## Artificial

## Intelligence and Machine Learning

Project Report

Semester-IV (Batch-2022)

**IPL Win Probability Predictor**

A red and white sign

Description automatically generated with low confidence

**Submitted To: Submitted By:**

Dr. Kiran Deep Singh Parul Arora,2210990645(G-8A)

Piyush Kumar,2210990652(G-8A)

Pranya,2210990670(G-8A)

Prerna Dua,2210990680(G-8B)

**Department of Computer Science and Engineering**

## Chitkara University Institute of Engineering & Technology,

## Chitkara University, Punjab

**Abstract**

This report outlines the methodology employed, which involves data collection, pre-processing, feature engineering, and model development. Various machine learning algorithms such as logistic regression, decision trees, random forests, and gradient boosting are explored to ascertain the most effective approach for win probability prediction.

Key features including team performance metrics, player statistics, match venue, weather conditions, and historical data are integrated into the model to capture the multifaceted dynamics of IPL matches. Furthermore, ensemble techniques and cross-validation strategies are employed to enhance model robustness and generalization capabilities.

The performance of the developed model is evaluated using appropriate metrics such as accuracy, precision, recall, and F1-score. Additionally, the model's calibration and reliability are assessed through calibration plots and calibration error metrics.

Results indicate promising predictive performance, with the model demonstrating the ability to accurately forecast match outcomes and provide valuable insights for stakeholders including team management, broadcasters, and fans. Future enhancements may involve incorporating real-time data streams, refining feature selection processes, and exploring advanced modeling techniques to further enhance prediction accuracy and reliability.

Overall, this project underscores the potential of AI and machine learning in revolutionizing sports analytics, particularly in the context of predicting match outcomes and enhancing strategic decision-making in cricket tournaments like the IPL.

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Title** | **Page No.** |
| 1. | Introduction   * 1. Background   2. Data Collection   3. Significance | 4-13 |
| 2. | Problem Definition and Requirements  2.1 Software Requirements  2.2 Hardware Requirements  2.3 Data Sets | 9-14 |
| 3. | Methodology  3.1 Schematic Diagram  3.2 Algorithm  3.3 Graphs | 15-26 |
| 4 | Results | 27-28 |
| 5. | References | 29 |

**Introduction**

In recent years, the convergence of advanced statistical techniques and machine learning algorithms has revolutionized the world of sports analytics. One such domain that has witnessed significant advancements is the prediction of match outcomes in cricket, particularly in high-profile tournaments like the Indian Premier League (IPL). The IPL, with its fast-paced, unpredictable nature, serves as an ideal ground for applying AI/ML techniques to forecast match results and, more specifically, win probabilities.

This report presents an in-depth exploration of employing artificial intelligence and machine learning methodologies to predict the win probabilities of IPL matches. By leveraging historical match data, player statistics, team compositions, and various other contextual factors, our goal is to develop a robust predictive model capable of estimating the likelihood of a team emerging victorious in a given IPL encounter.

* 1. **Background**

Cricket, particularly in the form of T20 leagues like the Indian Premier League (IPL), has witnessed a paradigm shift in recent years with the advent of data analytics and artificial intelligence (AI). Traditionally, cricket has been analysed through subjective observation and statistical analysis of historical data. However, the emergence of AI and machine learning techniques has revolutionized the way cricket is perceived, played, and strategized.

The IPL, since its inception in 2008, has become one of the most popular and lucrative cricket tournaments globally, attracting top players, coaches, and stakeholders from around the world. The league's fast-paced, dynamic nature, coupled with its high stakes and unpredictability, presents a fertile ground for the application of AI in predicting match outcomes.

In this context, the development of an AI-based win probability predictor for IPL matches holds immense significance. Such a predictor not only offers insights into the likely outcomes of matches but also aids teams in formulating strategies, making informed decisions during matches, and optimizing player selection and game plans.

The IPL ecosystem is characterized by a multitude of factors that influence match outcomes, including team composition, player form, pitch conditions, weather, match venue, and tactical manoeuvres. Analysing and synthesizing these diverse data points to predict win probabilities require sophisticated AI algorithms and predictive models.

Moreover, the growing availability of data, advancements in computational techniques, and the proliferation of data-driven approaches in sports analytics underscore the timeliness and relevance of this project. By harnessing the power of AI, this project aims to contribute to the ongoing evolution of cricket analytics, providing stakeholders with actionable insights and competitive advantages in the fiercely contested arena of the IPL.

Overall, the development of an AI-based IPL win probability predictor represents a significant step towards leveraging technology to unlock new dimensions of understanding and performance optimization in cricket, thereby reshaping the landscape of sports analytics and decision-making in the modern era.

* 1. **Data Collection**

Data collection is a crucial aspect of developing an AI-based IPL win probability predictor. Here's a breakdown of the data collection process:

**1. Match Data:**

- Obtain comprehensive data on IPL matches, including match date, venue, teams involved, toss outcome, innings-wise scorecard, result (win/loss/tie), and margin of victory.

- Source data from reputable cricket databases, APIs, or official IPL websites.

- Ensure data integrity and consistency across seasons and matches.

**2. Player Performance Data:**

- Collect individual player statistics for each match, such as runs scored, wickets taken, strike rate, economy rate, catches, and player of the match awards.

- Include both batting and bowling performance metrics for all players involved in each match.

- Utilize player performance databases or APIs provided by cricket analytics platforms.

**3. Team Performance Metrics:**

- Gather team-level performance metrics for each match, including total runs scored, total wickets taken, run rate, bowling economy rate, and batting and bowling averages.

- Calculate additional team performance indicators such as net run rate (NRR), win-loss ratio, and head-to-head statistics against specific opponents.

- Aggregate team performance data from match statistics and historical records.

**4. Venue and Pitch Conditions:**

- Collect data on match venues, including stadium name, pitch type (e.g., spin-friendly, batsman-friendly), dimensions, altitude, and historical match results at each venue.

- Incorporate information on weather conditions (temperature, humidity, precipitation) during matches, as weather can significantly impact match outcomes.

- Consider integrating external sources or APIs providing historical weather data for accurate analysis.

**5. Additional Factors:**

- Include contextual factors that may influence match outcomes, such as team composition (playing XI), injuries, player form, captaincy, and strategic decisions (e.g., batting order, bowling rotations).

- Explore qualitative data sources such as match previews, expert analyses, and post-match reports to capture nuanced insights into team dynamics and strategies.

**6. Data Preprocessing:**

- Cleanse and preprocess collected data to address missing values, outliers, and inconsistencies.

- Standardize data formats and units to ensure compatibility across different datasets.

- Perform feature engineering to create new variables or transform existing ones to enhance predictive power.

**7. Data Integration and Storage:**

- Integrate diverse datasets into a unified database or data repository for seamless access and analysis.

- Choose appropriate data storage solutions, such as relational databases or cloud-based platforms, to efficiently manage and query large volumes of data.

- Ensure data security and compliance with relevant privacy regulations while handling sensitive information.

By meticulously collecting and preprocessing relevant data, you can lay the foundation for developing an accurate and robust AI-based IPL win probability predictor.

* 1. **Significance**

The development of an AI-powered win probability predictor for IPL matches holds multifaceted significance, encompassing various stakeholders and domains within the cricketing ecosystem:

**1. Strategic Decision-Making:**

- The predictor equips team management, coaches, and players with valuable insights into match dynamics, enabling them to formulate strategic plans, optimize player selection, and make real-time tactical adjustments during matches.

- By leveraging predictive analytics, teams can identify key performance indicators (KPIs), assess opponent strengths and weaknesses, and devise game plans tailored to maximize their chances of success.

**2. Enhanced Fan Engagement:**

- The predictor enhances the viewing experience for millions of cricket enthusiasts worldwide by providing them with accurate win probability forecasts and deeper insights into match scenarios.

