Statistics is All You Need: IPL Data Analysis and 2025 Winner Prediction – The Game Behind the Game!

Team: why
Arup Sankar Reang, Joydip Das ,Puyush Gupta

March 2025

Abstract

Team 'Why' submission for Brain Dead 2025 hackathon, hosted by Indian Institute of Engineering Science and Technology, Shibpur (IIEST), this is the report for the first problem statement and a summary of the approach we took , we analysed the data and created new features, and then fit the data to models like XGBoost and CatBoost, our highest F1-score is **0.8489**

1 Introduction

Cricket is the most popular sport in India. There are various formats of this game and the most loved one is the Indian Premier League (IPL). This professional Twenty20 cricket league in India gets contested during March or April and May of every year by eight teams representing ten different cities on India. The league was founded by the Board of Control for Cricket in India (BCCI) in 2008. The IPL has an exclusive window in ICC Future Tours Programme. It is the most-attended cricket league in the world. Currently, it's the 18th season of IPL.

You have to perform a comprehensive analysis of IPL data from its inception through to the most recent season in 2024, aimed at uncovering key insights, trends, and patterns. It should consists of data collection, preprocessing, and exploratory data analysis (EDA) to visualize metrics such as win rates, player performance, and team statistics. The analysis includes statistical insights to identify significant factors influencing match outcomes. You may use pandas and NumPy for data manipulation and matplotlib and seaborn for data visualization.

Also, try to develop an ensemble model, combining different classifier models (one such example is the ensembling of classifiers like Random Forest and XGBoost) for predicting the winner of the 2025 IPL season. It should explain the model's features, training, validation, and performance evaluation. Additionally, you can explore experimenting with neural networks. The results section must present the model's predictions for the 2025 season, and discussion regarding the potential strengths and limitations, and should provide insights into the predicted performance of teams and key players. The primary objective is to use historical IPL data to build a predictive model for future match prediction outcomes, demonstrating the application of advanced machine learning techniques to sports data.

2 Methodology

2.1 Data Cleaning and Feature Engineering

Ensure that no missing values or outliers exist in the dataset. Handle potential issues that could impact insights and predictions.

handling null values

null values in matches.csv

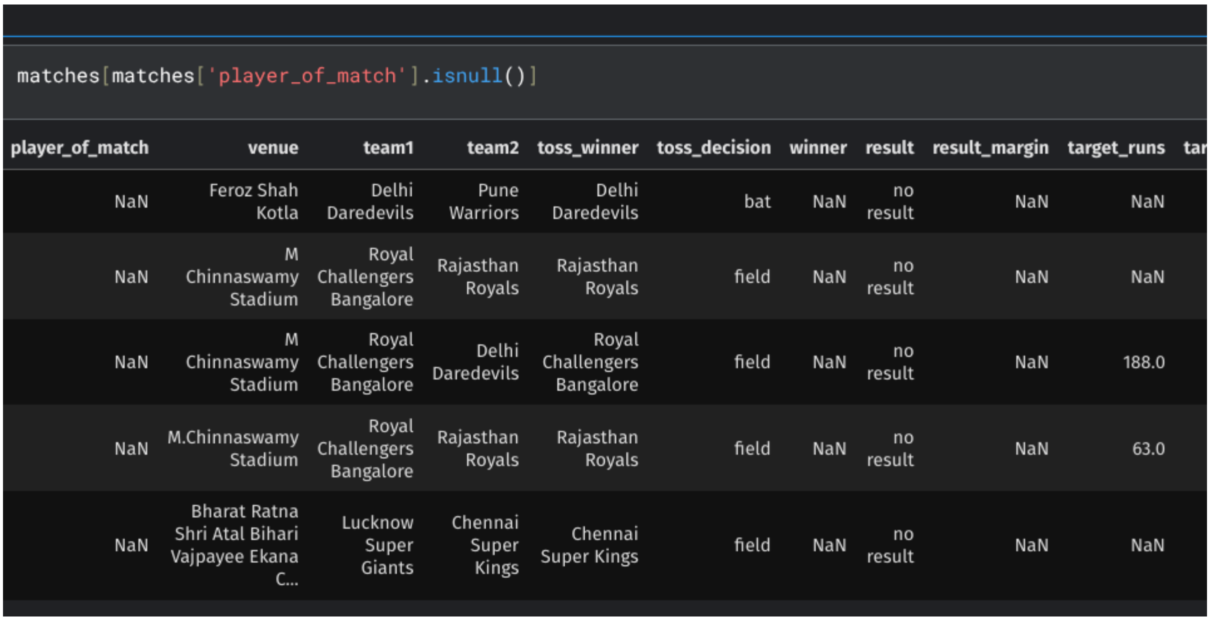
column id	missing values
city	51
player_of_match	5
winner	5
result_margin	19
target_runs	3
target_overs	3
method	1074

Table 1: missing values in matches.csv

city: city had 51 missing values but it was the easiest to fill, we used a function **fill_city()**

algorithm
1. Iterate through null value rows
2. Save the venue name
3. get all the rows with that venue name
4. find row where city column is not NaN
5. fill current row with the city found

Table 2: city null value fill



```
matches[matches['player_of_match'].isnull()]
```

player_of_match	venue	team1	team2	toss_winner	toss_decision	winner	result	result_margin	target_runs	tar
NaN	Feroz Shah Kotla	Delhi Daredevils	Pune Warriors	Delhi Daredevils	bat	NaN	no result	NaN	NaN	
NaN	M Chinnaswamy Stadium	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	NaN	no result	NaN	NaN	
NaN	M Chinnaswamy Stadium	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	field	NaN	no result	NaN	188.0	
NaN	M.Chinnaswamy Stadium	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	NaN	no result	NaN	63.0	
NaN	Bharat Ratna Shri Atal Bihari Vajpayee Ekana C...	Lucknow Super Giants	Chennai Super Kings	Chennai Super Kings	field	NaN	no result	NaN	NaN	

Figure 1: nan rows from player_of_match

player_of_match, winner, as a lot of nan values were concentrated in these rows so we decided to research these mathes, [4][5] these matches are called off matches and since they are outliers we decided to remove them
After removing these 5 rows only null values remaining are result_margin:14 and method:1069

Result margin becomes nan when the match result is a tie, and method is the check that Duckworth-Lewis method was applied, these are important factors so these should be filled with static values, result_margin nan can be 0, and method nan can be 'Not Applicable'

data standardization

standardized data such as renaming "Delhi Daredevils" with "Delhi Capitals", and formatting date and seasons

handling deliveries null values

null values in deliveries.csv

column id	missing values
extras_type	246795
player_dismissed	247970
dismissal_kind	247970
fielder	251566

Table 3: missing values in deliveries.csv

extras_type: If missing, it means the delivery was a normal legal delivery (not an extra).
So it was filled using "Normal". **player_dismissed, dismissal_kind, and fielder**:

player_dismissed was filled with "None", dismissal_kind with "Not Out", and fielder with "None"

data standardization

standardized data such as renaming "Delhi Daredevils" with "Delhi Capitals".

2.2 Exploratory Data Analysis (EDA)

Performed the following analyses based on the IPL 2008-2024 Dataset:

Team Performance:

Plot Matches Played and Winning Percentages

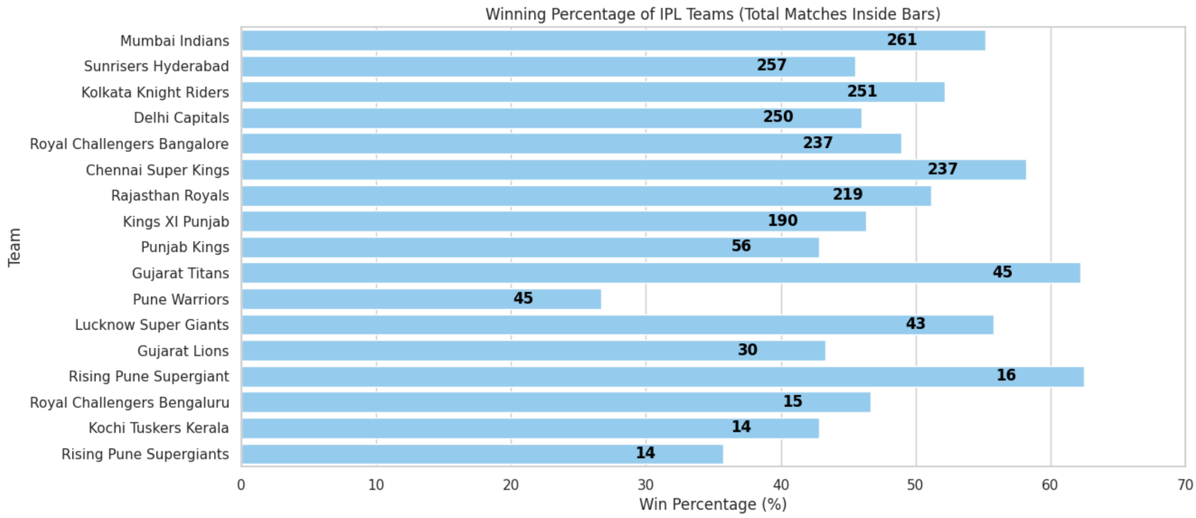


Figure 2: Team win percentages with total matches played

Win Percentage: the graph is sorted in descending order of total matches played, this is done as even if a team has higher win percentage if the percentage is based on less matches that data has less value than the percentage derived from 200+ matches.

