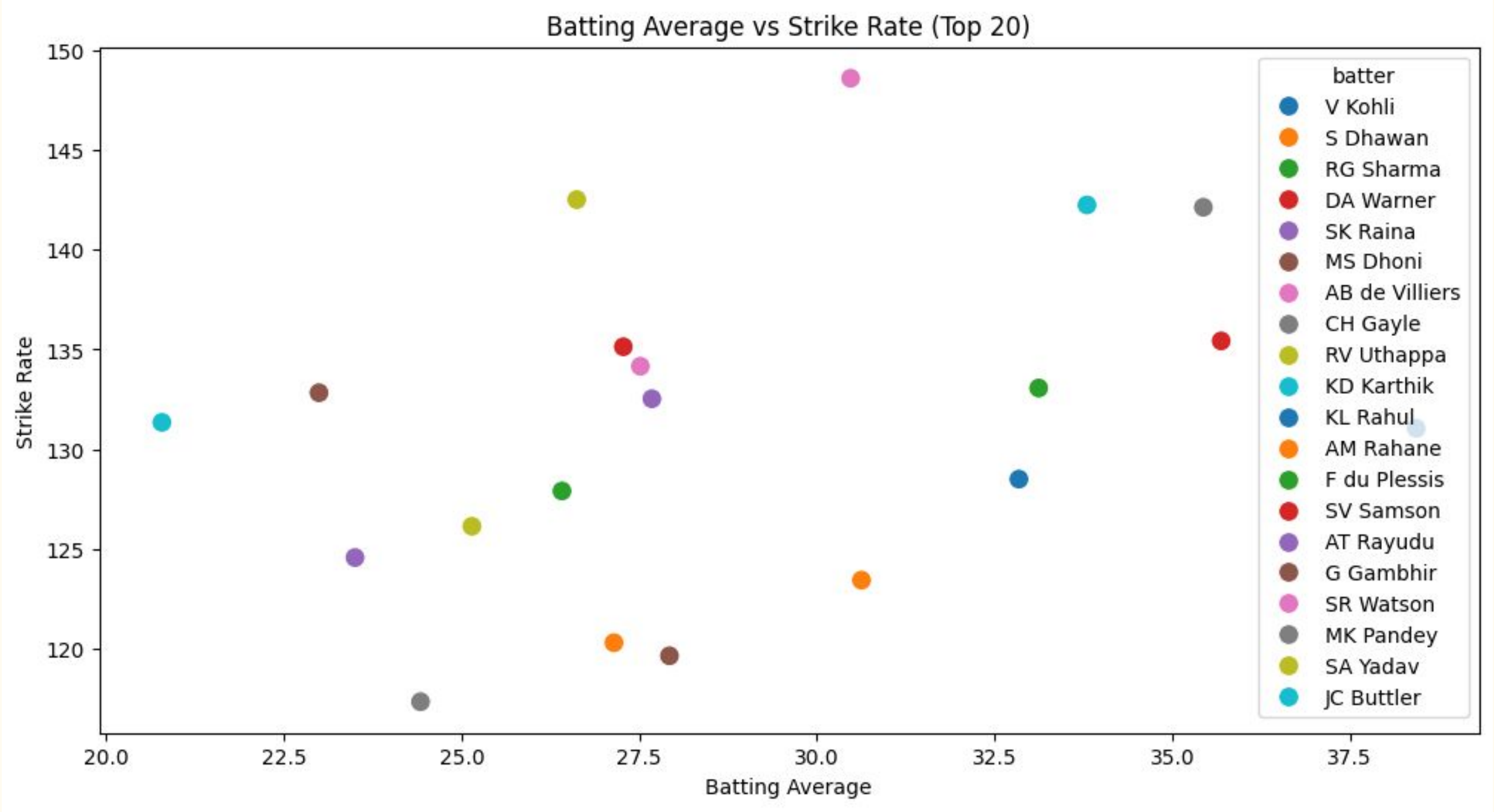

Total 4s and 6s by Teams
Fours
Sixes
Count
0
500
1000
1500
2000
2500
3000
3500
Chennai Super Kings
Delhi Capitals
Gujarat Titans
Kings XI Punjab
Kolkata Knight Riders
Lucknow Super Giants
Mumbai Indians
Rajasthan Royals
Royal Challengers Bangalore
Sunrisers Hyderabad
batting_team


Average Powerplay vs Death Overs Score
Powerplay (0-6)
Death Overs (16-20)
Total Runs
0
2000
4000
6000
8000
10000
12000
14000
Chennai Super Kings
Delhi Capitals
Gujarat Titans
Kings XI Punjab
Kolkata Knight Riders
Lucknow Super Giants
Mumbai Indians
Rajasthan Royals
Royal Challengers Bangalore
Sunrisers Hyderabad

Player Performance Metrics

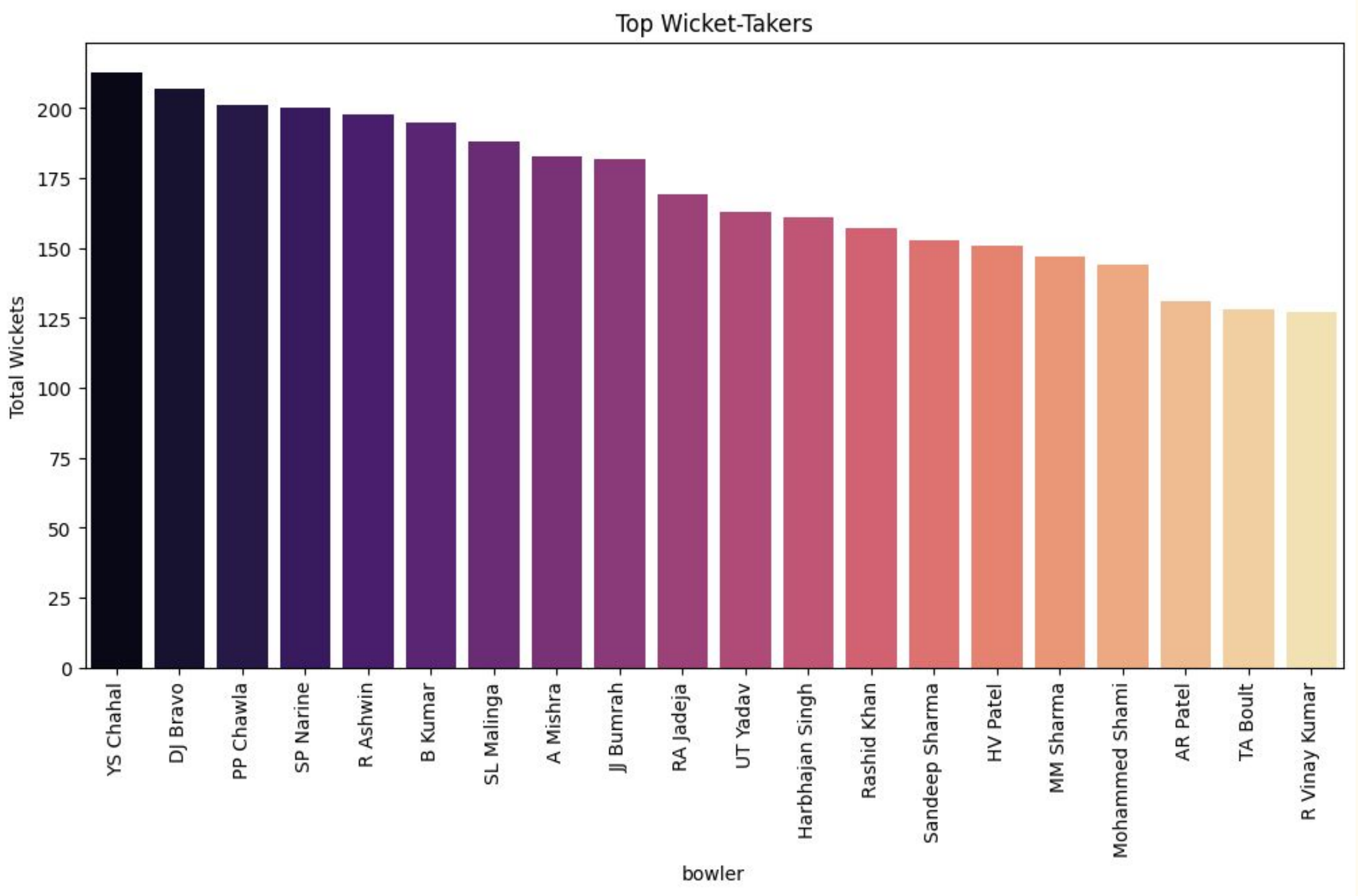
- Retrieved the top 20 run-scorers.
- Plotted Batting Average vs Batting Strike Rate for the top 20 run-scorers.
- Identified the highest Batting Average and Strike Rate for players with more than 50 matches.
- Plotted the top wicket-takers.
- Plotted the highest individual scores in a match.
- Analyzed the count of Man of the Match awards.
- Applied K-Means Clustering (k=3) to plot Batting Average vs Bowling Economy Rate, categorizing players as Batsmen, Bowlers, and All-Rounders.
- Identified the Top 10 batsmen in each run-scoring category:
  - Top 6’s scorer
  - Top 4’s scorer
  - Top 2’s scorer
  - Top 1’s scorer

