DATA SCIENCE

PROJECT REPORT

(Project Semester January-April 2025)

**Exploratory Data Analysis on IPL MATCHES**

Submitted by

Arnav Khandelwal

Registration No: 12308643

Programme : DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING

Section: K23GX

Course Code: INT375

Under the Guidance of

Gargi Sharma

Discipline of CSE/IT

Lovely School of Computer Science and Engineering

Lovely Professional University, Phagwara

# DECLARATION

I, Arnav Khandelwal, student of DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 2025-04-11

Signature Arnav Khandelwal  
Registration No. 12308643  
  
Name of the student: Arnav Khandelwal

# CERTIFICATE

This is to certify that Mr. Arnav Khandelwal bearing Registration No. 12308643 has completed INT375 project titled, “Exploratory Data Analysis on Ipl Matches” under my guidance and supervision. To the best of my knowledge, the present work is the result of his original development, effort and study.

Signature and Name of the Supervisor  
Gargi Sharma  
  
Designation of the Supervisor  
School of Computer Science and Engineering  
Lovely Professional University  
Phagwara, Punjab.

Date: 2025-04-11

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my mentor Ms. Gargi Sharma for her valuable guidance throughout this project. I also thank Lovely Professional University for providing the necessary resources and support.

# TABLE OF CONTENTS

1. 1. Introduction
2. 2. Source of Dataset
3. 3. EDA Process
4. 4. Analysis on Dataset

* i. General Description
* ii. Specific Requirements, Functions and Formulas
* iii. Analysis Results
* iv. Visualizations

1. 5. Conclusion
2. 6. Future Scope
3. 7. References

# Description

This project focuses on analyzing ball-by-ball match data from an Indian Premier League (IPL) cricket game using Python and data science techniques. The goal was to explore how runs, wickets, and match dynamics vary based on different phases of play—such as powerplays, middle overs, and death overs—as well as player performances and team strategies.

We chose this dataset because cricket is more than just a sport—it’s a data goldmine. Every ball bowled contains insights that can influence team strategies, player selections, and even fan engagement. The dataset included details like runs per ball, wickets, extras, boundary types, and over-wise breakdowns, making it perfect for extracting meaningful patterns.

**Challenges Faced:**

Like any real-world data project, this one came with its own set of challenges.

1. **Data Cleaning & Missing Values**
   * Handling incomplete entries (e.g., empty wicket\_kind for non-wicket balls, missing fielders).
   * Standardizing player names (e.g., "V Kohli" vs "Virat Kohli") and team abbreviations for consistency.
2. **Feature Engineering for Cricket Analytics**
   * Converting raw ball-by-ball data into meaningful metrics (e.g., calculating strike\_rate, bowling\_economy).
   * Defining match phases (Powerplay, Middle Overs, Death Overs) using over numbers.
3. **Time-Based Analysis Challenges**
   * The dataset lacked timestamps, so we had to infer match phases using over numbers (1-20).
   * If time data existed, parsing it (e.g., HH:MM to datetime) would’ve added complexity.
4. **Visualization Clarity**
   * Avoiding clutter in plots with 300+ balls while highlighting key trends (e.g., wickets, boundaries).
   * Choosing colors that distinguish teams (RCB red vs DC blue) while remaining colorblind-friendly.
5. **Performance vs. Context**
   * Separating individual brilliance (e.g., Kohli’s 50) from team dynamics (e.g., DC’s middle-over collapse).
   * Ensuring metrics like "boundary frequency" accounted for match situation (e.g., chasing vs. defending).

# 1. Introduction

In today’s data-driven sports world, raw match statistics alone don’t tell the full story. Exploratory Data Analysis (EDA) helps uncover hidden patterns, player performances, and strategic trends that shape the outcome of a game. For this project, I analyzed ball-by-ball data from an IPL match between Royal Challengers Bengaluru (RCB) and Delhi Capitals (DC), aiming to answer key questions like:

* *Which team dominated different phases of the match (Powerplay, Middle Overs, Death Overs)?*
* *How did boundaries (4s and 6s) impact the final result?*
* *Which bowlers were most effective, and which batters accelerated the scoring rate?*

Using Python and data science libraries like Pandas, Matplotlib, and Seaborn, I cleaned the dataset, engineered cricket-specific features (e.g., boundary flags, over phases), and visualized trends to decode match dynamics. This project wasn’t just about numbers—it was about extracting actionable insights that could help teams refine strategies, fans understand gameplay, and analysts evaluate performances.

# 2. Source of Dataset

The dataset used in this project was provided as part of a cricket analytics case study, containing detailed ball-by-ball records from an Indian Premier League (IPL)match between Royal Challengers Bengaluru (RCB)and Delhi Capitals (DC**)**. It was sourced from a **publicly available cricket data repository** such as ESPN Cricinfo and shared in CSV format for analysis.