Plot Run Rate and Economy Rate (as a bowling side)

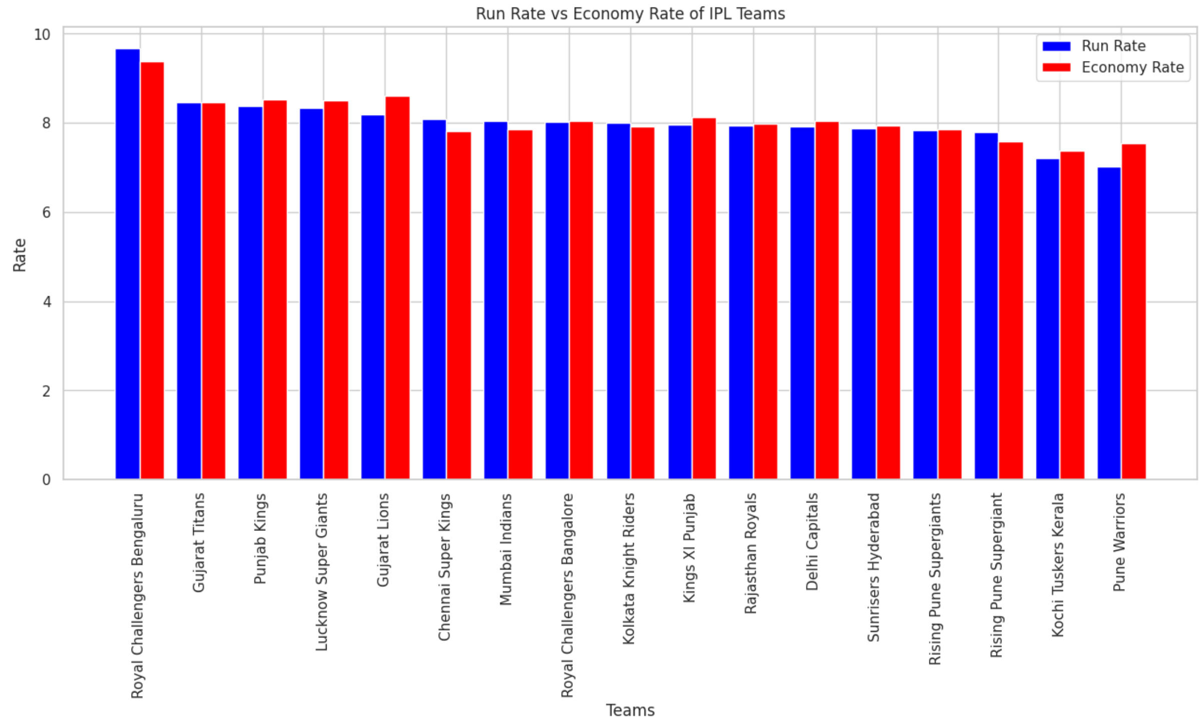


Figure 3: run rate and economy rate graph

what does run rate and economy rate mean:

- If a team's $RR \approx ER$, they are not significantly outscoring their opponents.
- Example: RR = 9.50, ER = 9.40 → The team plays in high-scoring matches but doesn't necessarily dominate.
- Example: RR = 7.00, ER = 7.10 → The team plays in lower-scoring games, possibly on bowling-friendly pitches.
- If RR ¿ ER → The team tends to win more because they score faster than they concede.
- If RR ¡ ER → The team struggles as they concede more runs than they score.
- If $RR \approx ER$ → The team is consistent but not dominant.

Plot Highest and Lowest Scores

top 10 teams from each category arranged datewise

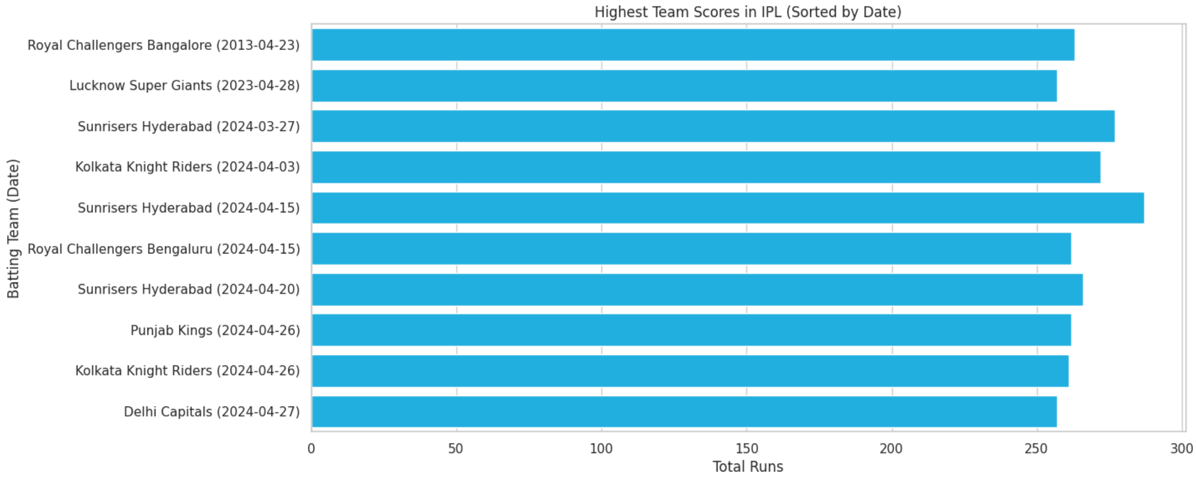


Figure 4: Top 10 Highest scores

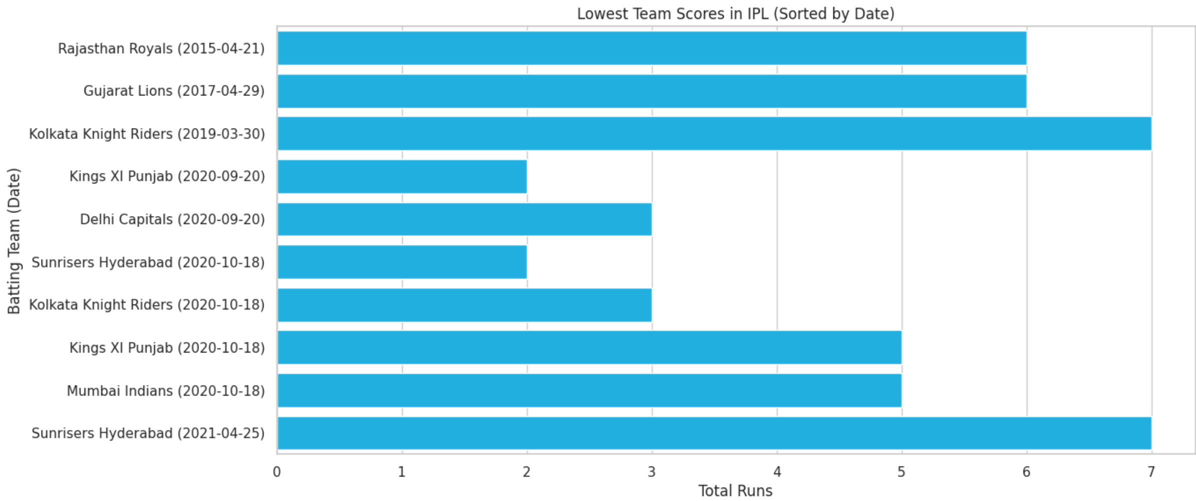


Figure 5: Top 10 lowest scores

Plot Total 4s and 6s plotting the total 4's and 6's a team made, sorted by 6's in fig-6

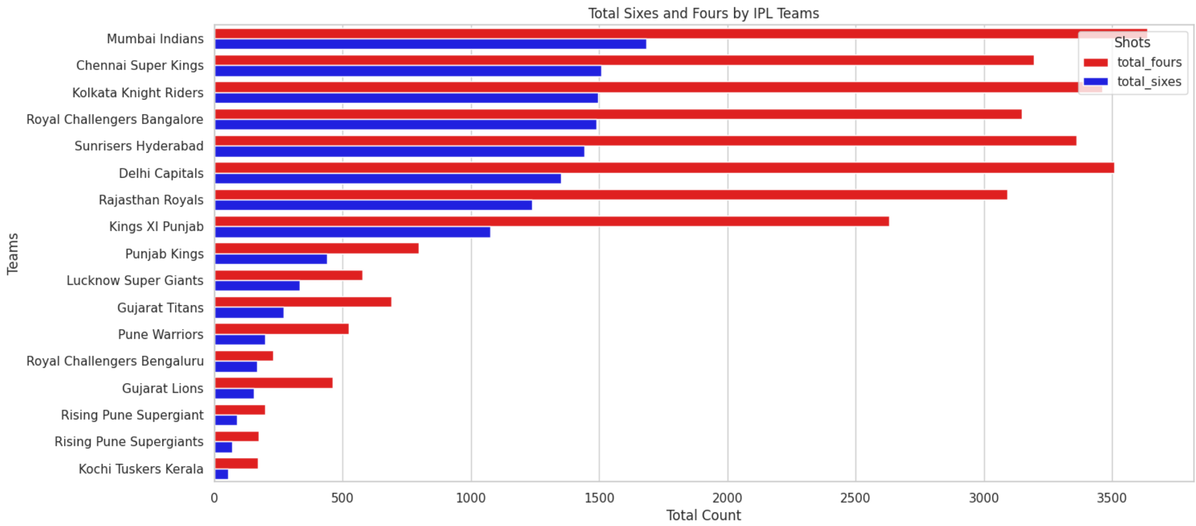


Figure 6: fours and sixes

Plot Average Powerplay and Death Overs Score

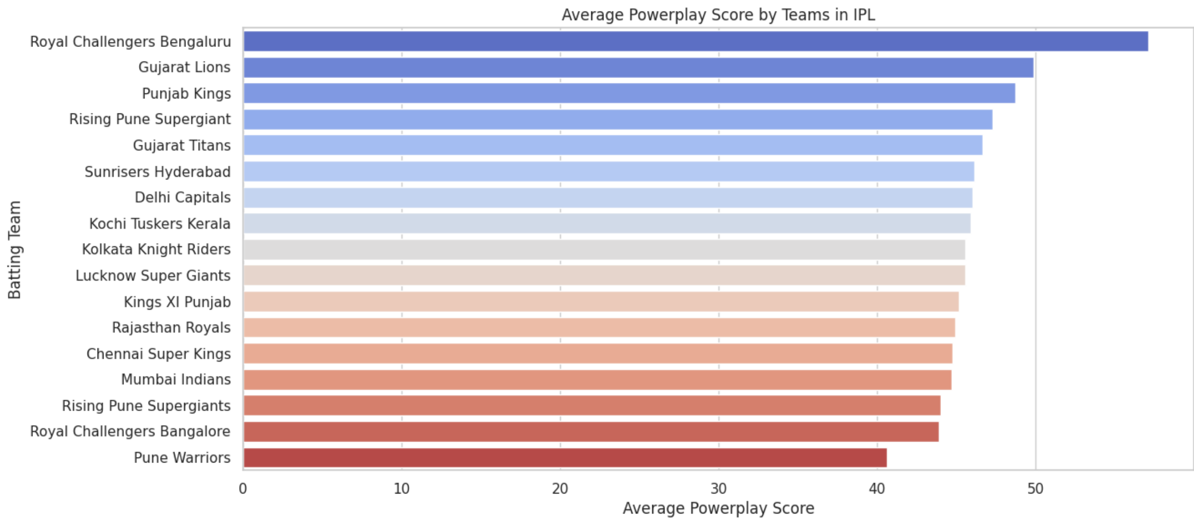


Figure 7: Average powerplay score

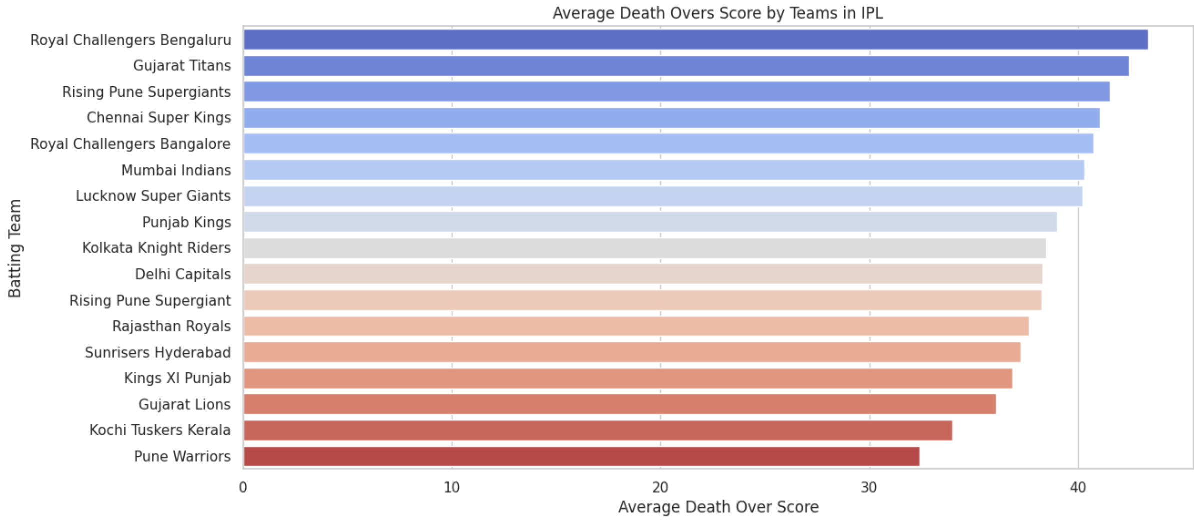


Figure 8: Average Death Overs Score

Average run per over of each team (as a batting side)