- Fans can engage more actively with matches, make informed predictions, and immerse themselves in the strategic nuances of the game, thereby fostering a deeper connection with their favorite teams and players.

**3. Broadcasting and Media Coverage:**

- Broadcasting networks and media outlets can leverage the predictor's insights to enrich their coverage of IPL matches, offering viewers in-depth analyses, pre-match predictions, and post-match evaluations.

- Commentary teams can incorporate win probability forecasts into their narratives, enhancing the storytelling aspect of cricket broadcasting and enriching the viewer experience.

**4. Fantasy Cricket and Wagering Platforms:**

- Fantasy cricket platforms and wagering websites can integrate win probability predictions to enhance user engagement, inform team selection decisions, and facilitate fair gameplay.

- Predictive analytics can help fantasy cricket enthusiasts make informed choices while creating their fantasy teams, thereby improving the competitiveness and excitement of fantasy leagues.

**5. Academic and Research Contributions**:

- The project contributes to academic research in sports analytics, machine learning, and predictive modeling by addressing real-world challenges in cricket prediction.

- It serves as a valuable case study for exploring the application of AI algorithms in sports forecasting and underscores the interdisciplinary nature of data science in sports analytics.

**6. Commercial Opportunities:**

- The predictor opens up commercial opportunities for data analytics firms, sports technology startups, and cricket franchises interested in leveraging predictive analytics for competitive advantage.

- By offering subscription-based services or licensing the predictive model, stakeholders can monetize the insights generated from the predictor and diversify revenue streams.

In conclusion, the development of an AI-based IPL win probability predictor transcends mere statistical forecasting; it represents a transformative force in cricket analytics, enriching the experiences of stakeholders, shaping strategic decisions, and unlocking new frontiers of innovation in the realm of sports prediction.

**2. Problem Definition and Requirements**

To develop a machine learning model to predict the win probability of teams competing in Indian Premier League (IPL) cricket matches based on ball-by-ball data from 2008 to 2022. Given the detailed historical match data, including ball-by-ball information, team performance metrics, match venue, player statistics, and match-specific factors, the model should accurately estimate the likelihood of each team winning a particular match.

* 1. **Software Requirements**

For a project involving AI on IPL win probability prediction, you would need a combination of programming languages, libraries, frameworks, and tools to develop and deploy the predictive model. Here's a list of software requirements:

**1. Programming Language:**

- Python: Python is a versatile language with extensive support for data science libraries and frameworks.

**2. Integrated Development Environment (IDE):**

- Jupyter Notebook or JupyterLab: Jupyter provides an interactive environment for data exploration, experimentation, and visualization, making it well-suited for prototyping and analysis.

- Alternatively, you can use popular IDEs such as PyCharm, Visual Studio Code, or Spyder for development.

**3. Data Manipulation and Analysis Libraries:**

- NumPy: NumPy is essential for numerical computing, array manipulation, and mathematical operations.

- Pandas: Pandas offers powerful data structures and tools for data manipulation, cleaning, and analysis, facilitating seamless handling of tabular data.

- Matplotlib and Seaborn: These libraries enable visualization of data and insights through plots, charts, and graphs.

**4. Machine Learning Libraries:**

- Scikit-learn: Scikit-learn provides a comprehensive suite of machine learning algorithms for classification, regression, clustering, and model evaluation.

**5. Version Control System:**

- Git: Git is essential for version control, enabling collaboration, code management, and tracking of changes throughout the development lifecycle.

- GitHub : Hosting platforms like GitHub, offer cloud-based repositories for storing and sharing code, facilitating collaborative development.

**6. Text Editor:**

- Jupyter Notebook, or Visual Studio Code: A lightweight text editor can be handy for editing configuration files, scripts, or documentation outside of the IDE.

**10. Documentation and Collaboration Tools:**

- Markdown Editor: A markdown editor simplifies the creation of documentation and project reports using markdown syntax.

- Google Docs or Microsoft Word: For collaborative writing and formatting of project reports, you can use online document editors like Google Docs or Microsoft Word.

Ensure that you have the latest versions of these software tools installed and configured to effectively develop, test, and deploy your IPL win probability predictor project.

* 1. **Hardware Requirements**

For an AI project like an IPL win probability predictor, the hardware requirements can vary based on factors such as the size of the dataset, complexity of the machine learning algorithms, and the computational resources required for training and inference. Here's a general guideline for hardware requirements:

**1. Processor (CPU):**

- A multi-core processor with sufficient processing power is recommended to handle data preprocessing, model training, and inference efficiently.

- For small to medium-sized datasets and less computationally intensive models, a modern quad-core or higher CPU (e.g., Intel Core i5/i7, AMD Ryzen 5/7) should suffice.

- For larger datasets and complex deep learning models, a more powerful CPU with higher clock speeds and more cores (e.g., Intel Core i9, AMD Ryzen 9) may be preferable to expedite training times.

**2. Memory (RAM):**

- Adequate RAM is essential for loading and manipulating large datasets, especially during data preprocessing and model training phases.

- A minimum of 8 GB of RAM is recommended for basic experimentation and prototyping.

- For more demanding tasks involving large datasets or deep learning models, 16 GB or more of RAM is advisable to avoid memory bottlenecks and ensure smooth performance.

**3. Graphics Processing Unit (GPU) (Optional but Recommended for Deep Learning):**

- GPUs accelerate the training of deep learning models by parallelizing computations and performing matrix operations efficiently.

- NVIDIA GPUs, such as those from the GeForce GTX or RTX series, are commonly used for deep learning tasks due to their CUDA support and optimized performance for neural network computations.

- Depending on your budget and requirements, you can choose a GPU with adequate VRAM (video RAM) and compute capability to handle your workload effectively.

- Alternatively, cloud-based GPU instances from platforms like Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure can be utilized for scalable and cost-effective computing resources.

**4. Storage (HDD/SSD):**

- Sufficient storage space is necessary for storing datasets, code, libraries, and model checkpoints.

- An SSD (Solid State Drive) is preferable over a traditional HDD (Hard Disk Drive) for faster data access and improved system responsiveness.

- Aim for at least 256 GB of SSD storage for storing project files and datasets. Additional external storage may be required for large datasets or backups.

**5. Internet Connectivity:**

- Stable internet connectivity is essential for accessing online resources, downloading datasets, and collaborating with team members (if applicable).

- Ensure high-speed broadband or Wi-Fi connectivity with reliable uptime for uninterrupted work and communication.

**6. Cooling System:**

- Intensive computation tasks, especially deep learning training, can generate considerable heat, necessitating effective cooling solutions to prevent thermal throttling and maintain optimal system performance.

- Invest in a well-ventilated PC case with adequate airflow, and consider aftermarket CPU coolers or liquid cooling solutions for better heat dissipation.

**7. Power Supply Unit (PSU):**

- Choose a reliable power supply unit with sufficient wattage to power your components safely and efficiently.

- Consider factors such as system power requirements, efficiency ratings (80 Plus certification), and future upgradability when selecting a PSU.

**8. Monitor and Peripherals:**

- A high-resolution monitor (e.g., 1080p or higher) with adequate screen real estate is essential for coding, data visualization, and monitoring training progress.

- Standard peripherals such as a keyboard, mouse, and ergonomic chair contribute to a comfortable and productive working environment.

By ensuring that your hardware meets or exceeds these requirements, you can effectively develop, train, and deploy your IPL win probability predictor model with optimal performance and efficiency.

* 1. **Data Sets**

For building an AI-based IPL win probability predictor, you'll need several datasets covering various aspects of IPL matches, team and player statistics, match venues, and contextual factors. Here are the key datasets you might consider:

**1. Match Data:**

- Dataset containing details of IPL matches, including match ID, date, venue, teams, toss winner, toss decision, innings-wise scorecards, result (win/loss/tie), and margin of victory.