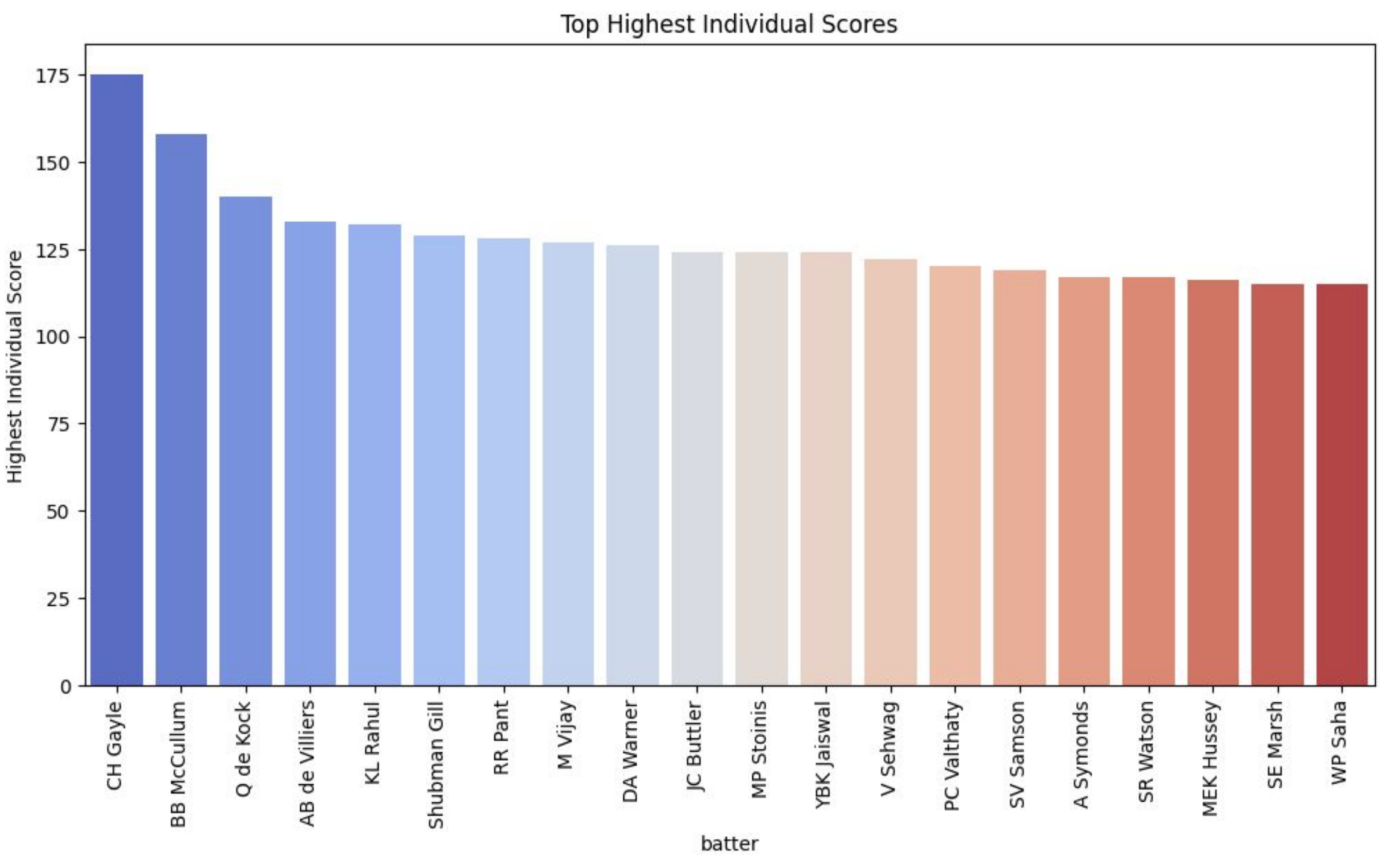

Top 20 Run Scorers
Total Runs
0
1000
2000
3000
4000
5000
6000
7000
8000
V Kohli
S Dhawan
RG Sharma
DA Warner
SK Raina
MS Dhoni
AB de Villiers
CH Gayle
RV Uthappa
KD Karthik
KL Rahul
AM Rahane
F du Plessis
SV Samson
AT Rayudu
G Gambhir
SR Watson
MK Pandey
SA Yadav
JC Buttler
batter