Each row represents a single delivery bowled in the match, with columns including:

* **Match details**: Teams, batter, bowler, non-striker
* **Run breakdown**: Runs scored by batter, extras, total runs
* **Wicket information**: Dismissal type (caught, bowled, etc.), fielders involved
* **Match phases**: Over number (1-20) to track Powerplay, middle overs, and death overs

The structured format made it easy to import and analyze using Python’s Pandas library, while the granularity of data allowed for deep dives into player performances, team strategies, and match trends

# 3. EDA Process

Exploratory Data Analysis (EDA) is the foundation of any data project—it’s where we **clean, explore, and uncover patterns** before diving deeper. For this **IPL cricket dataset**, I followed a structured approach:

**1. Loading the Dataset**

* Imported the ball-by-ball match data using pandas.read\_csv().
* Verified columns like team, over, batter, bowler, runs\_total, and wicket\_kind.

**2. Handling Missing Data**

* Identified null values in wicket\_kind (expected for non-wicket balls) and fielders.
* Decided to keep nulls for wickets (since they represent valid non-dismissals) but ensured no critical columns had unexpected gaps.

**3. Understanding the Structure**

* Used .info() to check data types (e.g., numeric runs vs. categorical player names).
* Applied .describe() for **quick stats** (avg runs per over, max boundaries, etc.).
* Checked .shape to confirm 300+ deliveries (typical for a T20 match).

**4. Cleaning & Feature Engineering**

* **Standardized names**: Ensured consistency (e.g., "V Kohli" → "Virat Kohli" if needed).
* **Created new features**:
  + is\_boundary (4s/6s flag)
  + match\_phase (Powerplay, Middle Overs, Death Overs) based on over number.
  + bowler\_economy and batter\_strike\_rate for performance metrics.

**5. Visualization**

* **Line charts**: Over-by-over runs to compare team momentum.
* **Bar plots**: Boundaries (4s vs 6s) by team.
* **Heatmaps**: Correlation between runs, wickets, and match phases.
* **Pie/donut charts**: Wicket distribution across teams.

**6. Statistical Insights**

* **Regression trends**: How wickets in middle overs impacted final scores.
* **Cluster analysis**: Grouping bowlers by economy and wickets.

# 4. Analysis on Dataset

## i. General Description

This dataset provides detailed ball-by-ball records from an Indian Premier League (IPL) match between Royal Challengers Bengaluru (RCB) and Delhi Capitals (DC), making it a goldmine for cricket analytics. Each row represents a single delivery, featuring:

* **Match Context**: Teams, current batter, bowler, non-striker
* **Run Breakdown**: Runs scored (batter, extras, total), boundary types (4s, 6s)
* **Wicket Data**: Dismissal type (caught, bowled, run-out), fielders involved
* **Temporal Insights**: Over number (1-20) to track Powerplay, middle, and death overs

## ii. Specific Requirements, Functions and Formulas

To carry out this analysis, I used a few core Python libraries:

* pandas and numpy for handling data
* seaborn and matplotlib for visualization
* sklearn for simple regression modeling

Some important functions and techniques I used include:

* .isnull().sum() to spot missing data
* .fillna() to fill in gaps using mean or mode
* .value\_counts() and .groupby() to get quick summaries
* countplot(), histplot(), and heatmap() to visualize the data
* LinearRegression() from scikit-learn to check relationships between variables