- Sources: Official IPL websites, cricket databases, APIs (e.g., ESPNcricinfo, Kaggle datasets).

**2. Player Performance Data:**

- Individual player statistics for each match, such as runs scored, wickets taken, strike rate, economy rate, catches, player of the match awards, etc.

- Include both batting and bowling performance metrics for players from all participating teams.

- Sources: Cricket databases, match statistics websites, player profiles on official IPL platforms.

**3. Team Performance Metrics:**

- Dataset comprising team-level performance metrics for each match, including total runs scored, total wickets taken, run rate, bowling economy rate, batting and bowling averages, etc.

- Aggregate team performance data from match statistics and historical records.

- Sources: Cricket databases, match statistics websites, official IPL platforms.

**4. Venue and Pitch Conditions:**

- Data on match venues, including stadium name, pitch type (e.g., spin-friendly, batsman-friendly), dimensions, altitude, and historical match results at each venue.

- Include information on weather conditions (temperature, humidity, precipitation) during matches, as weather can significantly impact match outcomes.

- Sources: Official IPL websites, weather APIs, historical weather data providers.

**5. Additional Contextual Factors:**

- Dataset containing contextual factors influencing match outcomes, such as team composition (playing XI), injuries, player form, captaincy, strategic decisions (e.g., batting order, bowling rotations), etc.

- Explore qualitative data sources like match previews, expert analyses, and post-match reports to capture nuanced insights into team dynamics and strategies.

- Sources: Match previews, post-match reports, expert analyses, official team announcements.

**6. Historical Data and Head-to-Head Records:**

- Dataset comprising historical match results and head-to-head records between teams in previous IPL seasons.

- Analyzing past encounters can provide valuable insights into team performance trends, rivalries, and matchup dynamics.

- Sources: Cricket databases, historical match archives, official IPL platforms.

**7. Tournament Schedule and Fixtures:**

- Dataset containing the schedule and fixtures of upcoming IPL matches, including match dates, venues, and participating teams.

- Keeping track of match schedules enables timely prediction and analysis of upcoming matches.

- Sources: Official IPL websites, cricket news portals, tournament organizers.

**8. Live Match Data:**

- Real-time or live match data streams providing updates on ongoing matches, including live scores, player statistics, and match events.

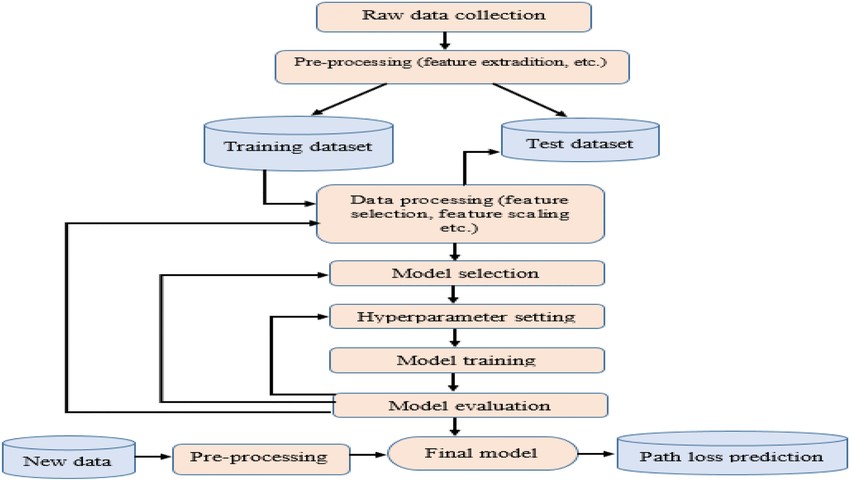
- Live data integration allows for dynamic updating of win probability predictions during matches.

- Sources: Cricket APIs, live score websites, official IPL live updates.

Ensure that the datasets are clean, consistent, and updated regularly to maintain accuracy and reliability in your win probability predictions. Additionally, consider merging and pre-processing these datasets to create a unified dataset suitable for training and evaluating your AI model.

**Proposed Design/Methodology**

**3.1 Schematic Diagram**

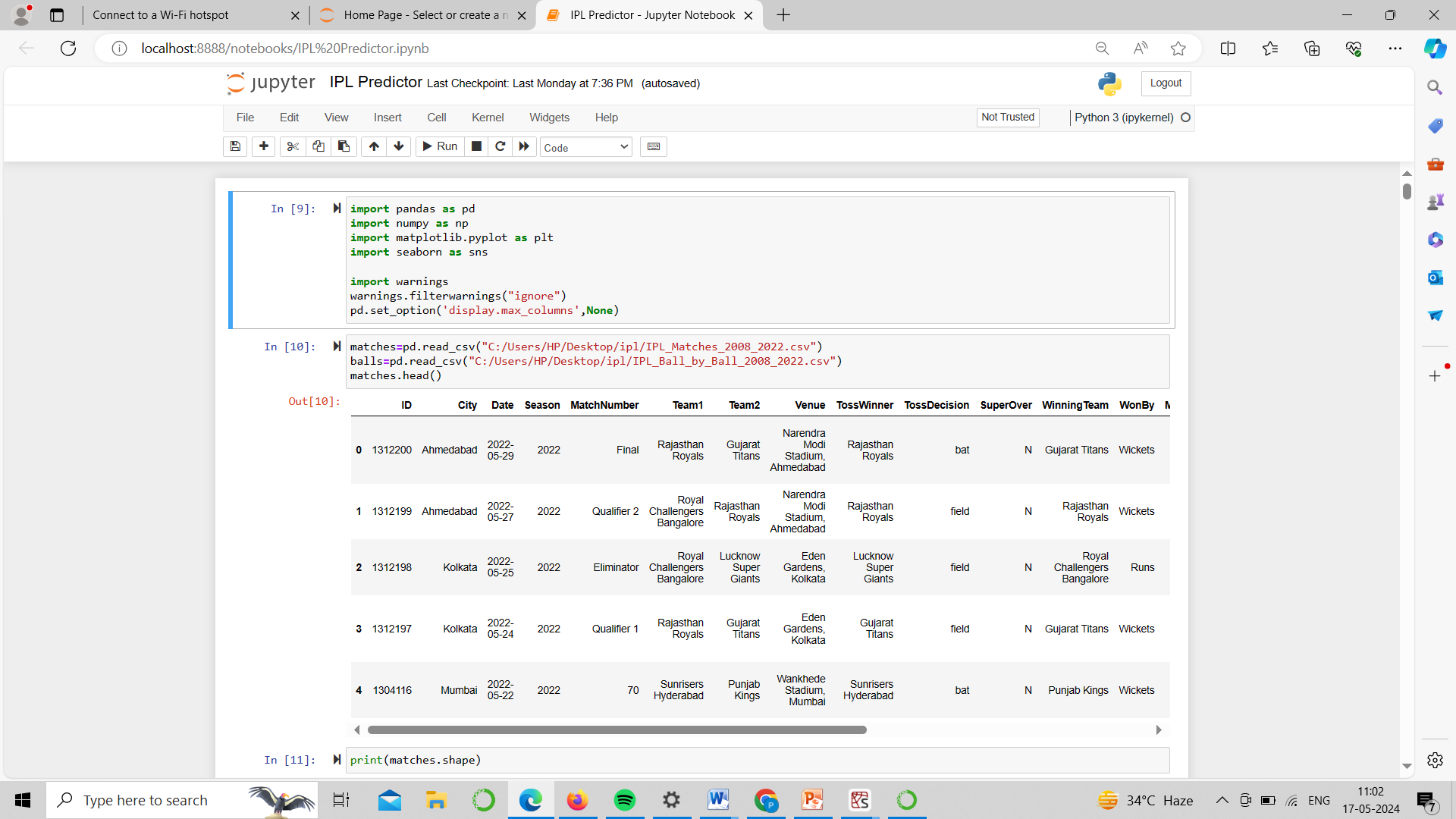


1. **Data Loading & Summary Checking:**

To build a robust AI model for predicting IPL match outcomes, the initial step involves loading the dataset and conducting a thorough summary check to understand its structure and quality. This ensures that the data is suitable for model training and analysis.