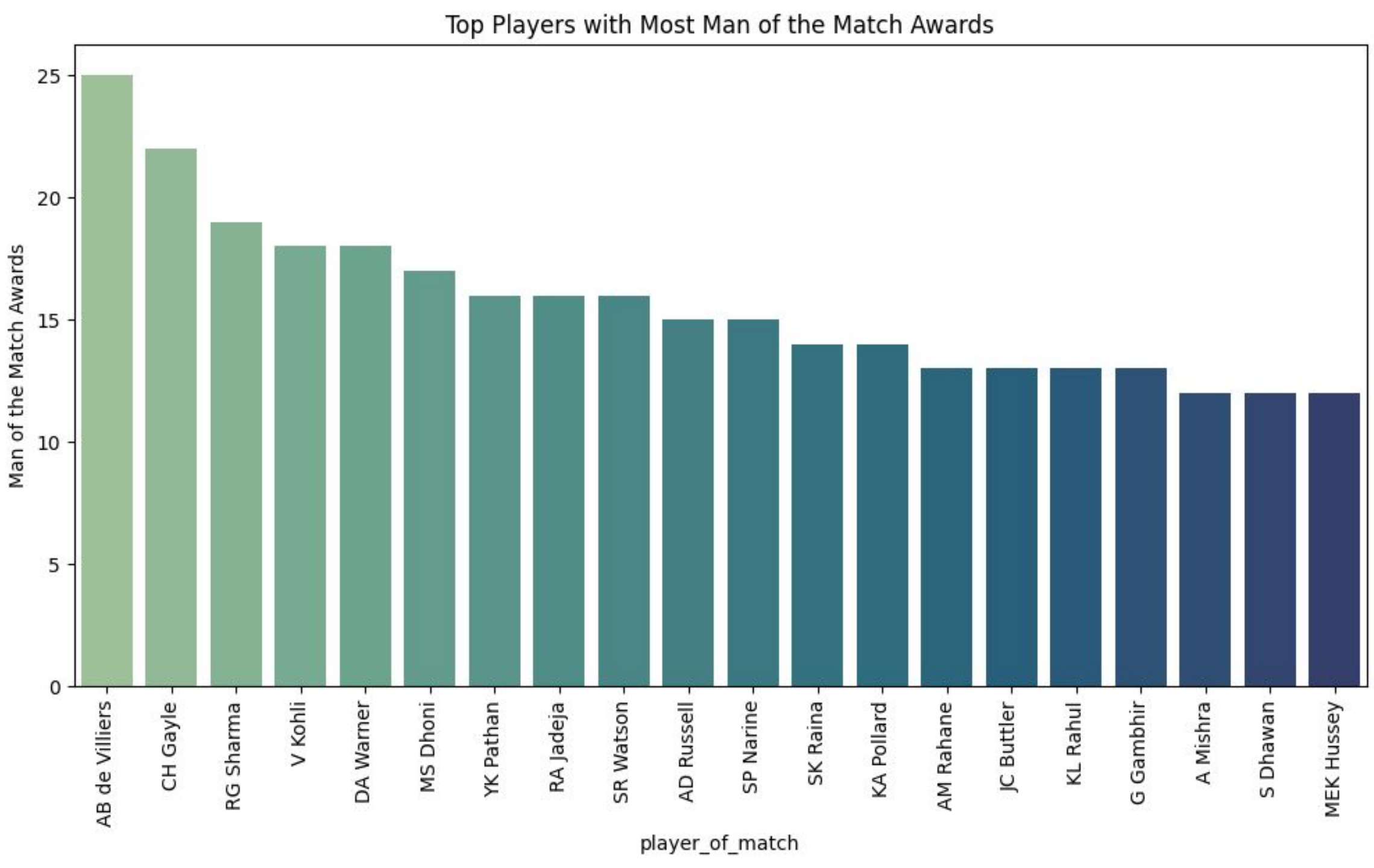

Batting Average vs Strike Rate (Top 20)
Strike Rate
Batting Average
150
145
140
135
130
125
120
20.0
22.5
25.0
27.5
30.0
32.5
35.0
37.5
batter
V Kohli
S Dhawan
RG Sharma
DA Warner
SK Raina
MS Dhoni
AB de Villiers
CH Gayle
RV Uthappa
KD Karthik
KL Rahul
AM Rahane
F du Plessis
SV Samson
AT Rayudu
G Gambhir
SR Watson
MK Pandey
SA Yadav
JC Buttler

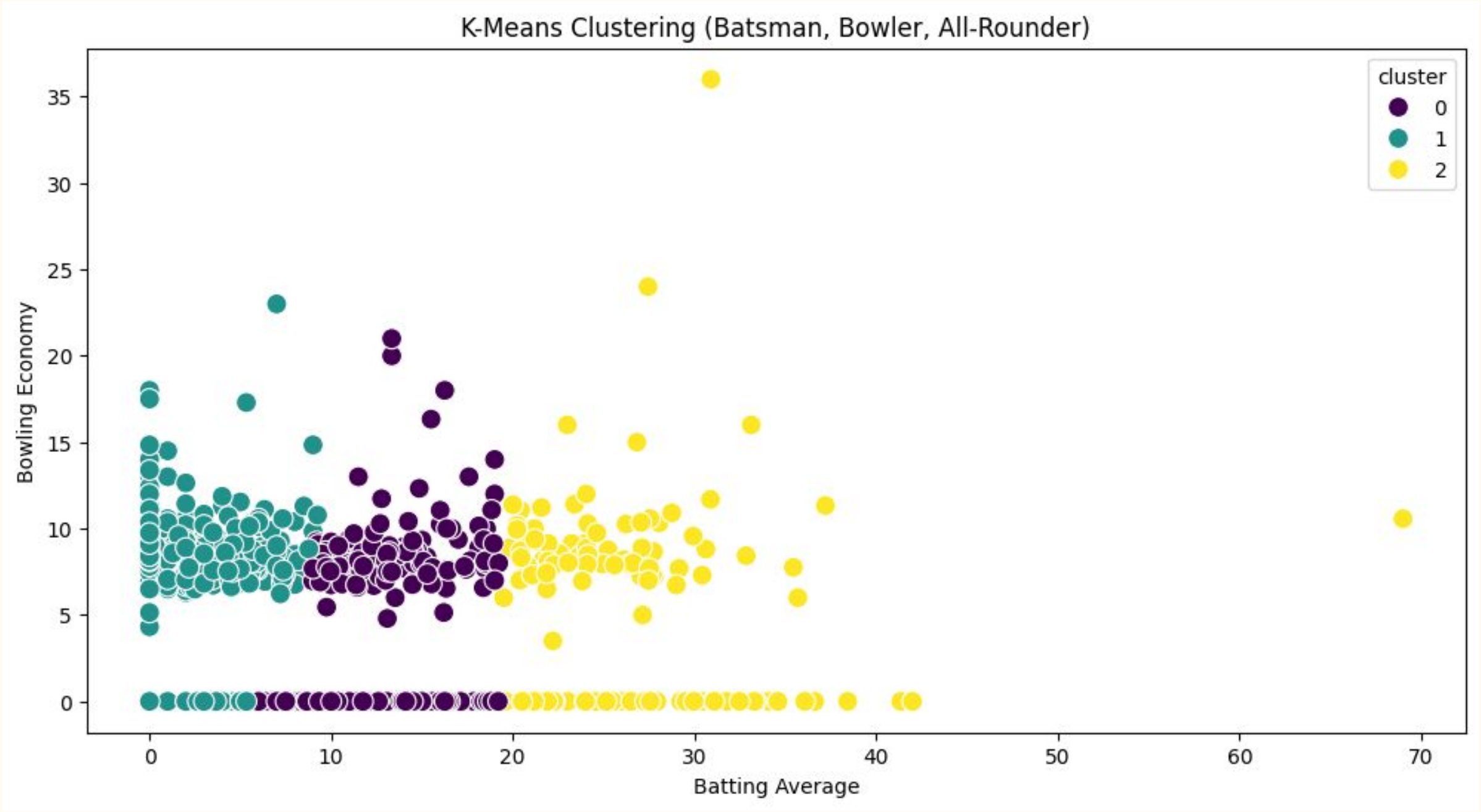
Highest Average and Strike Rate for players with >50 matches