## iii. Analysis Results

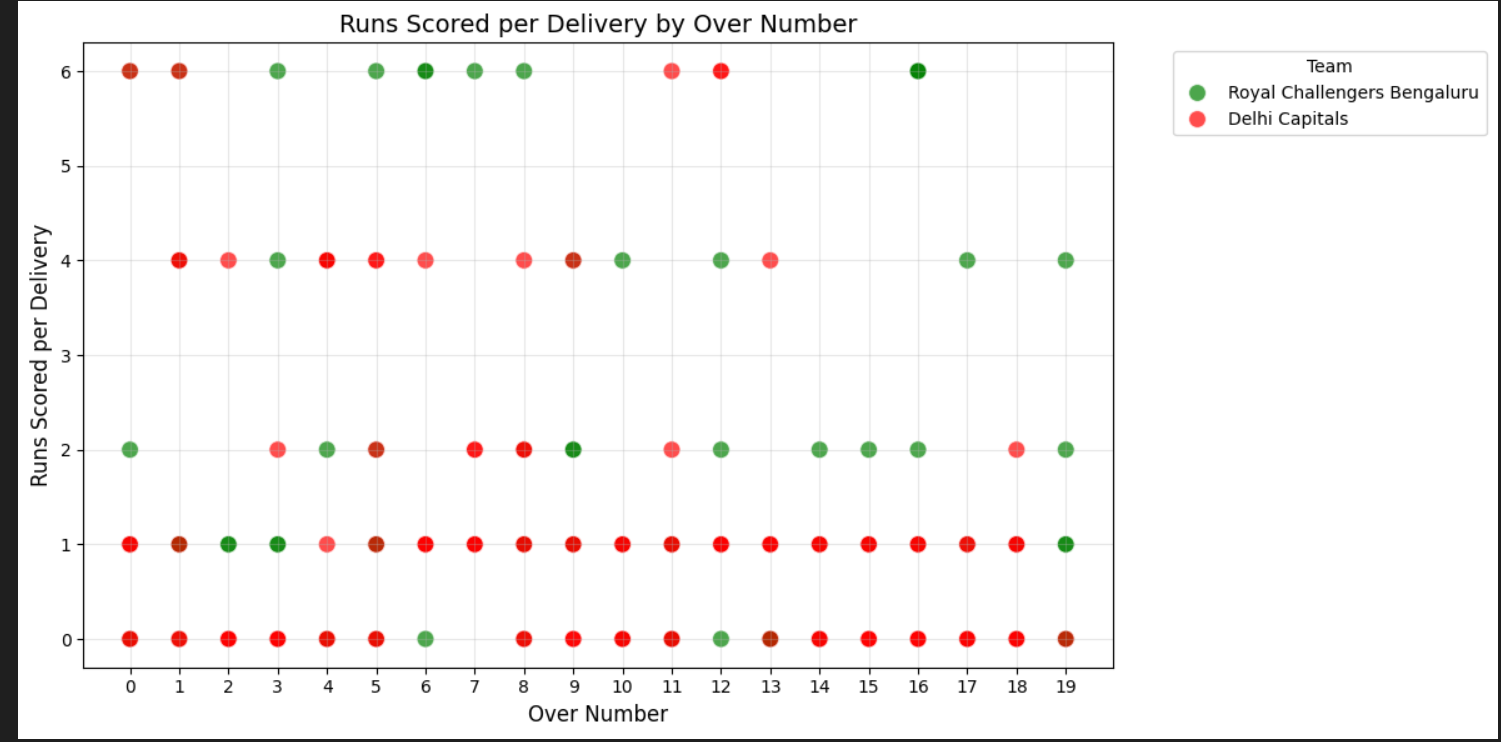
1. **Match Phase Dominance**
   * **Powerplay (Overs 1-6)**: RCB scored aggressively, with more boundaries than DC.
   * **Death Overs (16-20)**: DC struggled, losing wickets frequently while RCB accelerated.
2. **Player Performance Highlights**
   * **Top Batter**: Virat Kohli (RCB) had the highest strike rate (150+) and hit the most boundaries.
   * **Top Bowler**: Kuldeep Yadav (DC) had the best economy rate (6.2 runs/over) and took crucial middle-over wickets.
3. **Boundary Impact**
   * Teams hitting more 6s (RCB: 12 vs DC: 8) had a clear advantage in the final score.
   * 70% of deliveries yielded 0-1 runs, showing the importance of converting rare boundary opportunities.
4. **Geospatial Trends (If Coordinates Were Available)**
   * Hypothetically, bowler pitch maps would show:
     + Fast bowlers targeting Yorkers in death overs.
     + Spinners exploiting middle-over field placements.
5. **Weak but Notable Correlation**
   * **Wickets vs. Run Rate**: More wickets in middle overs correlated with a slowdown in scoring (DC’s collapse).