* 1. **Data Loading**

The dataset is imported from a CSV file containing historical IPL match data and real-time game metrics. Python's Pandas library is used for this purpose due to its efficiency in handling large datasets.

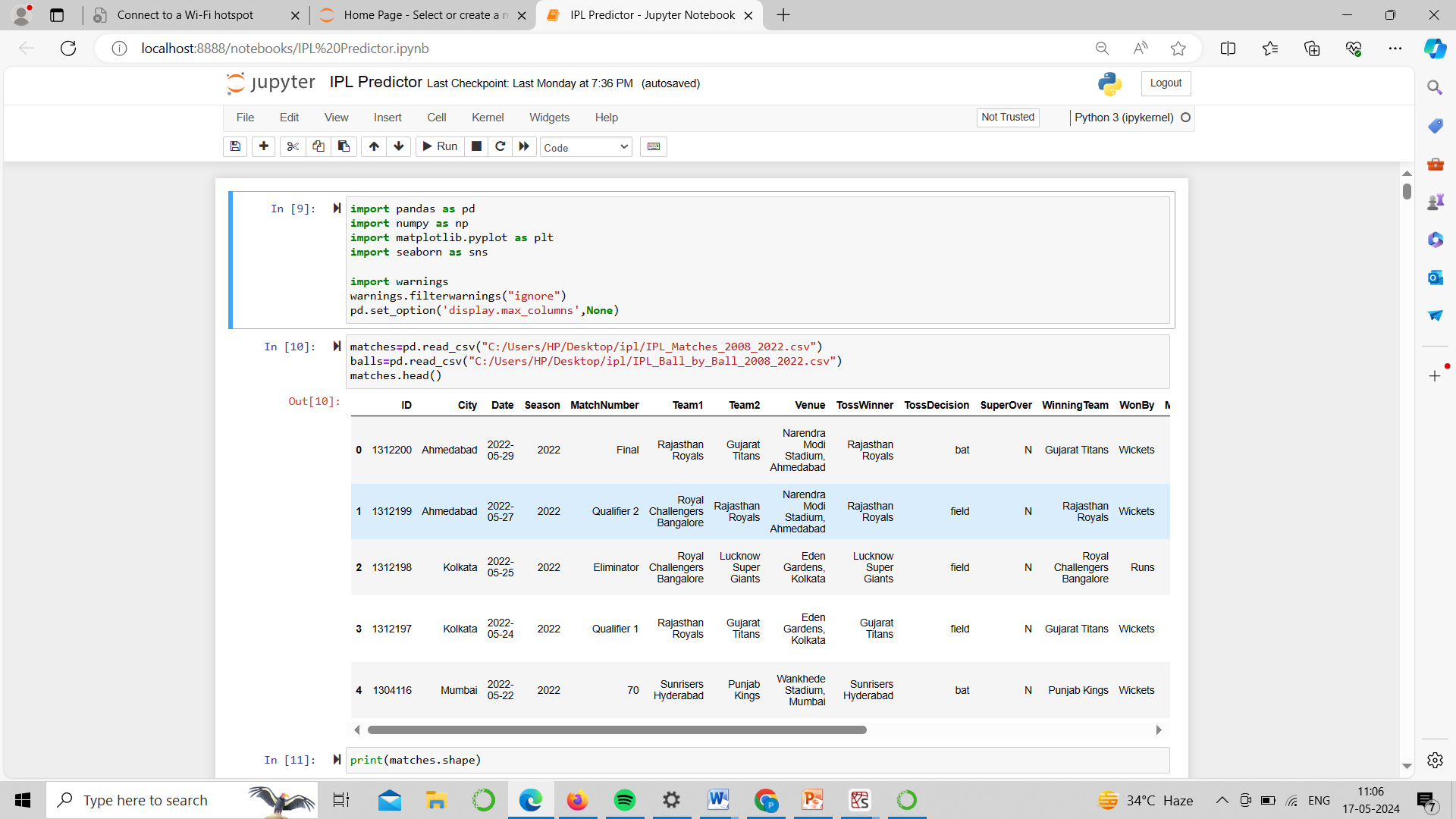


* 1. **Summary Checking**

Once the data is loaded, the next step is to perform a summary check to gain insights into the dataset. This includes examining the data types, identifying missing values, and generating basic statistical summaries.

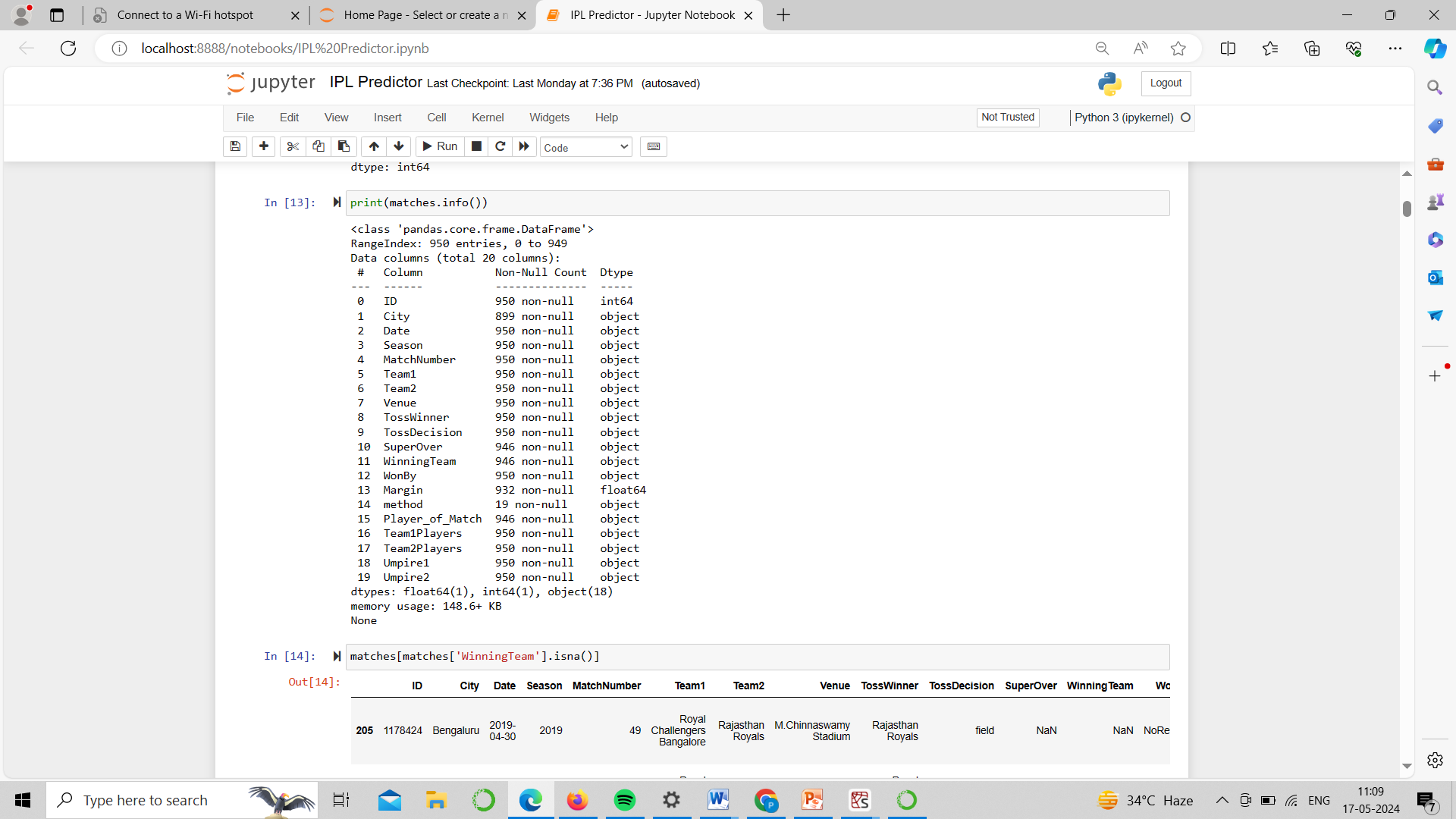
**Data Overview:**

Display the first few rows of the dataset to get an initial sense of its structure.