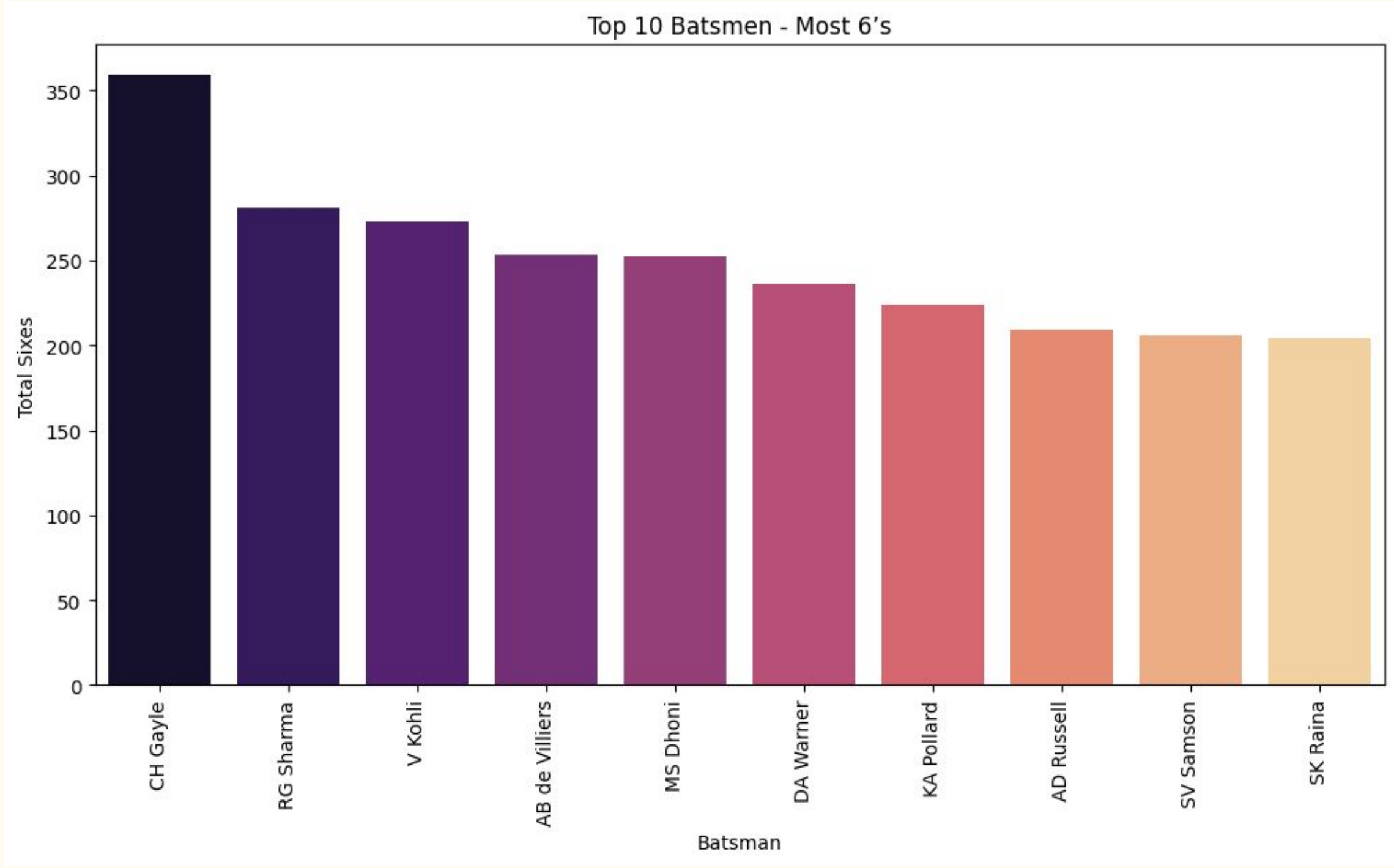
```
🏏 Player with Highest Batting Average (Min 50 Matches):
batting_avg     38.434426
matches        122.000000
Name: KL Rahul, dtype: float64

⚡ Player with Highest Strike Rate (Min 50 Matches):
strike_rate    164.224422
matches        104.000000
Name: AD Russell, dtype: float64
```

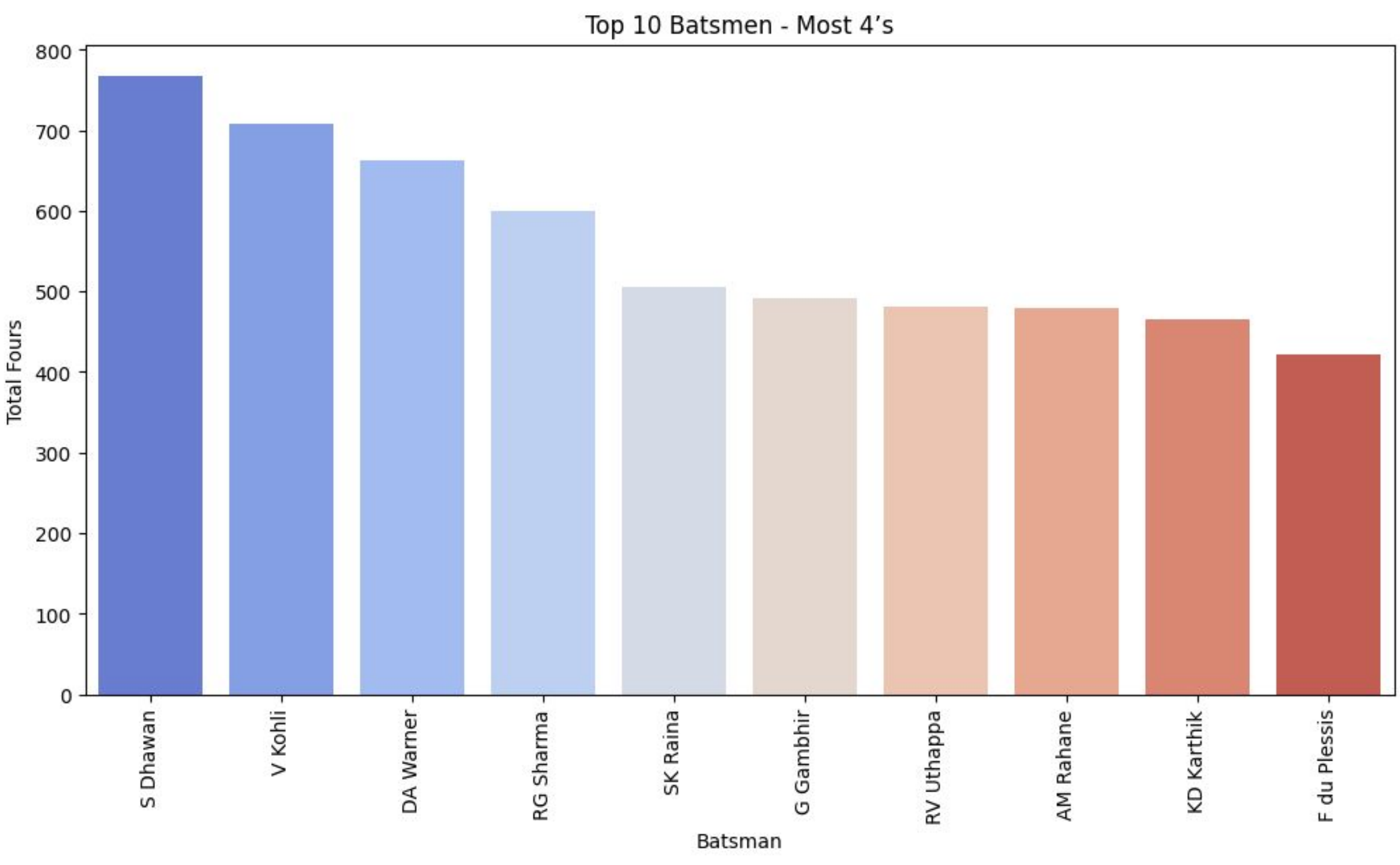

Top Wicket-Takers
Total Wickets
0
25
50
75
100
125
150
175
200
YS Chahal
DJ Bravo
PP Chawla
SP Narine
R Ashwin
B Kumar
SL Malinga
A Mishra
JJ Bumrah
RA Jadeja
UT Yadav
Harbhajan Singh
Rashid Khan
Sandeep Sharma
HV Patel
MM Sharma
Mohammed Shami
AR Patel
TA Boult
R Vinay Kumar
bowler