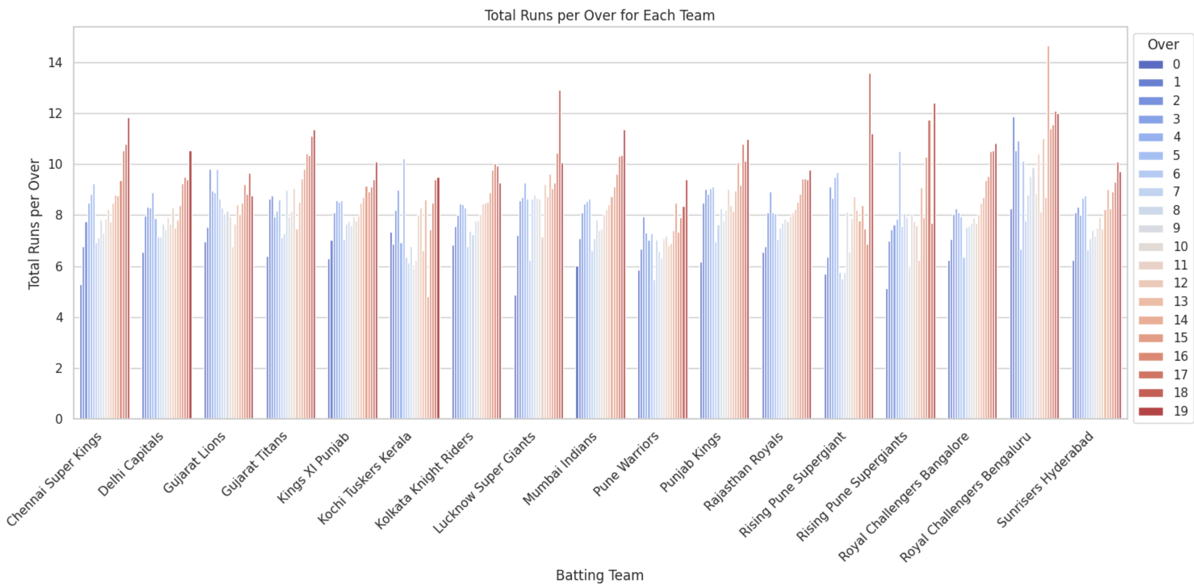


Figure 9: avg run per over of each team

Powerplay Analysis

The Powerplay (Overs 1-6) is crucial in T20 cricket, as it sets the tone for the innings. A strong start can boost a team's confidence, while a weak powerplay can put them on the back foot.

What Defines a Good or Bad Powerplay Score?

- **Excellent:** 50+ runs → Strong aggressive start, taking advantage of fielding restrictions.
- **Good:** 45-50 runs → Decent start, balanced approach.
- **Average:** 40-45 runs → Conservative start, focusing on wicket preservation.

- **Poor:** Below 40 runs → Slow start, could indicate early wickets or defensive play.

Team Performance in Powerplay

Powerplay Analysis

Best Performers (Explosive Starts)

Royal Challengers Bengaluru (57.13) → Best Powerplay Team

- Significantly higher than other teams.
- Indicates an aggressive batting approach with power hitters at the top.

Gujarat Lions (49.89), Punjab Kings (48.73), Rising Pune Supergiant (47.26), Sunrisers Hyderabad (47.19)

- Strong starts, likely due to aggressive openers.
- Above-average Powerplay scores.

Average Performers

Gujarat Titans (46.63) to Mumbai Indians (44.71)

- These teams have decent but not explosive powerplay performances.
- Likely balance aggression and wicket preservation.

Weak Powerplay Teams

Royal Challengers Bangalore (43.92), Deccan Chargers (43.64), Pune Warriors (40.61) → Lowest Powerplay Scores

- Struggle to score aggressively in the first 6 overs.

Player Performance:

Get the top 20 run-scorers
in fig-10 shows the top scorers and in the bars you have their stats, match played, score on avg, strike rate, fours and sixes