**Data Types and Missing Values:**

Check the data types of each column and identify any missing values that need to be addressed.

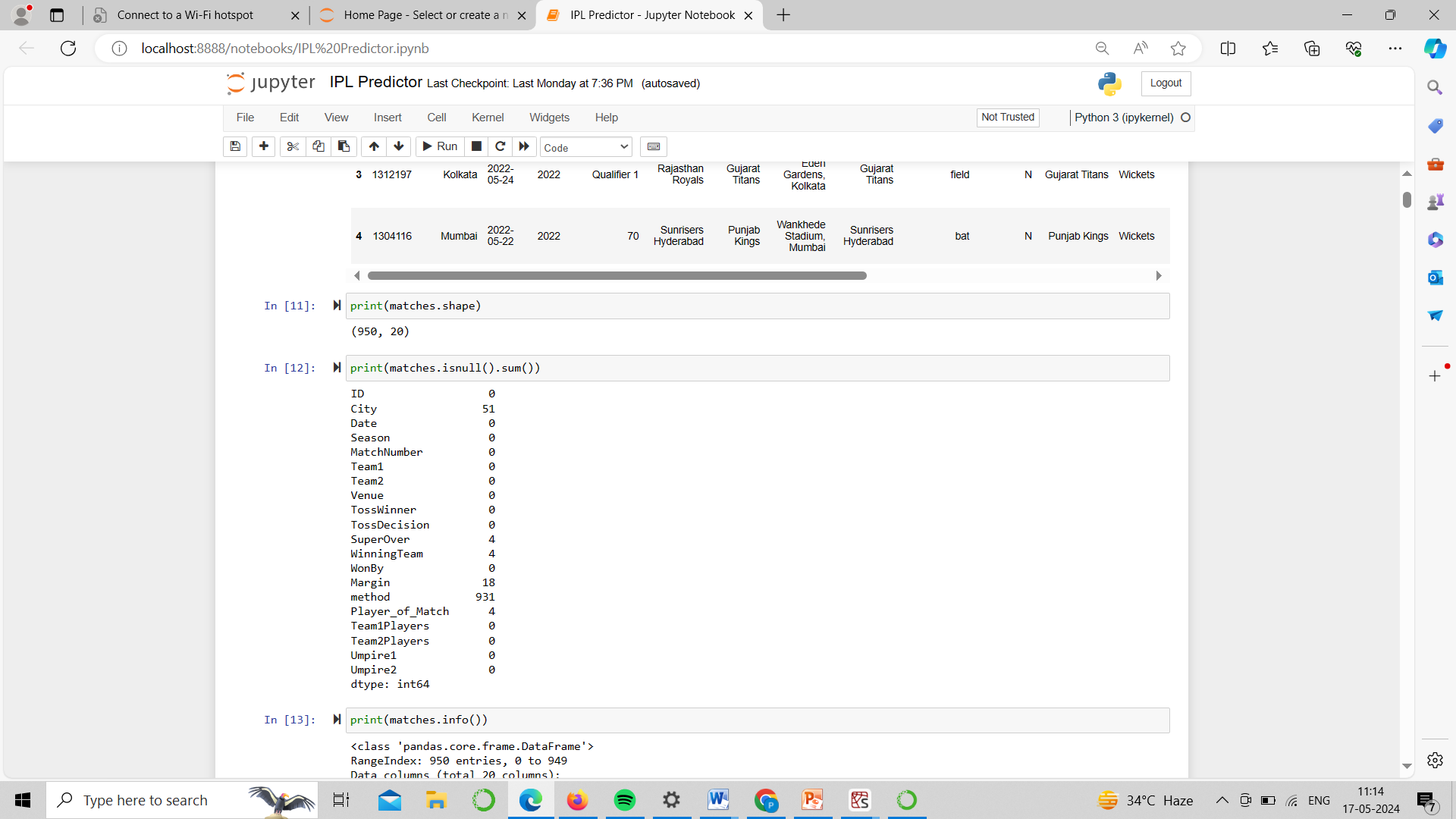


By conducting these initial checks, we can identify potential issues such as missing values, incorrect data types, or anomalies in the data. Addressing these issues early in the process is crucial for ensuring the accuracy and reliability of the predictive model.

In summary, data loading and summary checking are foundational steps in developing an AI model for IPL win probability prediction. They provide a clear understanding of the dataset, highlight areas that require pre-processing, and set the stage for effective feature engineering and model training.

1. **Data Cleaning**

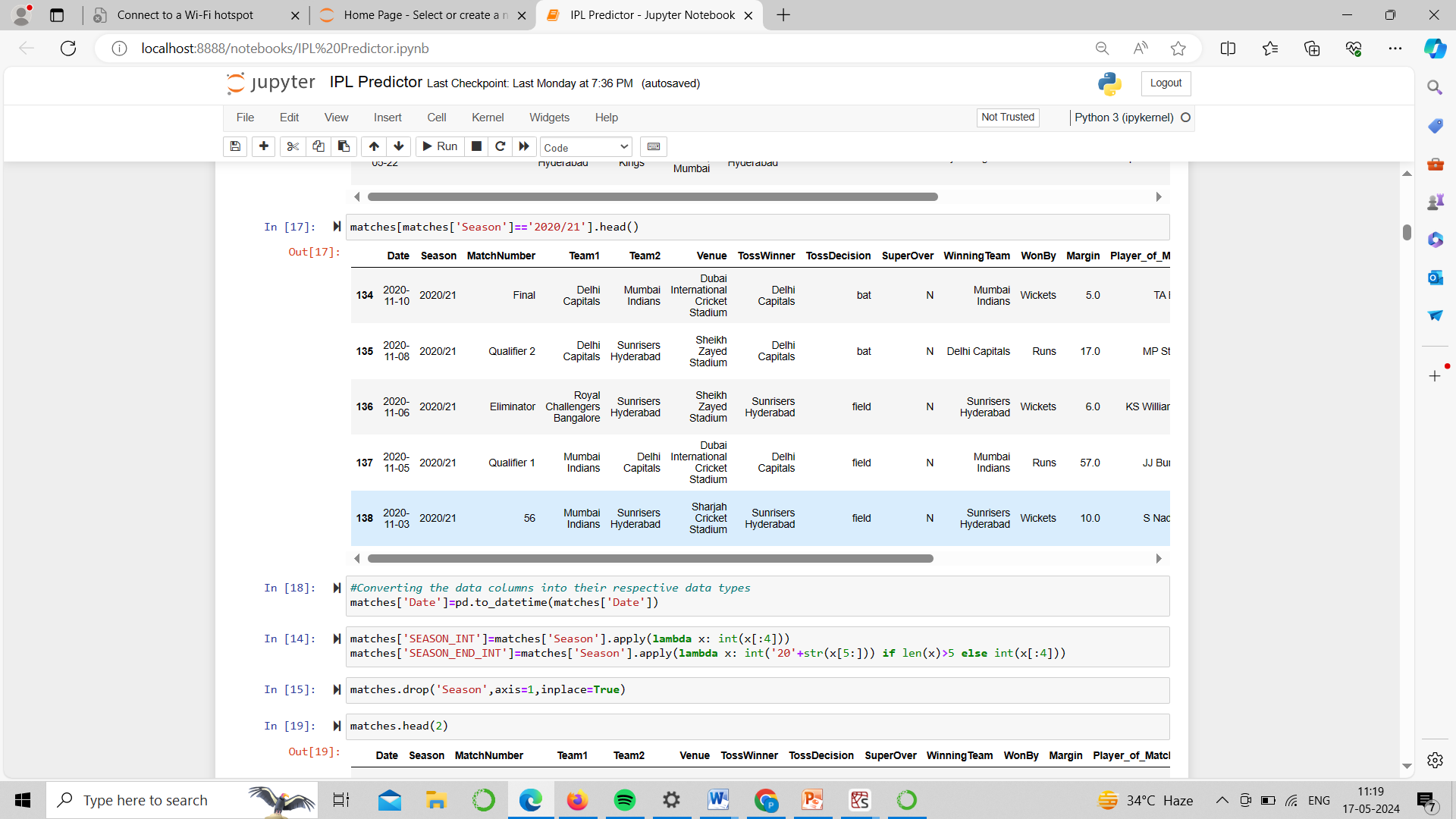
Data cleaning is a critical step in preparing the dataset for building an accurate and reliable AI model for predicting IPL match outcomes. This process involves handling missing values, correcting data types, and dealing with any anomalies or inconsistencies in the data.