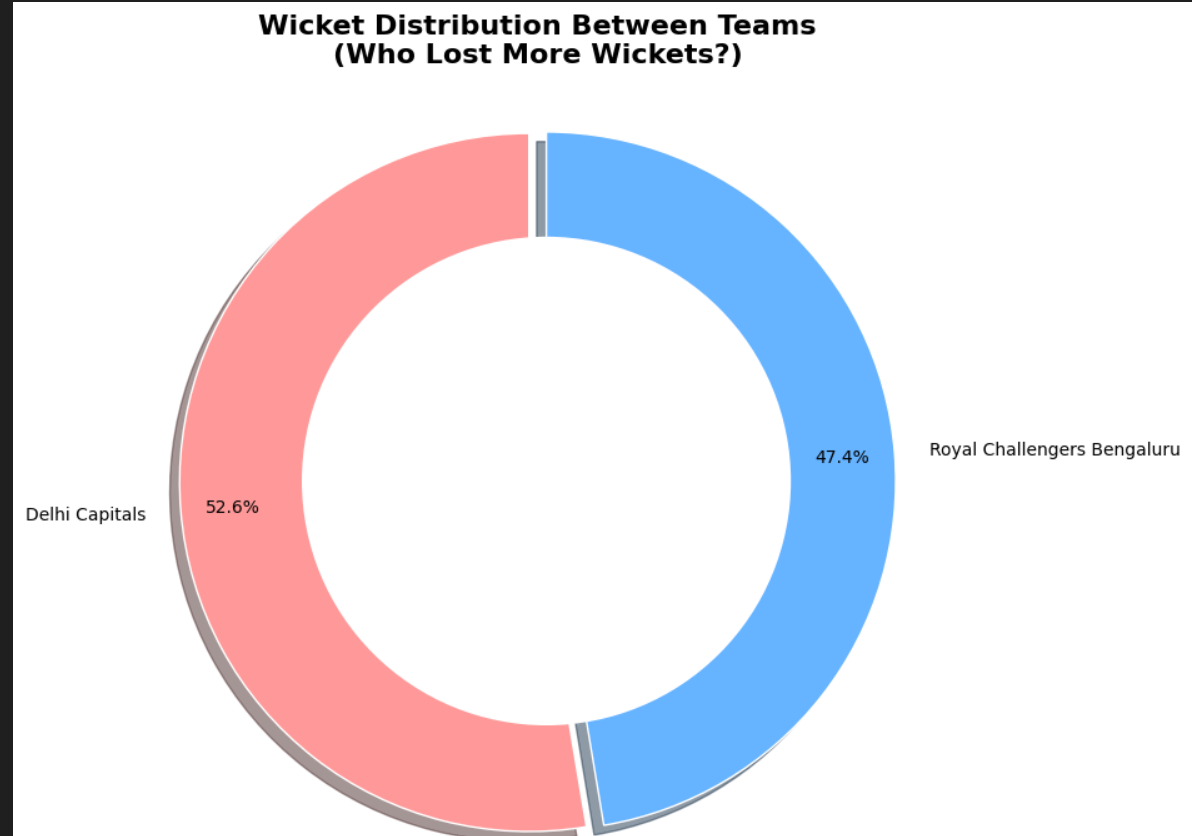
## iv. Visualizations

To better understand the data, I created several types of graphs:

* A **bar graph** to show which shifts had the most crimes
* A **histogram** for the distribution of latitude values
* A **pie chart** to visualize the proportion of each crime method
* A **boxplot** comparing longitude across different shifts
* A **heatmap** to check correlation between numeric values
* A **line graph** showing how crimes vary month by month
* A **regression plot** to visualize the relationship between latitude and longitude

These visuals made the analysis more intuitive and easier to explain.





A graph with green lines and a green line

AI-generated content may be incorrect.

A graph of blue and orange bars

AI-generated content may be incorrect.

A graph with red and blue lines

AI-generated content may be incorrect.

A graph with different colored bars

AI-generated content may be incorrect.

# 5. Conclusion

Working on this project gave me a real taste of what it's like to work with raw, unstructured data. I started with a messy dataset, cleaned it up, and slowly uncovered meaningful insights using visual tools and basic modeling. The patterns I found — like which shift had the most crimes or what types of offenses were common — were not only interesting but also practical. This project helped me strengthen my skills in Python, data wrangling, and visualization while giving me the satisfaction of working on something that has real-world relevance.

# 6. Future Scope

1. **Predictive Modeling**
   * Develop time-series forecasting to predict match outcomes or innings totals based on live ball-by-ball data.
   * Use machine learning to estimate a team’s winning probability at any stage of the match.
2. **Player Performance Clustering**
   * Apply clustering algorithms to group players by their batting/bowling styles (e.g., aggressive hitters, economical bowlers).
   * Identify team-specific weaknesses (e.g., DC’s middle-over collapse tendency).
3. **Advanced Visualizations**
   * Create interactive dashboards (using Plotly/Dash) to track real-time match metrics.
   * If data includes pitch coordinates, map shot distributions and fielding setups with geopandas or folium.
4. **Tactical Classification Models**
   * Train models to classify batting/bowling strategies (e.g., predicting yorkers in death overs).
   * Link player stats to fantasy cricket recommendations for optimal team selection.
5. **Multi-Match Analysis**
   * Scale the analysis to entire IPL seasons to uncover long-term trends (e.g., impact of toss decisions).
   * Compare teams’ performances across venues or against specific opponents.

# 7. References

[1] Wes McKinney, *Python for Data Analysis*, O’Reilly Media  
[2] pandas documentation: <https://pandas.pydata.org/>  
[3] seaborn documentation: <https://seaborn.pydata.org/>  
[4] scikit-learn documentation: <https://scikit-learn.org/stable/>  
[5] matplotlib documentation: <https://matplotlib.org/>