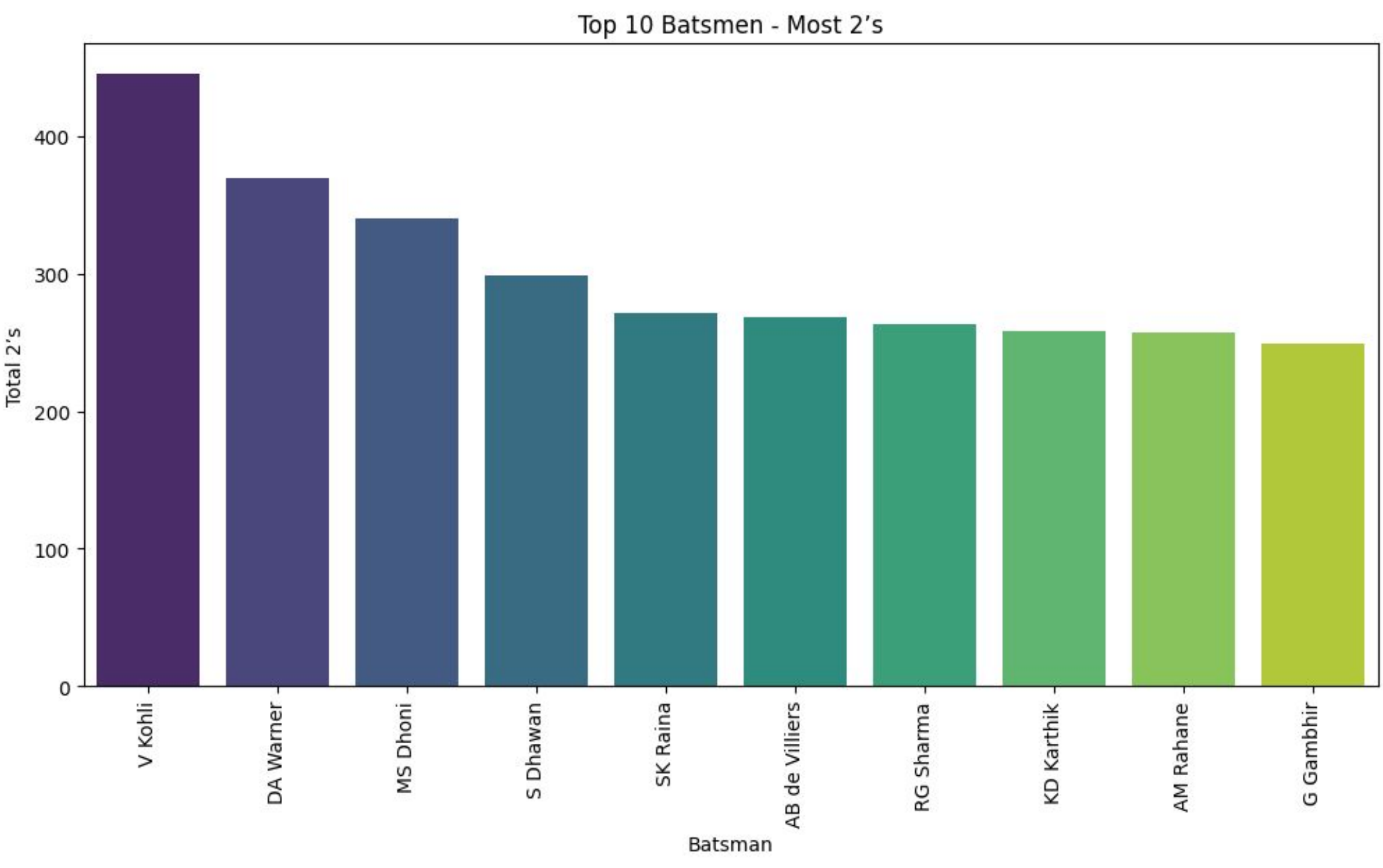

Top Highest Individual Scores
Highest Individual Score
0
25
50
75
100
125
150
175
CH Gayle
BB McCullum
Q de Kock
AB de Villiers
KL Rahul
Shubman Gill
RR Pant
M Vijay
DA Warner
JC Buttler
MP Stoinis
YBK Jaiswal
V Sehwag
PC Valthaty
SV Samson
A Symonds
SR Watson
MEK Hussey
SE Marsh
WP Saha
batter