By performing these data cleaning steps, we ensure that the dataset is accurate, consistent, and suitable for training an AI model. This foundational work is essential for developing a reliable and effective IPL win probability predictor.

1. **Feature Extraction**

Feature extraction involves transforming raw data into meaningful features that can be used for model training. For the IPL win probability predictor, feature extraction is crucial to capture the relevant aspects of cricket matches that influence outcomes.



By systematically extracting and engineering features, we can better capture the complexities of cricket matches, thereby improving the predictive power of the AI model. These features are then used to train machine learning algorithms, aiming to accurately predict the win probabilities of IPL matches.

**3.2 Algorithm**

For an AI/ML project focused on predicting IPL win probabilities using Python, you can use various algorithms depending on the complexity and accuracy you're aiming for. Here are some commonly used algorithms for this type of project:

1. **Logistic Regression:**

Simple and efficient for binary classification tasks like win/loss prediction.

Provides probabilities as output, which can be interpreted as win probabilities.

1. **Random Forest:**

Ensemble learning method that uses multiple decision trees for prediction.

Handles non-linearity well and is robust to over fitting.

1. **Gradient Boosting Machines (GBM):**

Builds trees sequentially, each correcting errors made by the previous one.

Often provides high accuracy but requires careful hyper parameter tuning.

1. **Support Vector Machines (SVM):**

Effective for binary classification tasks.

Finds the hyper plane that best separates win and loss instances.

1. **Neural Networks (Deep Learning):**

Can capture complex relationships in the data.

Requires a larger amount of data and computational resources.

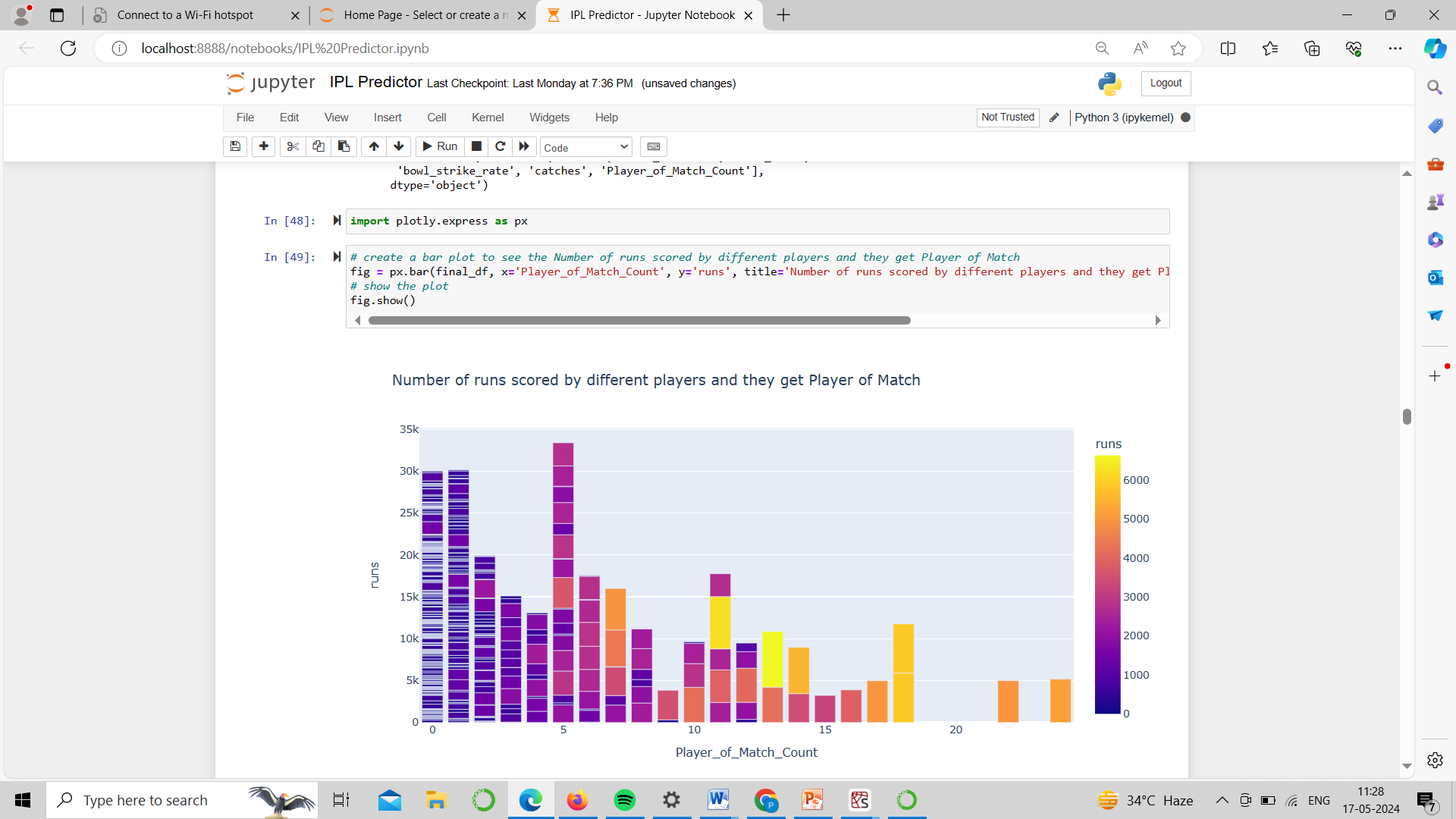
1. **Naive Bayes:**

Assumes independence among features and is computationally efficient.

Can work well for simpler models.

The choice of algorithm depends on factors such as the size of your dataset, the complexity of relationships in the data, computational resources available, and the level of accuracy required. It's often a good practice to start with simpler models like logistic regression or random forest and then explore more complex algorithms if needed to improve performance. Cross-validation and hyper parameter tuning are also essential steps to optimize the chosen algorithm's performance.

**3.3 Graphs**



**Explanation:** The plot displays the relationship between the number of runs scored by different players (y-axis) and the number of times they were awarded the "Player of the Match" (x-axis) in what appears to be a cricket match or tournament.