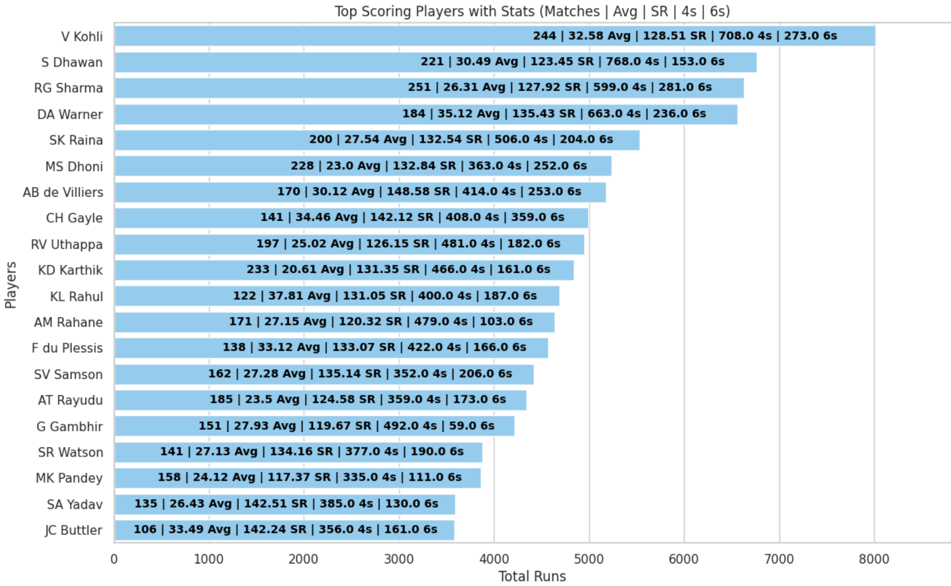


Figure 10: top 20 run scorers

Plot Batting Average vs Batting Strike Rate for the top 20 run-scorers

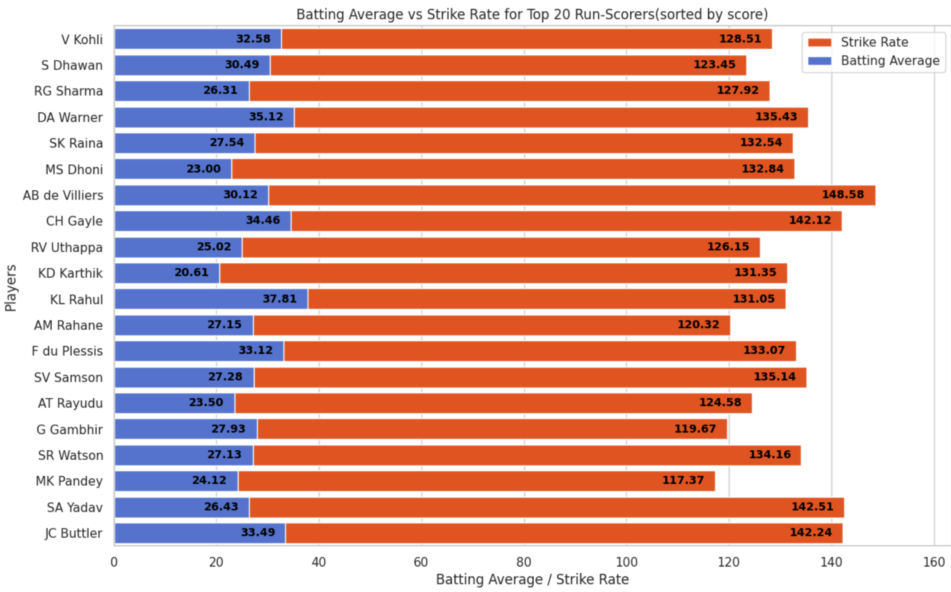


Figure 11: batting avg vs batting strike rate

Find Highest Average and Strike Rate for players with >50 matches

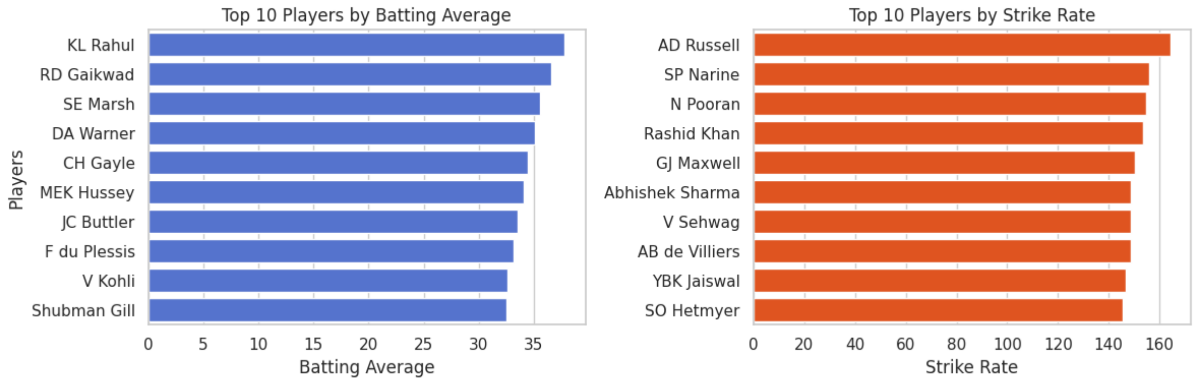


Figure 12: top 10 player with highest bating avg and strike rate

Plot top wicket-takers

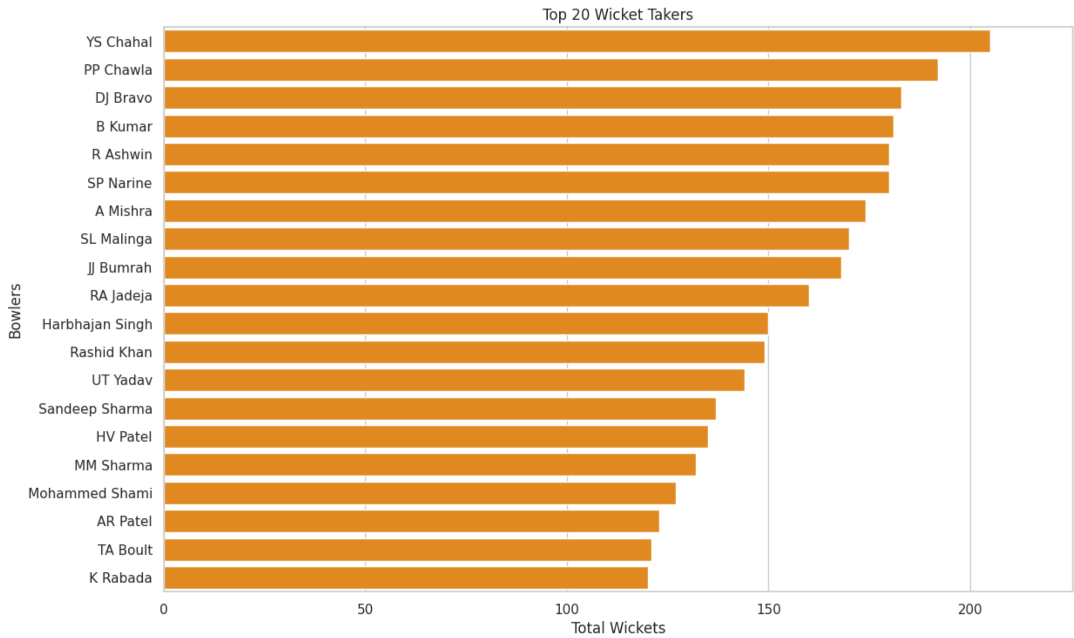


Figure 13: top 20 wicker takers

Plot top highest individual scores

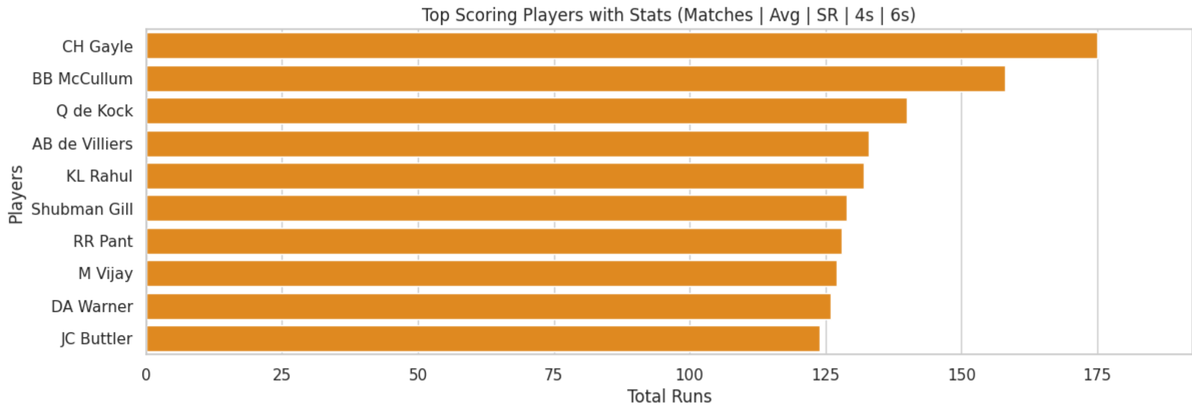


Figure 14: top individual scores by a player

Man of the Match Count Analysis

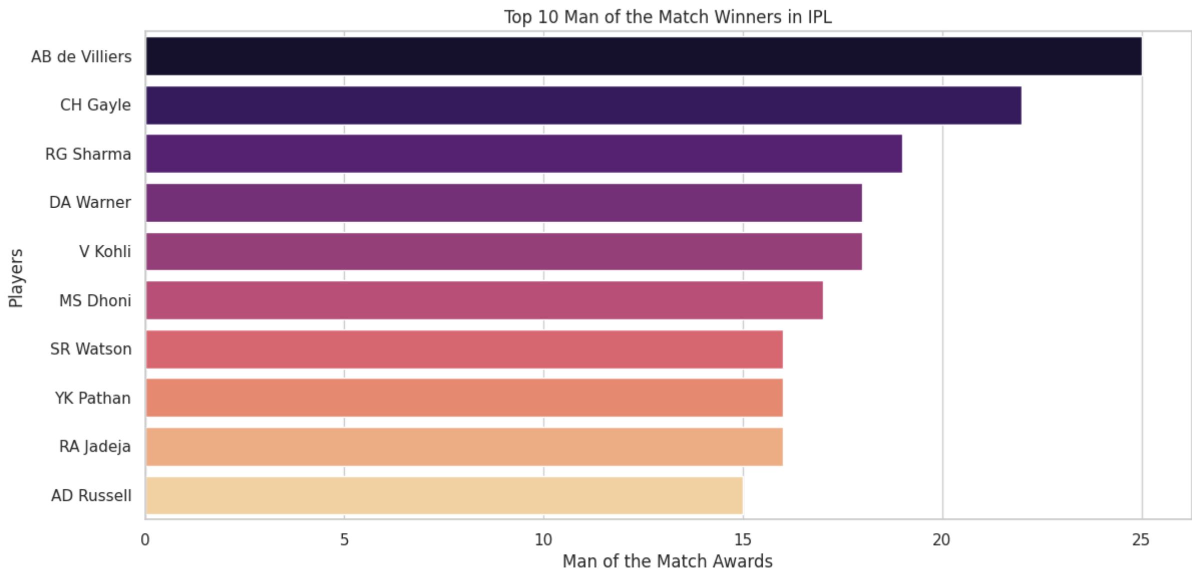


Figure 15: Top 10 Man of the Match Winners in IPL

Use K-Means Clustering to plot Batting Average vs Bowling Economy Rate for number of clusters = 3 (Batsman, Bowler, All Rounder)

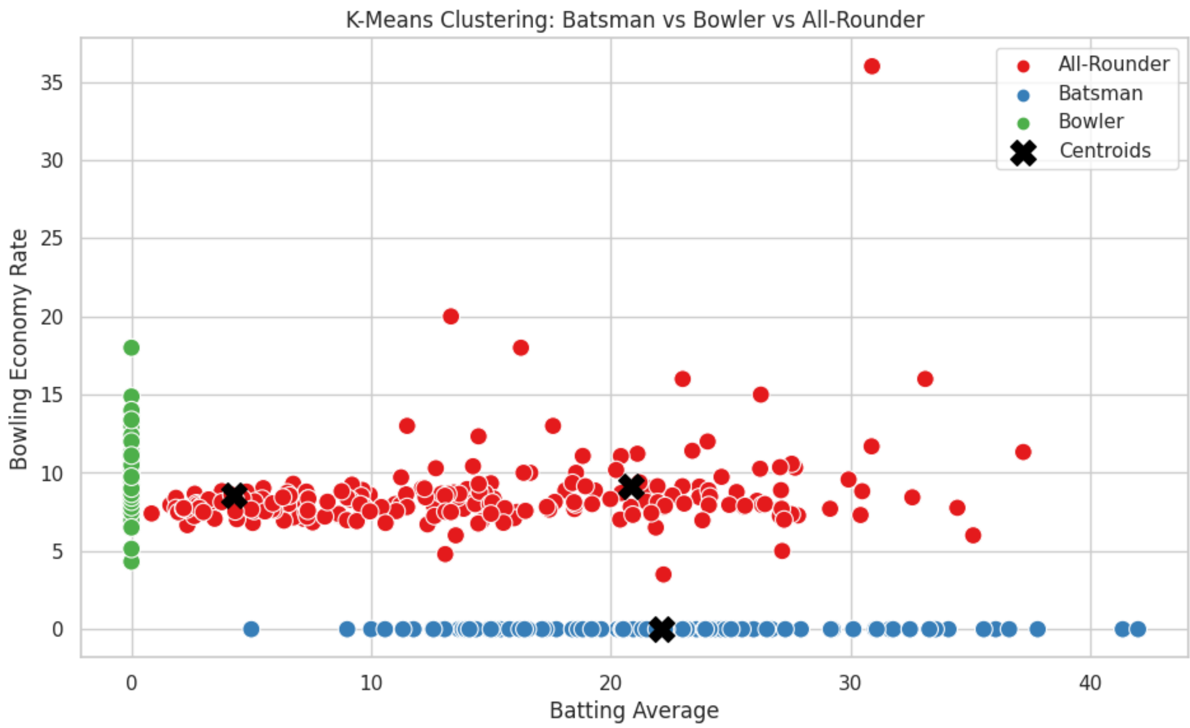


Figure 16: clusters from K-means clustering

Identify Top 10 Batsmen in each run category:

Top 10 Singles Scorers	Singles
V Kohli	2591
S Dhawan	2102
RG Sharma	1996
SK Raina	1708
DA Warner	1682
MS Dhoni	1554
AM Rahane	1537
AT Rayudu	1495
KL Rahul	1464
KD Karthik	1464

Table 4: Top 1's scorer

Top 10 Double Scorers	Doubles
V Kohli	445
DA Warner	370
MS Dhoni	340
S Dhawan	299
SK Raina	271
AB de Villiers	268
RG Sharma	263
KD Karthik	258
AM Rahane	257
G Gambhir	249

Table 5: Top 2's scorer

Top 10 Fours Hitters	Fours
S Dhawan	768
V Kohli	708
DA Warner	663
RG Sharma	599
SK Raina	506
G Gambhir	492
RV Uthappa	481
AM Rahane	479
KD Karthik	466
F du Plessis	422

Table 6: Top 4's scorer

Top 10 Sixes Hitters	Sixes
CH Gayle	359
RG Sharma	281
V Kohli	273
AB de Villiers	253
MS Dhoni	252
DA Warner	236
KA Pollard	224
AD Russell	209
SV Samson	206
SK Raina	204

Table 7: Top 6's scorer

Seasonal Analysis:

Calculate average runs per match per season

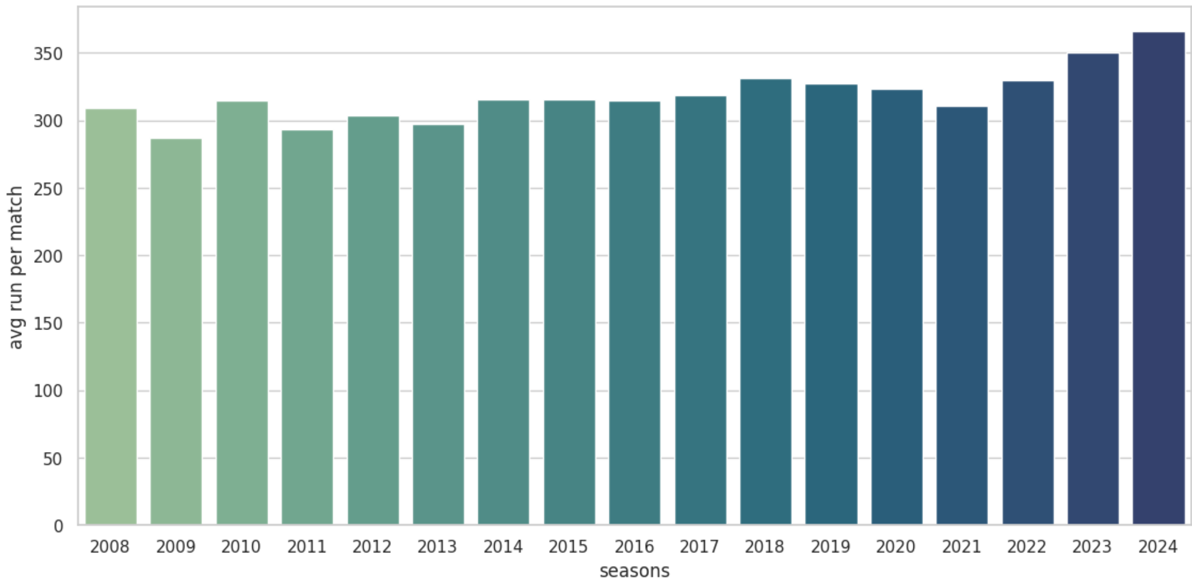


Figure 17: Average runs per match per season

Identify targets of 200+ runs per season

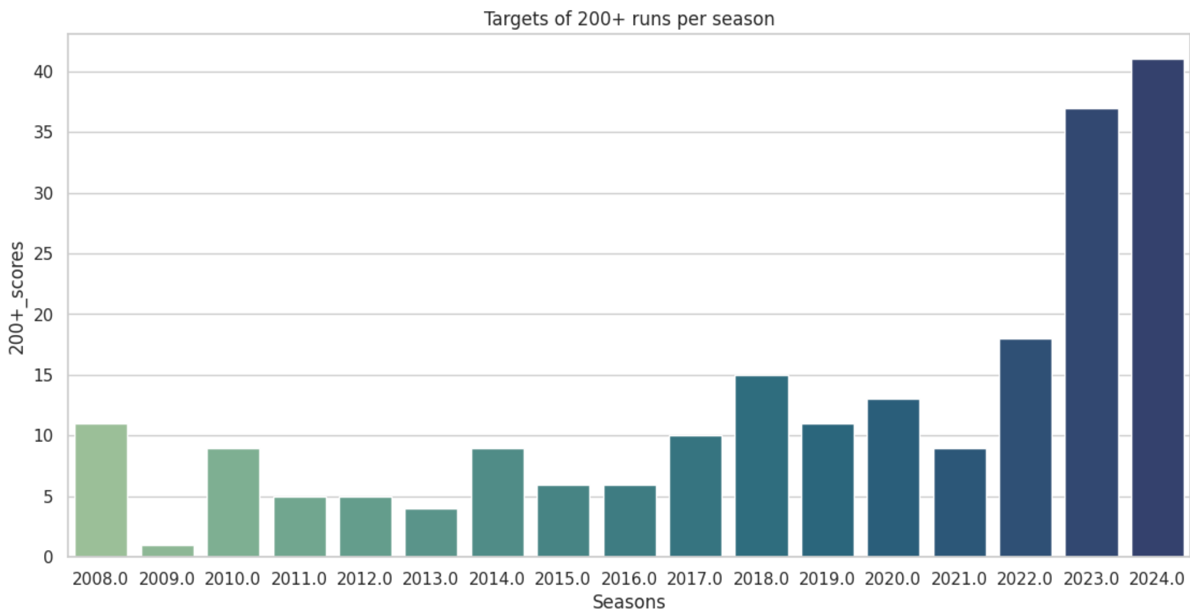


Figure 18: Targets of 200+ runs per season

Find the average score of each team per season

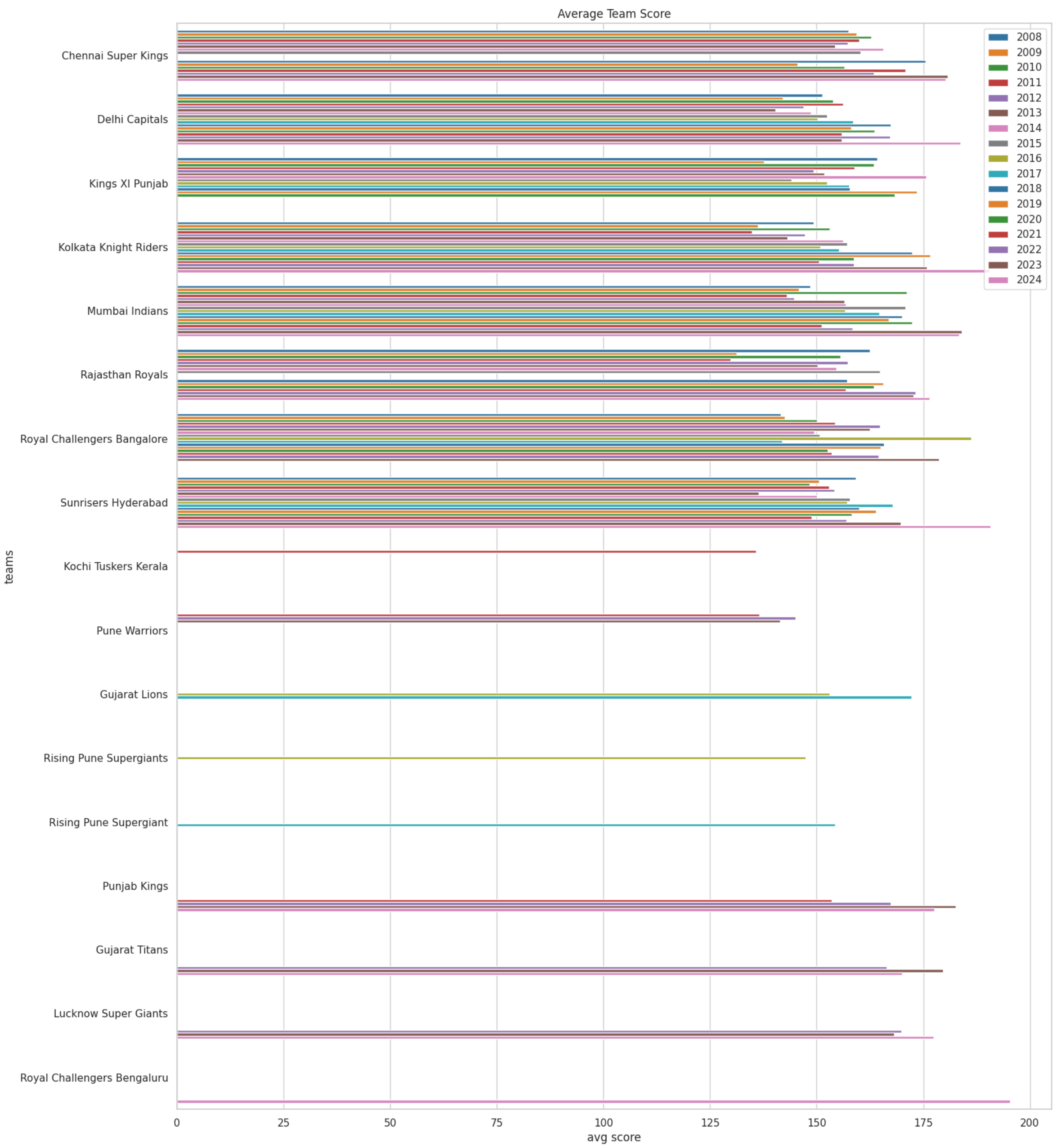


Figure 19: Average score of each team per season

Analyze runs of Orange Cap Holders per season

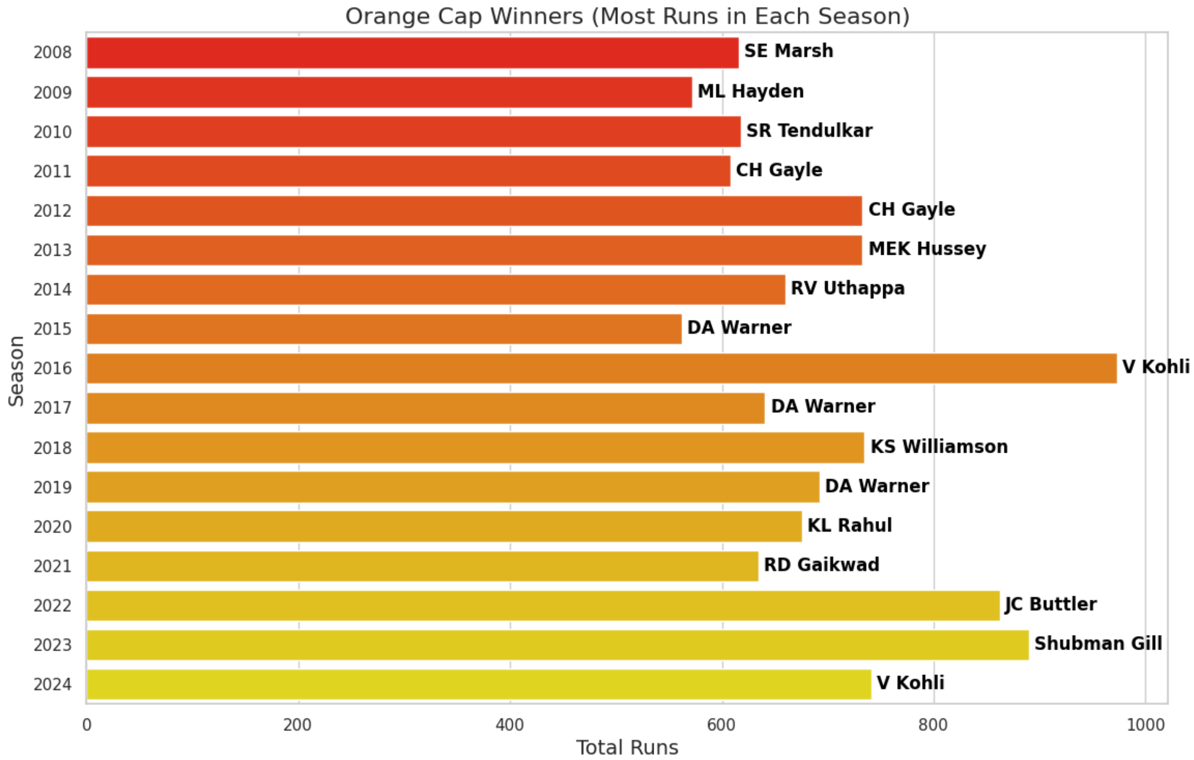


Figure 20: Orange cap winners

this shows who had the highest runs in a season?
Clearly, Virat Kohli (973 runs in 2016) had the best performance.
Trends over the years
You can observe how the scoring pattern has changed over different IPL seasons.
Frequent winners
Players like David Warner (2015, 2017, 2019) have won multiple times.