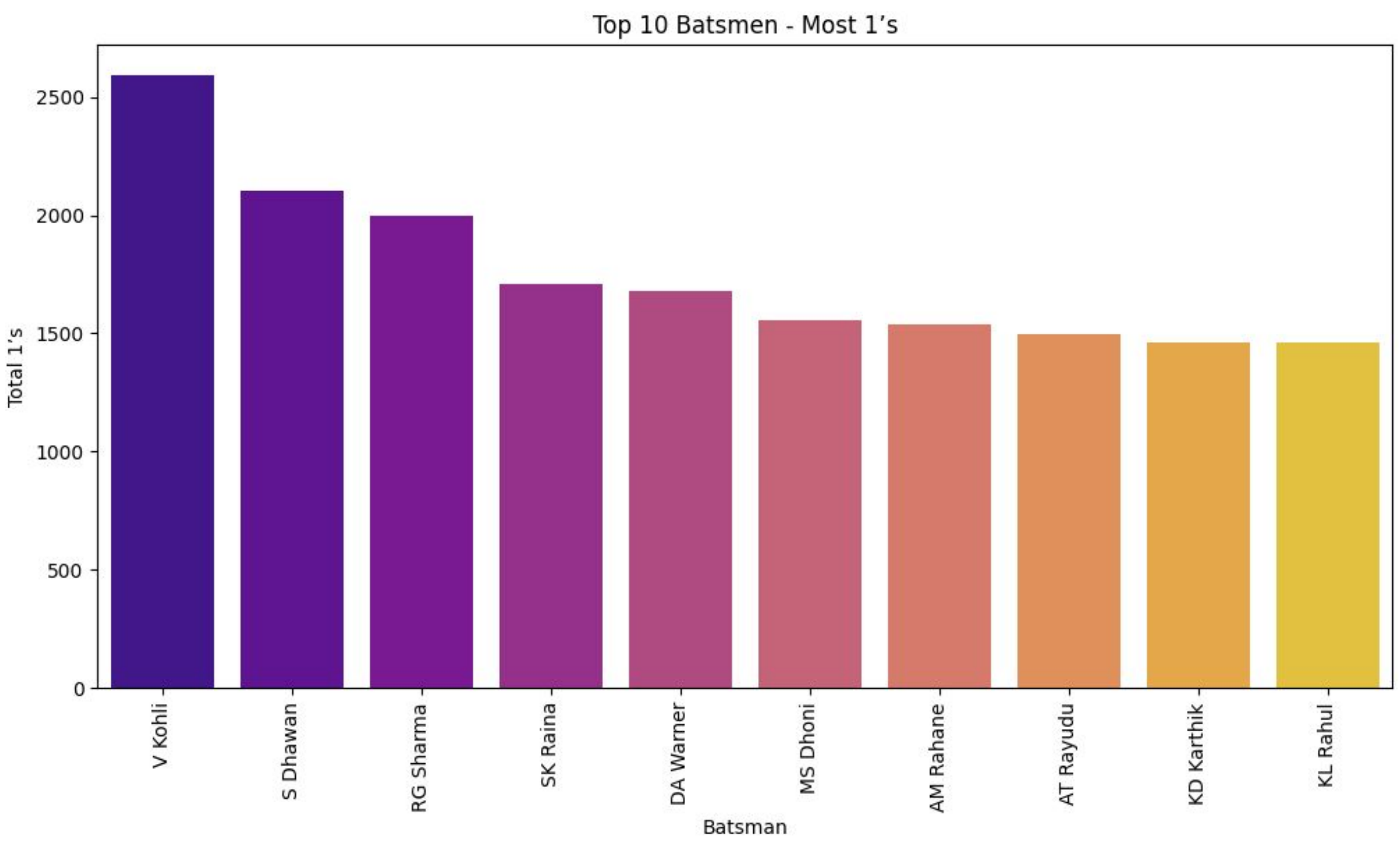

Top Players with Most Man of the Match Awards
Man of the Match Awards
0
5
10
15
20
25
AB de Villiers
CH Gayle
RG Sharma
V Kohli
DA Warner
MS Dhoni
YK Pathan
RA Jadeja
SR Watson
AD Russell
SP Narine
SK Raina
KA Pollard
AM Rahane
JC Buttler
KL Rahul
G Gambhir
A Mishra
S Dhawan
MEK Hussey
player_of_match


K-Means Clustering (Batsman, Bowler, All-Rounder)
cluster
0
1
2
Bowling Economy
Batting Average
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70


Top 10 Batsmen - Most 6’s
Total Sixes
0
50
100
150
200
250
300
350
CH Gayle
RG Sharma
V Kohli
AB de Villiers
MS Dhoni
DA Warner
KA Pollard
AD Russell
SV Samson
SK Raina
Batsman


Top 10 Batsmen - Most 4’s
Total Fours
0
100
200
300
400
500
600
700
800
S Dhawan
V Kohli
DA Warner
RG Sharma
SK Raina
G Gambhir
RV Uthappa
AM Rahane
KD Karthik
F du Plessis
Batsman


Top 10 Batsmen - Most 2’s
Total 2’s
0
100
200
300
400
V Kohli
DA Warner
MS Dhoni
S Dhawan
SK Raina
AB de Villiers
RG Sharma
KD Karthik
AM Rahane
G Gambhir
Batsman


Top 10 Batsmen - Most 1’s
Total 1’s
2500
2000
1500
1000
500
0
V Kohli
S Dhawan
RG Sharma
SK Raina
DA Warner
MS Dhoni
AM Rahane
AT Rayudu
KD Karthik
KL Rahul
Batsman

Seasonal Analysis

- Calculated the average runs per match for each season.
- Identified matches with targets of 200+ runs per season.
- Determined the average score of each team per season.
- Analyzed the runs scored by Orange Cap Holders per season.
- Tracked the wickets taken by Purple Cap Holders per season.
- Identified the top 10 bowlers for each season.

Average runs per match for each season

```
Average runs per match per season:
season
2007/08    309.258621
2009       286.894737
2009/10    314.716667
2011       301.673913
2012       305.310345
2013       297.016667
2014       315.516667
2015       311.067797
2016       319.260870
2017       322.790698
2018       331.683333
2019       323.900000
2020/21    323.600000
2021       312.391304
2022       330.000000
2023       342.800000
2024       361.318182
dtype: float64
```

```
200+ targets per season:
season
2007/08    57
2009       55
2009/10    58
2011       41
2012       56
2013       58
2014       58
2015       56
2016       45
2017       40
2018       59
2019       58
2020/21    59
2021       44
2022       59
2023       59
2024       43
dtype: int64
```

Average team score per season:

team1	Sunrisers Hyderabad	Mumbai Indians	Royal Challengers Bangalore	\
season				
2007/08	342.000000	281.142857	291.714286	
2009	292.400000	288.000000	287.250000	
2009/10	308.857143	330.666667	307.222222	
2011	317.500000	293.500000	289.666667	
2012	292.714286	279.714286	324.000000	
2013	263.571429	319.375000	293.285714	
2014	323.571429	334.142857	322.428571	
2015	291.000000	350.500000	286.375000	
2016	298.571429	305.833333	365.857143	
2017	308.285714	328.000000	285.800000	
2018	301.111111	345.666667	352.166667	
2019	342.500000	315.200000	305.000000	
2020/21	320.166667	348.444444	312.600000	
2021	280.400000	311.142857	293.571429	
2022	388.500000	328.400000	319.750000	
2023	354.142857	333.333333	343.375000	
2024	366.250000	335.500000	NaN	

team1	Kolkata Knight Riders	Kings XI Punjab	Chennai Super Kings	\
season				
2007/08	263.285714	350.428571	321.222222	
2009	276.250000	269.166667	310.272727	
2009/10	309.571429	338.000000	316.600000	
2011	266.000000	325.166667	323.666667	
2012	300.142857	306.142857	299.000000	
2013	275.428571	312.000000	329.222222	
2014	321.625000	308.875000	327.222222	
2015	337.166667	285.857143	304.888889	
2016	317.333333	336.000000	NaN	
2017	313.500000	319.833333	NaN	
2018	350.333333	314.142857	360.333333	
2019	340.444444	351.857143	298.833333	
2020/21	305.800000	340.166667	289.750000	
2021	308.000000	NaN	353.777778	
2022	297.000000	NaN	322.900000	
2023	332.571429	NaN	367.875000	
2024	365.285714	NaN	366.166667	

team1	Rajasthan Royals	Delhi Capitals	Lucknow Super Giants
season			
2007/08	302.285714	318.571429	NaN
2009	NaN	268.000000	NaN
2009/10	300.571429	309.857143	NaN
2011	287.400000	309.400000	NaN
2012	320.714286	319.555556	NaN
2013	285.625000	286.857143	NaN
2014	290.714286	292.285714	NaN
2015	351.000000	285.142857	NaN
2016	NaN	304.333333	NaN
2017	NaN	341.333333	NaN
2018	299.500000	345.333333	NaN
2019	320.800000	312.166667	NaN
2020/21	348.500000	325.545455	NaN
2021	317.166667	301.142857	NaN
2022	340.769231	350.500000	343.428571
2023	355.428571	301.500000	291.800000
2024	351.250000	406.500000	348.333333

team1	Gujarat Titans
season	
2007/08	NaN
2009	NaN
2009/10	NaN
2011	NaN
2012	NaN
2013	NaN
2014	NaN
2015	NaN
2016	306.750000
2017	358.166667
2018	NaN
2019	NaN
2020/21	NaN
2021	NaN
2022	312.166667
2023	364.750000
2024	312.666667

Orange Cap Holders per season:

	season	batter	batsman_runs
115	2007/08	SE Marsh	616
229	2009	ML Hayden	572
446	2009/10	SR Tendulkar	618
502	2011	CH Gayle	608
684	2012	CH Gayle	733
910	2013	MEK Hussey	733
1088	2014	RV Uthappa	660
1148	2015	DA Warner	562
1383	2016	V Kohli	973
1422	2017	DA Warner	641
1594	2018	KS Williamson	735
1694	2019	DA Warner	692
1866	2020/21	KL Rahul	676
2051	2021	RD Gaikwad	635
2144	2022	JC Buttler	863
2423	2023	Shubman Gill	890
2606	2024	V Kohli	741