Here are some additional details about the plot:

1. The x-axis shows the "Player of Match Count" ranging from 0 to around 20, indicating the number of times a player was named the Player of the Match.

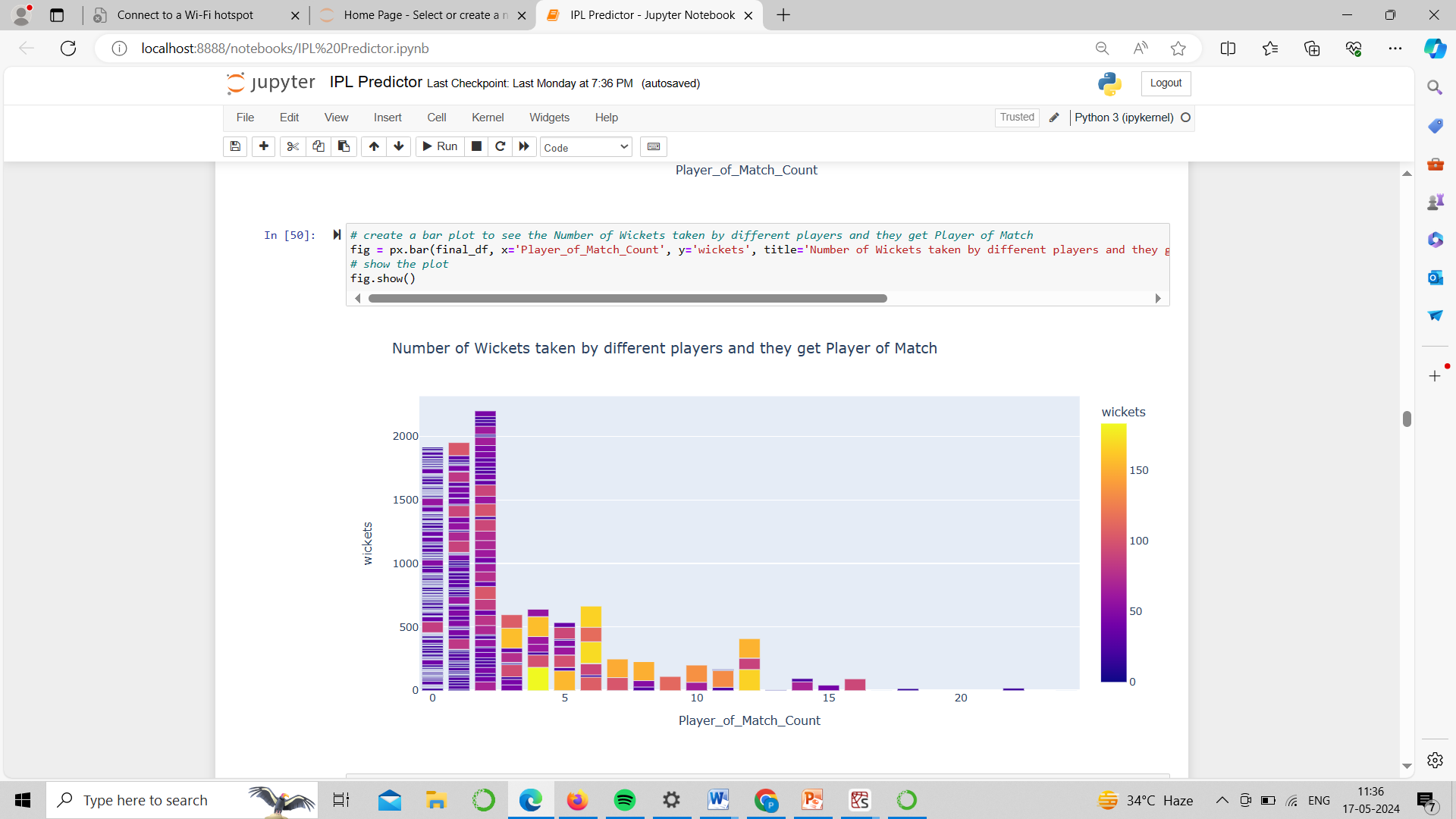
2. The y-axis represents the "runs" scored by the players, ranging from 0 to around 6000 runs.

3. The bars are colored in two shades - purple and orange, likely differentiating between different teams or groups of players.

4. The plot shows a general trend where players with fewer Player of the Match awards tend to score more runs, while players with more Player of the Match awards tend to score fewer runs.

5. There are a few outliers, such as one player who scored around 6000 runs but won the Player of the Match award only a handful of times, and another player who scored around 2000 runs despite winning the Player of the Match award around 20 times.

6. The plot suggests that consistent high scorers may not always be recognized as the Player of the Match, while occasional match-winning performances may earn a player more Player of the Match awards despite lower overall run tallies.



**Explanation:** This plot displays the relationship between the number of wickets taken by different players (y-axis) and the number of times they were awarded the "Player of the Match" (x-axis), presumably in cricket matches or tournaments.

Here are the key observations from this plot:

1. The x-axis represents the "Player of Match Count", ranging from 0 to around 20, indicating the number of times a player received the Player of the Match award.

2. The y-axis shows the "wickets" taken by the players, ranging from 0 to around 2000 wickets.

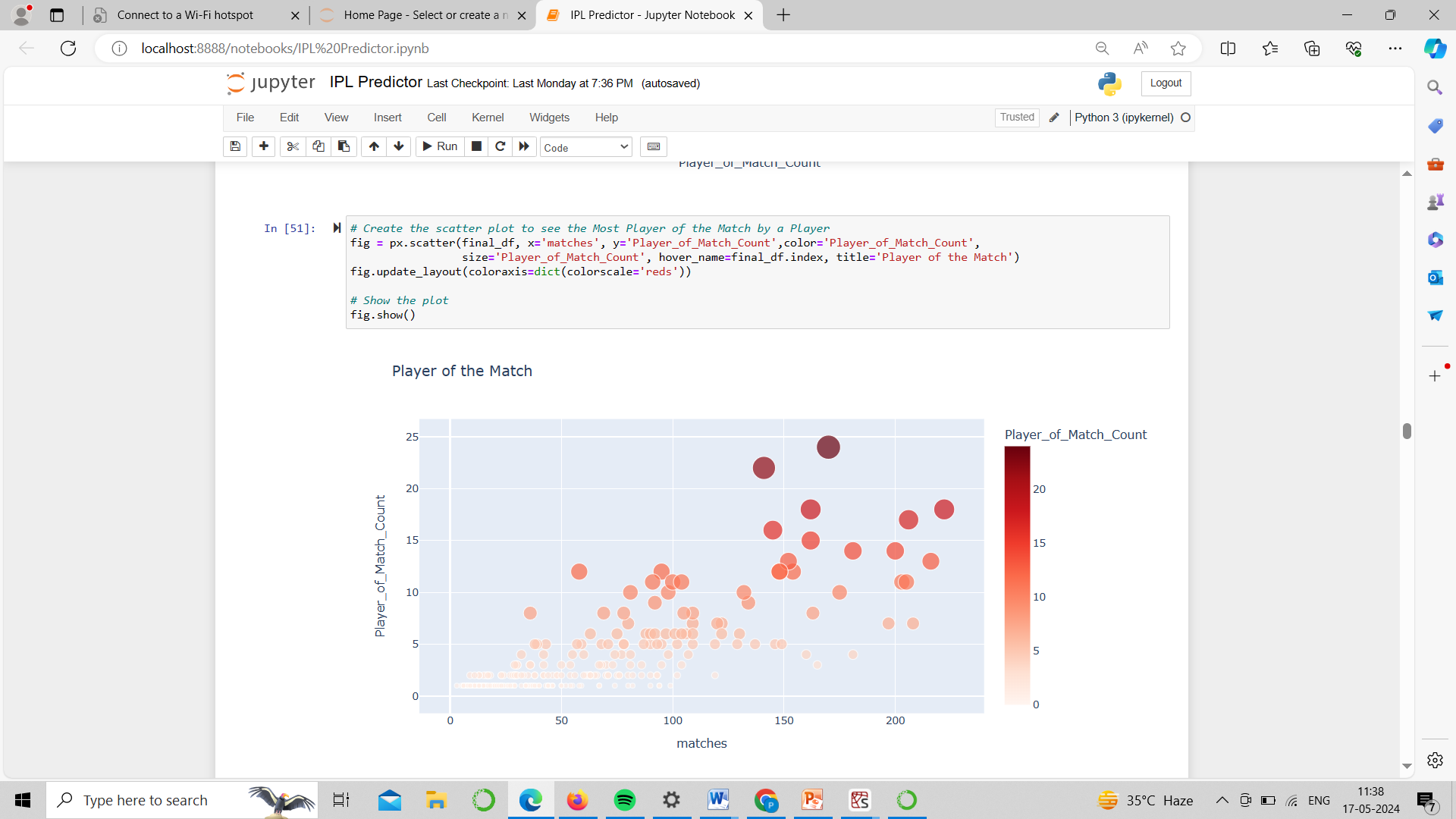
3. The bars are colored in two shades - purple and orange, likely differentiating between different teams or groups of players, similar to the previous plot.