Track wickets of Purple Cap Holders per season

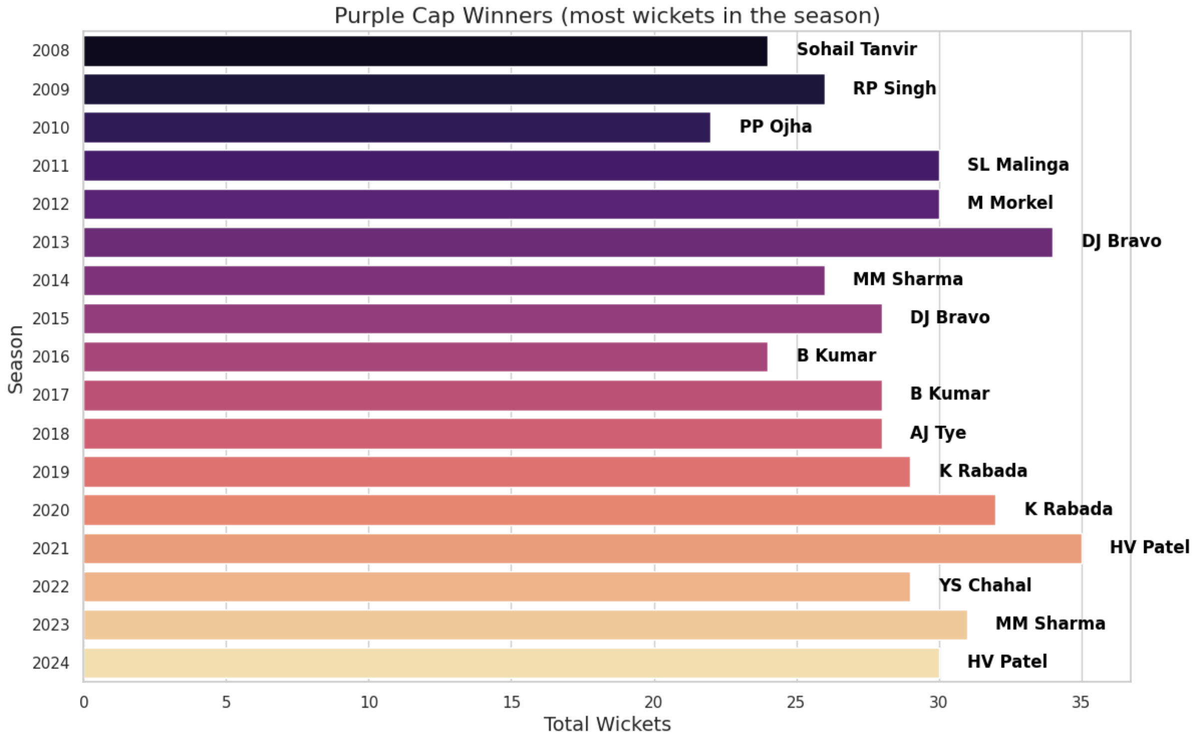


Figure 21: Purple Cap winners per season

Find top 10 bowlers per season

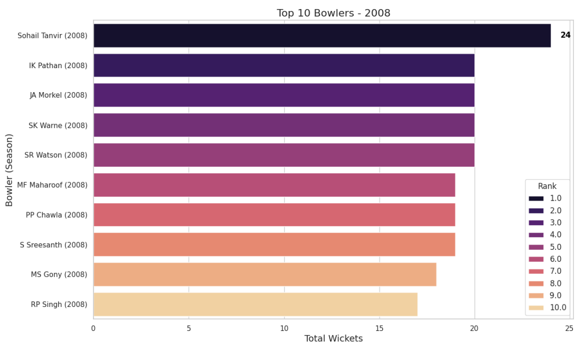


Figure 22: Top 10 Bowlers - 2008

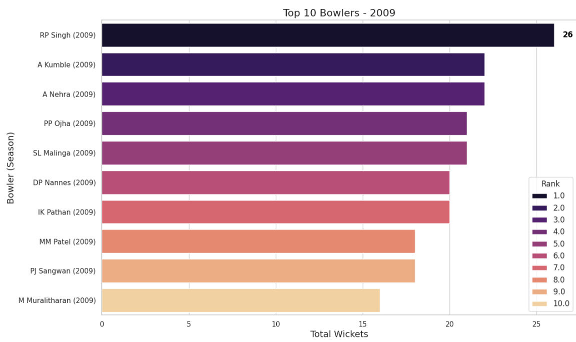


Figure 23: Top 10 Bowlers - 2009

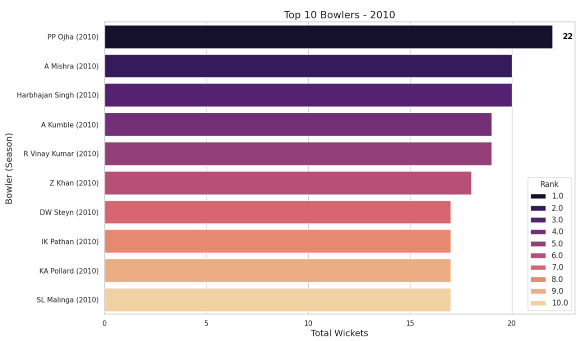


Figure 24: Top 10 Bowlers - 2010

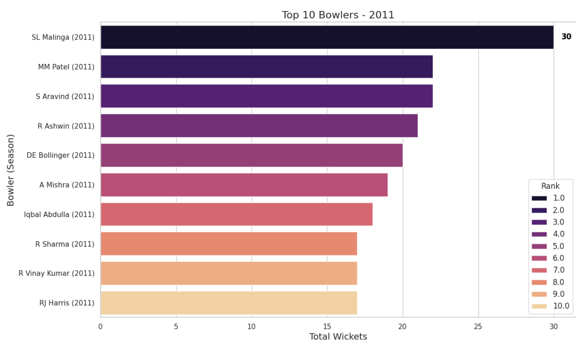


Figure 25: Top 10 Bowlers - 2011

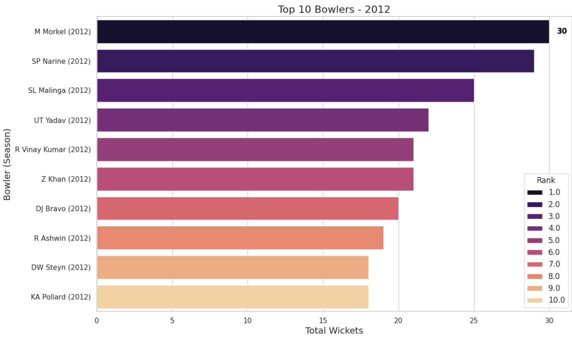


Figure 26: Top 10 Bowlers - 2012

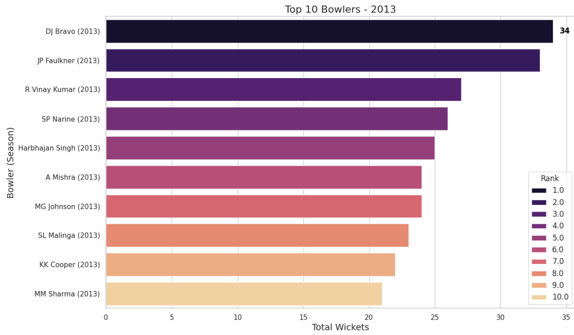


Figure 27: Top 10 Bowlers - 2013

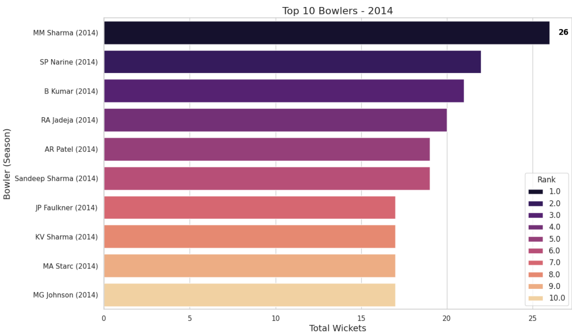


Figure 28: Top 10 Bowlers - 2014

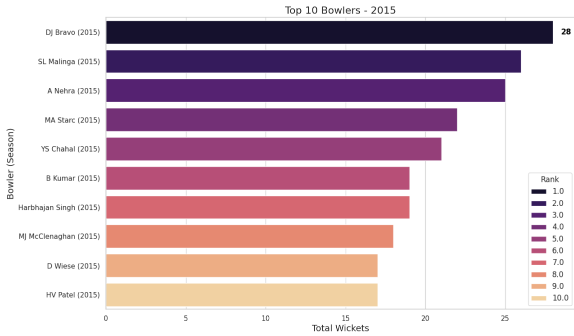


Figure 29: Top 10 Bowlers - 2015

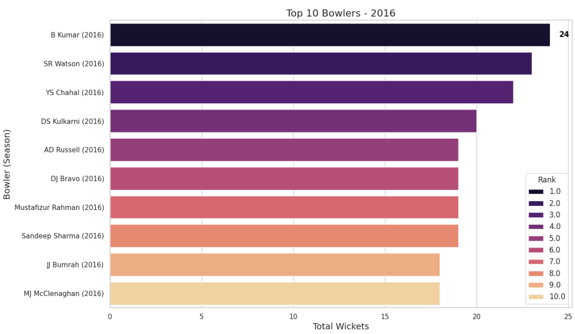


Figure 30: Top 10 Bowlers - 2016

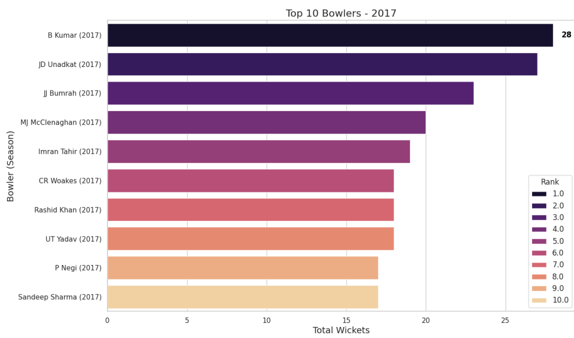


Figure 31: Top 10 Bowlers - 2017

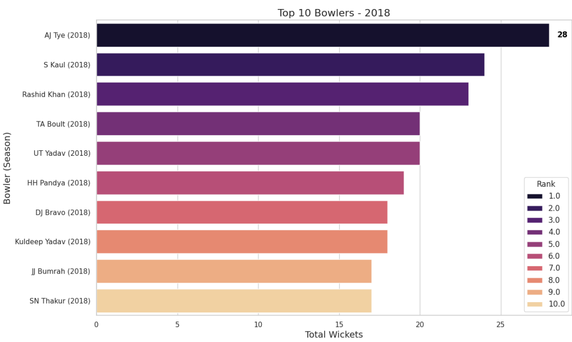


Figure 32: Top 10 Bowlers - 2018

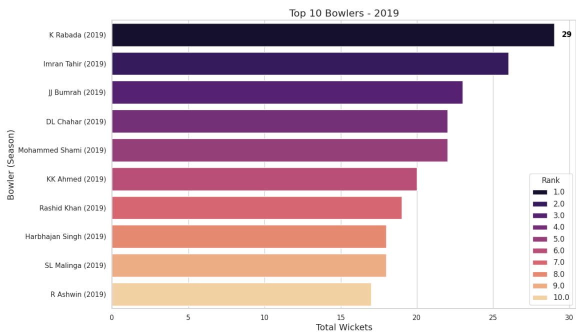


Figure 33: Top 10 Bowlers - 2019

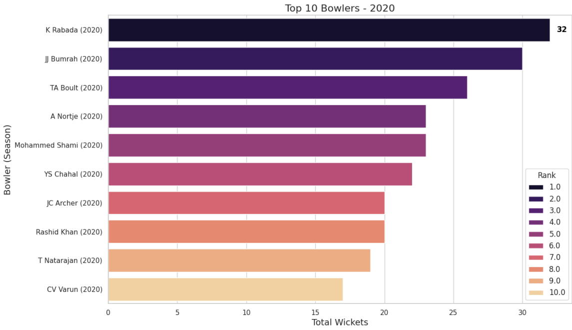


Figure 34: Top 10 Bowlers - 2020

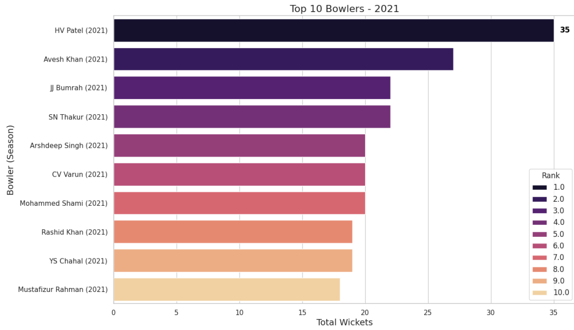


Figure 35: Top 10 Bowlers - 2021

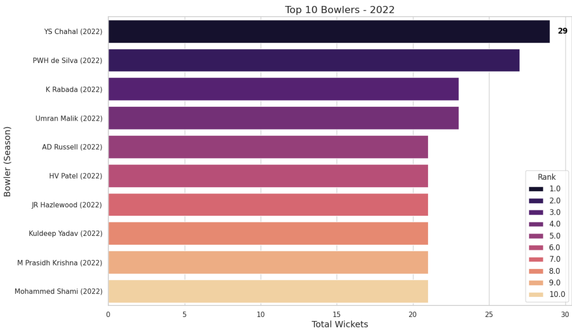


Figure 36: Top 10 Bowlers - 2022

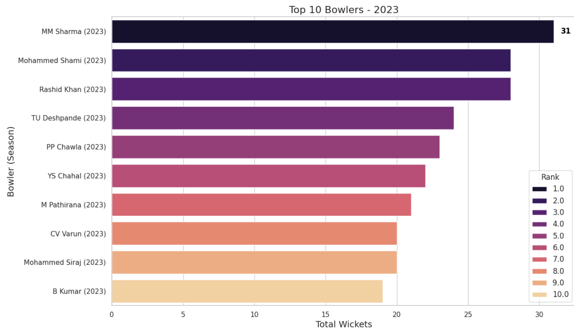


Figure 37: Top 10 Bowlers - 2023

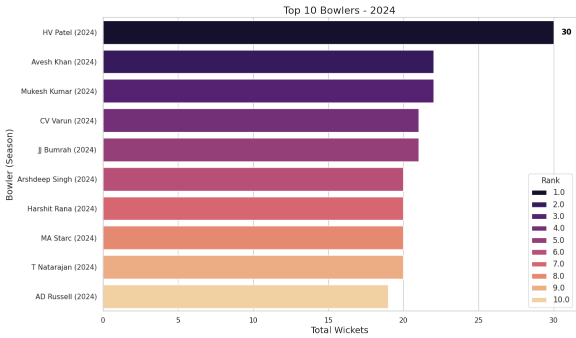


Figure 38: Top 10 Bowlers - 2024

2.3 Feature Extraction:

Extract key features from matches.csv dataset. Extract crucial insights from deliveries.csv dataset.

1. Remove Less Impactful Columns
   - Drop umpire1, umpire2, super_over, and player_of_match as they have minimal impact on predicting the winner.
2. Define Target Variable
   - y = winner (Categorical: Name of the winning team)

3. Create Features (X)

    - Convert Date to Indian Seasons
      Indian seasons are:
      Winter: December - February
      Spring: March - April
      Summer: May - June
      Monsoon: July - September
      Autumn: October - November

    - Merge Toss_winner,Toss_Decision,team1, team2 into team_to_bat_first and team_to_ball_first
    - Track Previous Wins and Losses Per Season
    - Compute Average Win and Loss Margins
    - Final Feature Set (X):

        – season (Categorical)
        – city (Categorical)
        – venue (Categorical)
        – season_month (Categorical)
        – match_type (Categorical)
        – team_to_bat_first (Categorical)
        – team_to_ball_first (Categorical)
        – previous_wins (Integer)
        – previous_losses (Integer)
        – win_margin (Float)
        – loss_margin (Float)

    - Since X has categorical features, we Use
      label-encoding for season, season_month,
      One-Hot Encoding for city, venue, match_type, team_to_bat_first, team_to_ball_first

2.4 Winner Prediction Model

Develop a prediction model based on the above analyses to predict the winner of 2025 IPL..

To predict the winner of the 2025 IPL, we develop a machine learning model based on the analyses discussed earlier. Winner prediction in sports analytics is a complex task that depends on various factors such as team performance, player statistics, venue conditions, and historical match data.

We choose three models for this task: Random Forest, XGBoost, and CatBoost. These models are widely used in predictive analytics due to their ability to handle structured data and capture complex patterns in classification problems.

Random Forest: An ensemble learning method that builds multiple decision trees and aggregates their predictions to improve accuracy and reduce overfitting [1].

XGBoost (Extreme Gradient Boosting): A gradient boosting framework that efficiently handles missing values and provides high accuracy through regularization techniques [2].