Purple Cap Holders per season:

	season	bowler	wickets
75	2007/08	Sohail Tanvir	22
152	2009	RP Singh	23
241	2009/10	PP Ojha	21
367	2011	SL Malinga	28
437	2012	M Morkel	25
509	2013	DJ Bravo	32
638	2014	MM Sharma	23
694	2015	DJ Bravo	26
773	2016	B Kumar	23
860	2017	B Kumar	26
942	2018	AJ Tye	24
1048	2019	Imran Tahir	26
1141	2020/21	K Rabada	32
1211	2021	HV Patel	32
1379	2022	YS Chahal	27
1448	2023	Mohammed Shami	28
1516	2024	HV Patel	24

Top 10 bowlers per season:

	season	bowler	wickets
77	2007/08	Sohail Tanvir	24
30	2007/08	IK Pathan	20
32	2007/08	JA Morkel	20
70	2007/08	SK Warne	20
72	2007/08	SR Watson	20
...	...	...	...
1530	2024	Arshdeep Singh	20
1544	2024	Harshit Rana	20
1566	2024	MA Starc	20
1610	2024	T Natarajan	20
1522	2024	AD Russell	19

Dataset Preprocessing

In this study, the dataset underwent preprocessing to ensure it was ready for model training. Initially, the features (X) and target variable (y) were separated. The dataset was then split into training and testing subsets using an 80-20 ratio to allow for model evaluation on unseen data. The splitting process was conducted using `train_test_split` from the `sklearn.model_selection` module, ensuring a randomized but reproducible split by setting `random_state=1`.

To handle categorical variables such as `batting_team`, `bowling_team`, and `city`, one-hot encoding was applied. This transformation was performed using `OneHotEncoder` within a `ColumnTransformer`, ensuring that categorical variables were encoded into numerical values while preserving the remaining features.

Model Architecture and Training Methodology

The machine learning model was implemented using a pipeline approach to streamline preprocessing and training. The pipeline was constructed with two main steps:

1. **Data Transformation** - The `ColumnTransformer` applied one-hot encoding to categorical variables.
2. **Model Training** - A `LogisticRegression` model with the `liblinear` solver was used as the classifier.

The model was trained using the `fit` function on the training dataset (`X_train`, `y_train`). Additionally, the trained model was saved using the `pickle` module to facilitate future reuse without retraining. Later, the saved model was reloaded, and predictions were generated using `predict_proba` for classification probability outputs.

Performance Evaluation

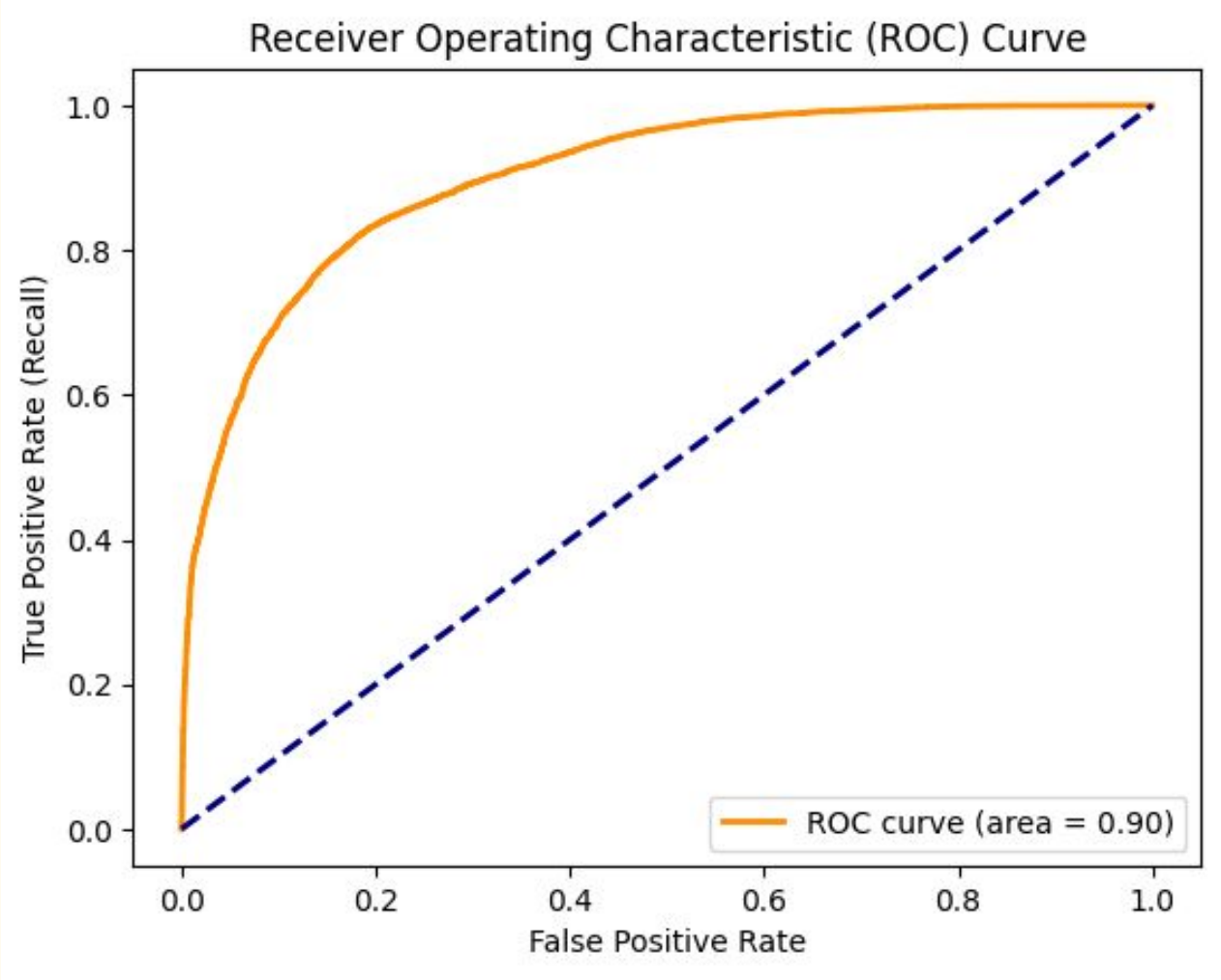
The trained model's performance was assessed using various evaluation metrics:

- **Accuracy**: The proportion of correctly classified instances.
- **Recall**: The model's ability to identify positive instances correctly.
- **F1-score**: A balance between precision and recall.
- **ROC-AUC Score**: A measure of the model’s ability to distinguish between classes.

The model's predictions were compared with actual test labels (`y_test`) using `accuracy_score`, `recall_score`, and `f1_score` from `sklearn.metrics`. Additionally, a Receiver Operating Characteristic (ROC) curve was plotted using `matplotlib.pyplot`, showing the trade-off between true positive and false positive rates. The area under the curve (AUC) was also computed to quantify the model’s discriminatory power.

ROC Curve

The ROC (Receiver Operating Characteristic) curve provides key insights into the classification model's performance, particularly in terms of accuracy, precision, recall, and overall discrimination ability.



Results and Discussion

Analysis of Results based on ROC Curve:

1. **True Positive Rate (Recall)**:
    - The y-axis represents the **True Positive Rate (Recall)**, which indicates how well the model correctly identifies positive cases.
    - A higher recall means the model captures more actual positives.
    - In this case, the curve shows a strong recall, especially at lower false positive rates.