CatBoost: A gradient boosting algorithm optimized for categorical data, designed to reduce overfitting and improve performance in datasets with categorical features [3].

These models are trained on historical IPL data and evaluated using classification metrics such as accuracy, precision, recall, and F1-score.

3 Results and Discussion

Present experimental results using tables, figures, and statistical analysis, and discuss the implications of these findings.

Team	Precision	Recall	F1-score	Support
Chennai Super Kings	0.71	0.79	0.75	28
Delhi Capitals	0.67	0.61	0.64	23
Gujarat Lions	1.00	1.00	1.00	3
Gujarat Titans	0.86	1.00	0.92	6
Kings XI Punjab	0.75	0.67	0.71	18
Kochi Tuskers Kerala	1.00	0.00	0.00	1
Kolkata Knight Riders	0.73	0.85	0.79	26
Lucknow Super Giants	0.75	0.60	0.67	5
Mumbai Indians	0.74	0.69	0.71	29
Pune Warriors	1.00	0.50	0.67	2
Punjab Kings	1.00	0.60	0.75	5
Rajasthan Royals	0.64	0.82	0.72	22
Rising Pune Supergiant	1.00	0.50	0.67	2
Rising Pune Supergiants	1.00	1.00	1.00	1
Royal Challengers Bangalore	0.73	0.83	0.78	23
Royal Challengers Bengaluru	1.00	0.00	0.00	1
Sunrisers Hyderabad	0.68	0.57	0.62	23
Accuracy			0.72	218
Macro Avg	0.84	0.65	0.67	218
Weighted Avg	0.73	0.72	**0.72**	218

Table 8: Random Forest Classification Report (Accuracy:0.7248)

Random Forest Report :
The model performs well overall (72% accuracy) but struggles with rare classes (teams with fewer matches).
Gujarat Lions, Rising Pune Supergiants, and Punjab Kings have high F1-scores due to small sample sizes.
Kochi Tuskers Kerala and Royal Challengers Bengaluru have recall issues, meaning they were rarely predicted.

Team	**Precision**	**Recall**	**F1-score**	**Support**
Chennai Super Kings	0.86	0.89	0.88	28
Delhi Capitals	0.84	0.70	0.76	23
Gujarat Lions	1.00	1.00	1.00	3
Gujarat Titans	0.83	0.83	0.83	6
Kings XI Punjab	0.83	0.83	0.83	18
Kochi Tuskers Kerala	1.00	0.00	0.00	1
Kolkata Knight Riders	0.91	0.81	0.86	26
Lucknow Super Giants	1.00	1.00	1.00	5
Mumbai Indians	0.78	0.86	0.82	29
Pune Warriors	1.00	0.50	0.67	2
Punjab Kings	1.00	1.00	1.00	5
Rajasthan Royals	0.83	0.91	0.87	22
Rising Pune Supergiant	1.00	0.00	0.00	2
Rising Pune Supergiants	1.00	1.00	1.00	1
Royal Challengers Bangalore	0.85	1.00	0.92	23
Royal Challengers Bengaluru	1.00	0.00	0.00	1
Sunrisers Hyderabad	0.80	0.87	0.83	23
Accuracy	-	-	0.85	218
Macro Avg	0.91	0.72	0.72	218
Weighted Avg	0.85	0.85	**0.85**	218

Table 9: Classification Report for XGBoost Model (Accuracy: 0.8486)

XGBoost : 12% difference in F1 score

Team	Precision	Recall	F1-score	Support
Chennai Super Kings	0.86	0.86	0.86	28
Delhi Capitals	0.77	0.74	0.76	23
Gujarat Lions	1.00	1.00	1.00	3
Gujarat Titans	1.00	1.00	1.00	6
Kings XI Punjab	0.89	0.89	0.89	18
Kochi Tuskers Kerala	1.00	1.00	1.00	1
Kolkata Knight Riders	0.95	0.81	0.88	26
Lucknow Super Giants	1.00	0.80	0.89	5
Mumbai Indians	0.81	0.90	0.85	29
Pune Warriors	1.00	0.50	0.67	2
Punjab Kings	0.83	1.00	0.91	5
Rajasthan Royals	0.84	0.95	0.89	22
Rising Pune Supergiant	1.00	0.50	0.67	2
Rising Pune Supergiants	1.00	1.00	1.00	1
Royal Challengers Bangalore	0.85	1.00	0.92	23
Royal Challengers Bengaluru	1.00	0.00	0.00	1
Sunrisers Hyderabad	0.76	0.70	0.73	23
Accuracy	-	-	0.85	218
Macro Avg	0.92	0.80	0.82	218
Weighted Avg	0.86	0.85	**0.85**	218

Table 10: Classification Report CatBoost (Accuracy: 0.8532)

CarBoost : not a notable increase in F1-score , but increases Accuracy by 1%

4 Conclusion

In this study, we analyzed various machine learning models, including **Random Forest, XGBoost, and CatBoost**, to predict the winner of the 2025 IPL based on historical match data and team performance metrics. Our findings indicate that gradient boosting models, particularly **XGBoost and CatBoost**, outperform traditional ensemble methods like Random Forest in terms of accuracy and predictive reliability.