False Positive Rate (1 - Specificity):

- The x-axis represents the **False Positive Rate (FPR)**, indicating how often the model incorrectly classifies negatives as positives.

- A lower FPR is preferred, as it means fewer false alarms.

- Here, the model maintains a relatively low FPR for most thresholds, indicating good performance.

Area Under the Curve (AUC = 0.90):

- The AUC score of **0.90** suggests that the model is highly effective at distinguishing between the two classes.

- AUC values closer to **1.0** indicate a better-performing model, while a score of **0.5** would imply random guessing.

Accuracy vs Precision vs Recall:

- **Accuracy**: The overall correctness of predictions. Since the ROC curve is well above the diagonal, the accuracy is likely high.

- **Precision**: Not directly represented in the ROC curve, but can be inferred. Precision depends on the ratio of true positives to all predicted positives (true positives + false positives). If the curve had a steeper drop-off, it might indicate more false positives, lowering precision.

- **Recall**: Directly represented on the y-axis. The model captures a large proportion of actual positives, suggesting good recall.

The metrics of evaluation showed the effectiveness of the model in predicting the target variable. The accuracy score gave a general measure of correct predictions, whereas the recall and F1-score made sure the model was not biased towards any class. The ROC-AUC score also validated the model's capability to separate classes well.

Despite achieving a reasonable performance, potential improvements could be made, such as:

- Implementing hyperparameter tuning for the logistic regression model.
- Experimenting with alternative classifiers like Random Forest or Support Vector Machines.
- Enhancing feature selection to improve model generalization.

Overall, the implemented approach demonstrated a structured and effective methodology for training and evaluating a classification model, with room for further enhancements to optimize performance.

Conclusion

The results of our predictive modeling offer valuable insights into the expected performance of teams and key players in the upcoming season. While the model provides a data-driven approach to forecasting match results, it is important to acknowledge potential limitations, such as the influence of unpredictable game-day factors, player form fluctuations, and external conditions like weather and injuries. Nonetheless, this study highlights the practical application of data science and machine learning in sports analytics, paving the way for further research and refinements in predictive modeling techniques for cricket and other sports.

By integrating historical data with advanced analytics, this research contributes to a deeper understanding of IPL dynamics and enhances the predictive capabilities for future tournaments. The methodologies employed can be extended and improved upon in subsequent studies, offering a foundation for more sophisticated predictive frameworks in sports analytics.

References

- *"Data-Driven Cricket: A Machine Learning Approach to IPL Score Prognostication"* by S. Lalitha, Anurag Das, Prasanth Ayitapu, RS Nair, Satvik Raghav

- *"Predicting Results of Indian Premier League T-20 Matches using Machine Learning"* by Shilpi Agrawal, Suraj Pal Singh, Jayash Kr. Sharma

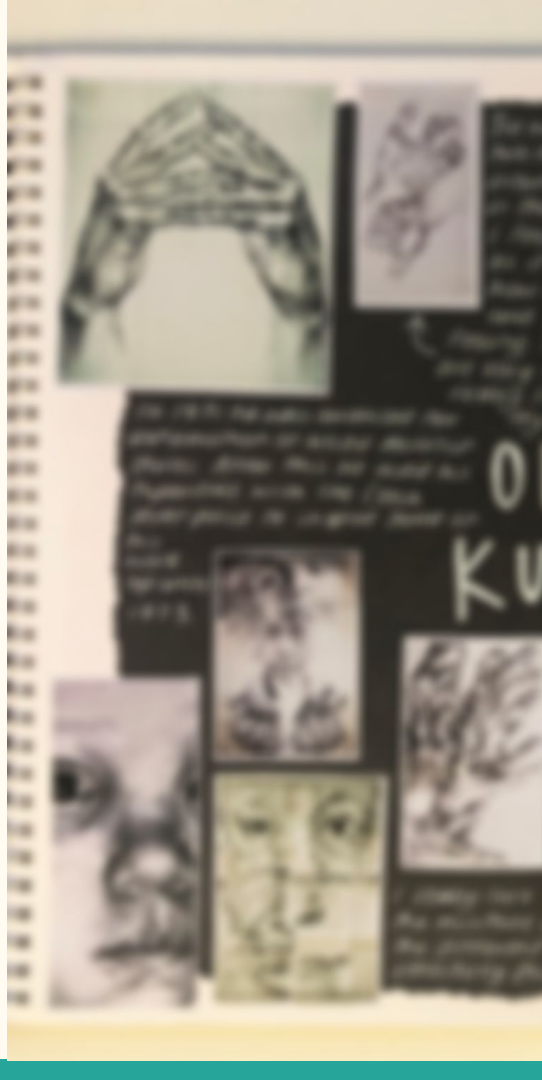
- CampusX on youtube.com

Research Article Summarization

Using Advanced NLP Techniques

Table of Contents

1. Introduction
2. Dataset Preprocessing
3. Model Architecture and Training Methodology
4. Performance evaluation
5. Results and Discussion
6. Conclusion
7. References



Introduction

With the rapid increase in scientific articles, scientists are no longer able to cope with the immense amount of literature. Therefore, we focused on creating an extractive-abstractive combined model based on cutting-edge Large Language Models (LLMs) to produce succinct, readable, and meaningful summaries of scientific research papers. Structured scientific text was our point of interest without losing readability and major findings.

Dataset Preprocessing

We started by gathered several research article dataset collections, such as:

1. **CompScholar Dataset**
2. **PubMed Dataset** (millions of biomedical research articles with annotated abstracts)
3. **arXiv Dataset** (annotated documents from a variety of scientific domains)

Each dataset had full-text research articles, abstracts, metadata (titles, keywords, citations, authors), and summarization labels.

We conducted various preprocessing operations to ready the data:

1. **Tokenization**: We tokenized the text with Hugging Face's tokenizer to represent words as numbers.
2. **Cleaning**: We extracted unnecessary special characters and redundant data.
3. **Text Normalization**: We did lowercasing, stemming, and lemmatization wherever needed.
4. **Dataset Splitting**: We split the data into training, validation, and test sets.
5. **Handling Missing Data**: We identified and filled missing fields in research papers.

Model Architecture and Training Methodology

We used transformer-based models and fine-tuned pre-trained models such as PEGASUS, BART, and Longformer. The steps included:

1. **Tokenization and Data Preparation**: We prepared input text using Hugging Face's auto-tokenizer.
2. **Encoder-Decoder Architecture**:
- The encoder took full-length research papers as input.
- The decoder produced short summaries maintaining essential insights.
3. **Loss Function**: We employed cross-entropy loss to optimize summarization efficiency.

4. **Fine-Tuning**: We fine-tuned the pre-trained models on the CompScholar and PubMed datasets.
5. **Batch Processing**: We processed tokenized inputs in batches for computational efficiency.
6. **Optimization**: We used learning rate scheduling and gradient clipping to prevent overfitting.
7. **Checkpointing**: We saved model weights periodically for evaluation and further fine-tuning.

Conclusion

Through this project, we were able to develop and test a hybrid summarization model that can effectively summarize research papers efficiently. Although models such as PEGASUS and BART were good, we were able to identify areas for improvement, especially in domain-specific summarization and enhancing knowledge retention. Optimization of computational efficiency and the development of reinforcement learning-based summarization methods are some of the areas of future work.

References

1. PubMed Datasets

2. arXiv Datasets

3. Hugging Face Transformers Documentation

4. Recent Research Papers on Summarization Techniques

5. Youtube Transcripts (YouTube Channel : Learning Logic)