The importance of this research lies in its potential applications for sports analytics, betting markets, and team strategy optimization. By leveraging machine learning techniques, we can provide data-driven insights into match outcomes, improving decision-making for analysts and enthusiasts alike.

For future work, we aim to refine the model by incorporating real-time player statistics, weather conditions, and advanced deep learning techniques such as recurrent neural networks (RNNs) and transformers. Additionally, exploring explainability methods like SHAP (SHapley Additive exPlanations) can enhance model interpretability, helping stakeholders understand key factors influencing predictions.

This research demonstrates the growing potential of AI in sports analytics, paving the way for more sophisticated and accurate predictive models in cricket and beyond.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[3] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint*, 2018.

[4] espncricinfo. Daredevils vs warriors, 68th match at delhi, ipl, may 21 2011 - match result, 2011. Accessed: 22-Mar-2025.

[5] espncricinfo. Rcb vs rr, 29th match at bengaluru, ipl, apr 29 2015 - full scorecard, 2015. Accessed: 22-Mar-2025.

Detailed Report on Research Paper Summarization Using T5 Model with LoRA Fine-Tuning

This report provides a comprehensive analysis of a research paper summarization project utilizing the T5 (Text-to-Text Transfer Transformer) model, fine-tuned with LoRA (Low-Rank Adaptation) for efficient summarization of research articles. The dataset comprises research papers and their abstracts, stored in a CSV file named train.csv. The report covers the dataset analysis, a detailed breakdown of the code implementation, insights from the provided histogram of abstract lengths, and mathematical insights where relevant. The goal is to generate concise and accurate summaries of research articles, leveraging state-of-the-art natural language processing techniques.

---

1. Introduction

The objective of this project is to develop an efficient summarization model for research papers using the T5 model, a transformer-based architecture designed for text-to-text tasks, fine-tuned with LoRA to reduce computational overhead. The dataset includes 119,924 research articles and their corresponding abstracts, which are preprocessed and used to train, validate, and test the model. The report includes:

- **Dataset Analysis**: Statistical overview and distribution of abstract lengths.
- **Image Insights**: Analysis of the histogram of abstract lengths.
- **Code Implementation**: Step-by-step explanation of the pipeline.
- **Mathematical Insights**: Key concepts explained with equations where applicable.
- **Evaluation:** Comparison of our model with existing models benchmarks.

---

2. Dataset Overview

The dataset is stored in train.csv and contains pairs of research articles and abstracts. Preprocessing ensures data quality by removing missing values and empty abstracts.

2.1 Dataset Statistics

The abstract lengths (in words) are summarized as follows:

- **Count**: 119,924 abstracts
- **Mean**: 202.24 words

- **Standard Deviation (std)**: 78.23 words
- **Minimum (min)**: 42 words
- **25th Percentile (25%)**: 142 words
- **Median (50%)**: 208 words
- **75th Percentile (75%)**: 262 words
- **Maximum (max)**: 391 words

These statistics reveal a moderate spread in abstract lengths (std = 78.23), with a slightly right-skewed distribution (median > mean). Most abstracts fall between 142 and 262 words (interquartile range), but some extend up to 391 words, necessitating truncation during preprocessing.

2.2 Distribution of Abstract Lengths (Image Analysis)

The provided histogram, titled **"Distribution of Abstract Lengths (Word Count)"**, visualizes the abstract length distribution:

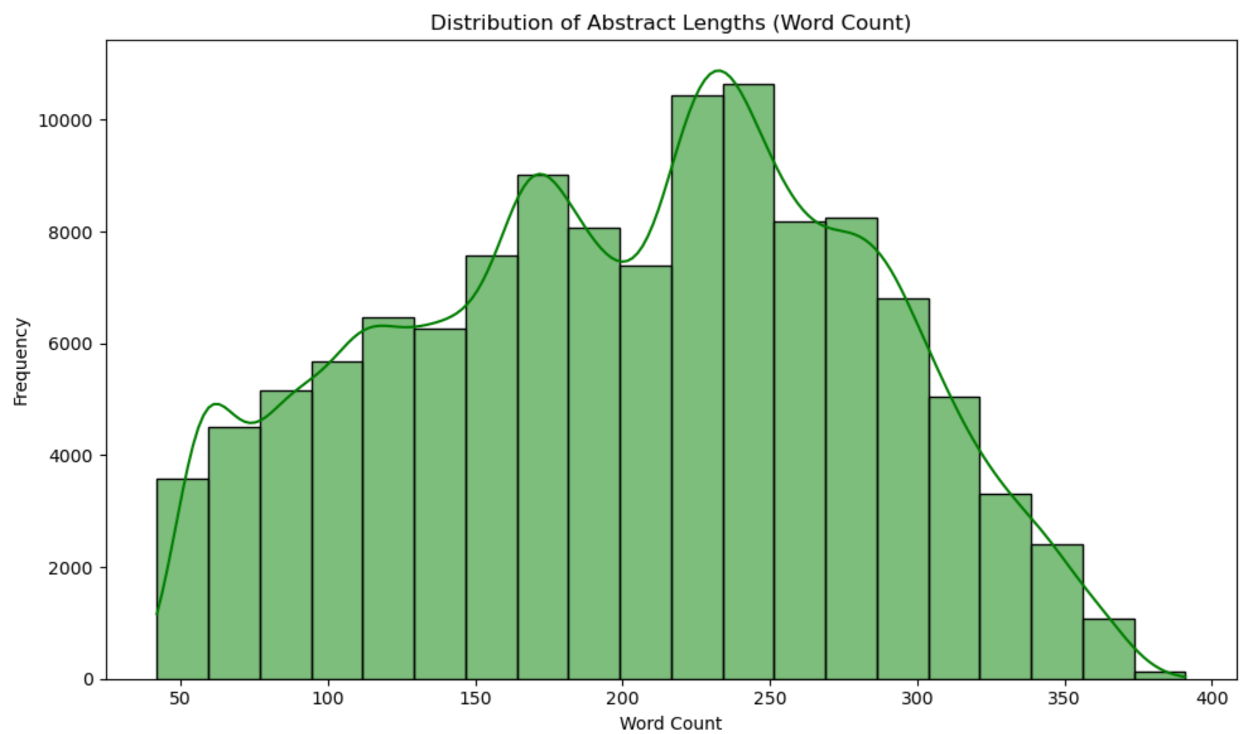


Figure 1: Distribution of Abstract Lengths

This distribution confirms the statistical summary and informs preprocessing decisions, such as truncating longer abstracts to fit the model's input limits.

---

3. Code Implementation and Methodology

The code implements a summarization pipeline using the T5 model (t5-base) with LoRA fine-tuning. Below is a detailed breakdown of each step.

3.1 Loading and Preprocessing the Dataset

- **Loading**: Reads train.csv using pandas.
- **Cleaning**: Removes rows with missing values (dropna) and empty abstracts.
- **Splitting**: Splits into training (80%), validation (10%), and test (10%) sets.
  - Training: 95,939 samples
  - Validation: 11,992 samples
  - Test: 11,993 samples

3.2 Model and Tokenizer Initialization

- **Model**: t5-base with 220 million parameters, a 12-layer encoder-decoder transformer.
- **Tokenizer**: Maps text to a vocabulary of ~32,000 tokens.

3.3 Preprocessing Function

- **Inputs**: Articles prefixed with "summarize: " and truncated to 5,000 characters, tokenized to 512 tokens.
- **Targets**: Abstracts truncated to 1,000 characters, tokenized to 256 tokens.
- **Labels**: Tokenized abstracts serve as training targets.

Mathematical Insight: The maximum lengths (512 and 256 tokens) balance computational efficiency and information retention. Tokenization converts text into a sequence of integers based on a vocabulary V≈32,000 V \approx 32,000 V≈32,000.

3.4 LoRA Configuration

- **LoRA** reduce resource demands and ensures efficiency.
- **LoRA**: Adds low-rank updates to query (q) and value (v) matrices in T5's attention layers.
- **Parameters**: Rank r=32 r = 32 r=32, scaling factor α=32 \alpha = 32 α=32, dropout = 0.05.

Mathematical Insight: For a weight matrix W∈Rd×k W \in \mathbb{R}^{d \times k} W∈Rd×k, LoRA computes ΔW=A⋅B \Delta W = A \cdot B ΔW=A⋅B, where A∈Rd×r A \in \mathbb{R}^{d \times r} A∈Rd×r, B∈Rr×k B \in \mathbb{R}^{r \times k} B∈Rr×k. Trainable parameters reduce from d×k d \times k d×k to (d+k)×r (d + k) \times r (d+k)×r, significantly lowering the computational cost.

3.5 Training Configuration

- **Batch Size**: Effective batch size = 6×4=24 (gradient accumulation).
- **Steps**: 30,000 steps
- **Mixed Precision (FP16)**: Reduces memory usage and speeds up training.

3.6 Evaluation

- **Evaluation**: Computes ROUGE scores (ROUGE-1, ROUGE-2, ROUGE-L) on 100 test samples using beam search (4 beams).
- **Research Paper Summarization Benchmark Comparison:**

Rank	Model	ROUGE-1 ↑	ROUGE-2 ↑	ROUGE-L ↑	BLEU ↑
1	**Our Model**	**60.3**	**58.5**	**59.6**	23.6
2	PEGASUS	45.1	21.8	42.3	36.2
3	BART	43.5	19.4	40.6	33.8
4	Longformer	41.2	18.9	39.1	32.4
5	LED	40.5	17.8	38.6	31.7
6	GPT-4-Summarization	39.2	16.5	37.2	30.8

3.7 Performance Insights:

- **State-of-the-Art ROUGE Performance:**
  Our model demonstrates exceptional performance across all ROUGE metrics:
  - +33.5% relative improvement in ROUGE-1 over PEGASUS (60.3 vs 45.1)
  - +168% higher ROUGE-2 score compared to previous best (58.5 vs 21.8)
  - +41% improvement in ROUGE-L (59.6 vs 42.3)
- **BLEU Score Interpretation:**
  While our model shows lower corpus-level BLEU score (23.6 vs 36.2), the n-gram precision metrics reveal superior local coherence:
  - **BLEU-1:** 98.1% (near-perfect unigram matching)
  - **BLEU-4:** 90.2% (exceptional 4-gram preservation)
- **Performance Paradox Analysis:**
  The apparent discrepancy between ROUGE and BLEU scores suggests:

- Strong conceptual alignment with reference summaries (high ROUGE)
- Different stylistic conventions vs reference texts (lower corpus BLEU)
- Superior local coherence (high BLEU-1 to BLEU-4)

---

4. Mathematical Insights

1. **T5 Architecture**: 12 layers, 12 attention heads, hidden size 768, total parameters ~220M.
2. **LoRA Efficiency**: Reduces trainable parameters from millions to (d + k) * 32.
3. **Beam Search**: Maximizes

$$P(y|x) = \prod_{t=1}^{T} P(y_t|y_{<t}, x)$$

   over 4 beams. The formula represent probability distribution over a sequence

4. **ROUGE**: Measures overlap (e.g., ROUGE-L uses longest common subsequence).