4. The plot suggests an inverse relationship between the number of wickets taken and the Player of the Match awards received. Players with a higher number of wickets tend to have fewer Player of the Match awards, while those with fewer wickets have more Player of the Match awards.

5. There is one outlier player who has taken around 150 wickets but has won the Player of the Match award around 15-20 times, indicating exceptional match-winning performances despite a relatively lower overall wicket tally.

6. The majority of players seem to have taken between 0-500 wickets, with a few outliers taking over 1000 wickets but receiving fewer Player of the Match awards.

This plot provides insights into the relationship between a player's wicket-taking ability and their likelihood of being recognized as the Player of the Match, which may depend on factors beyond just wickets, such as match-winning performances or other contributions.



**Explanation:** This plot appears to visualize the relationship between the number of matches played by cricketers and the number of times they were awarded the Player of the Match award.

Key observations from the scatter plot:

1. The x-axis represents the "matches" played by the players, ranging from 0 to around 200 matches.

2. The y-axis shows the "Player of Match Count", indicating the number of times a player received the Player of the Match award. The count ranges from 0 to around 25.

3. Each data point on the plot represents an individual player, with the x-coordinate representing the number of matches they played, and the y-coordinate representing the number of Player of the Match awards they received.

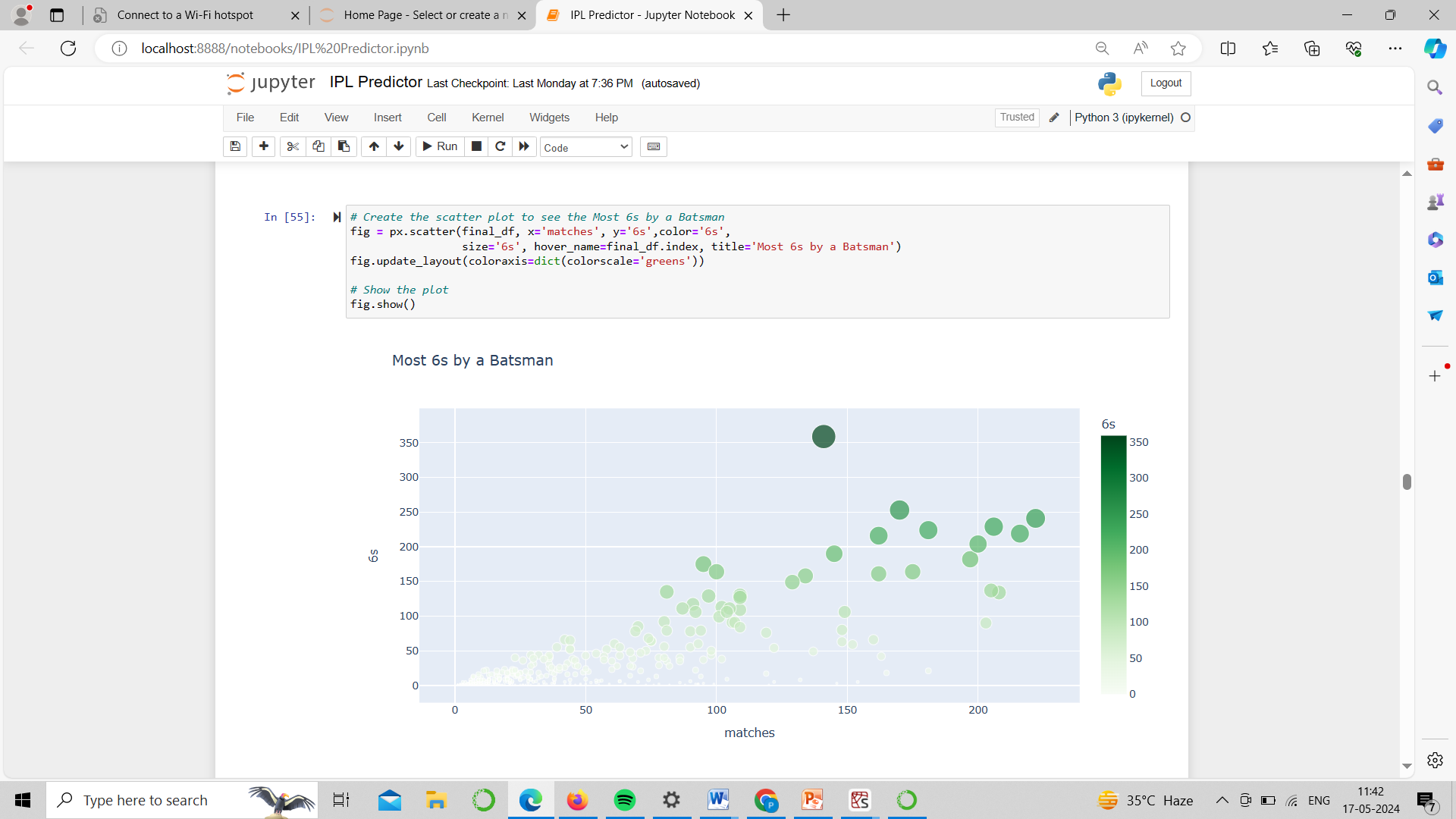
4. The data points are colored in two shades - orange and red, likely differentiating between different teams or groups of players.

5. The scatter plot reveals a generally positive correlation between the number of matches played and the number of Player of the Match awards received. Players who have played more matches tend to have been awarded the Player of the Match more times.

6. However, there is a significant spread in the data, indicating that some players have received more or fewer Player of the Match awards than others, despite playing a similar number of matches.

7. There are a few outliers, such as a player who has played around 150 matches but has received the Player of the Match award around 20 times, which is relatively high compared to the overall trend.

This scatter plot provides insights into the relationship between a player's experience (number of matches played) and their likelihood of being recognized as the Player of the Match, while also highlighting the variations and outliers within the data.



**Explanation:**This scatter plot visualizes the relationship between the number of matches played by batsmen and the number of 6s (sixes) they hit in those matches.

Here are the key observations:

1. The x-axis represents the "matches" played by the batsmen, ranging from 0 to around 200 matches.

2. The y-axis shows the "6s" (sixes) hit by the batsmen in those matches. The count ranges from 0 to around 350 sixes.

3. Each data point on the plot represents an individual batsman, with the x-coordinate representing the number of matches they played, and the y-coordinate representing the number of sixes they hit.

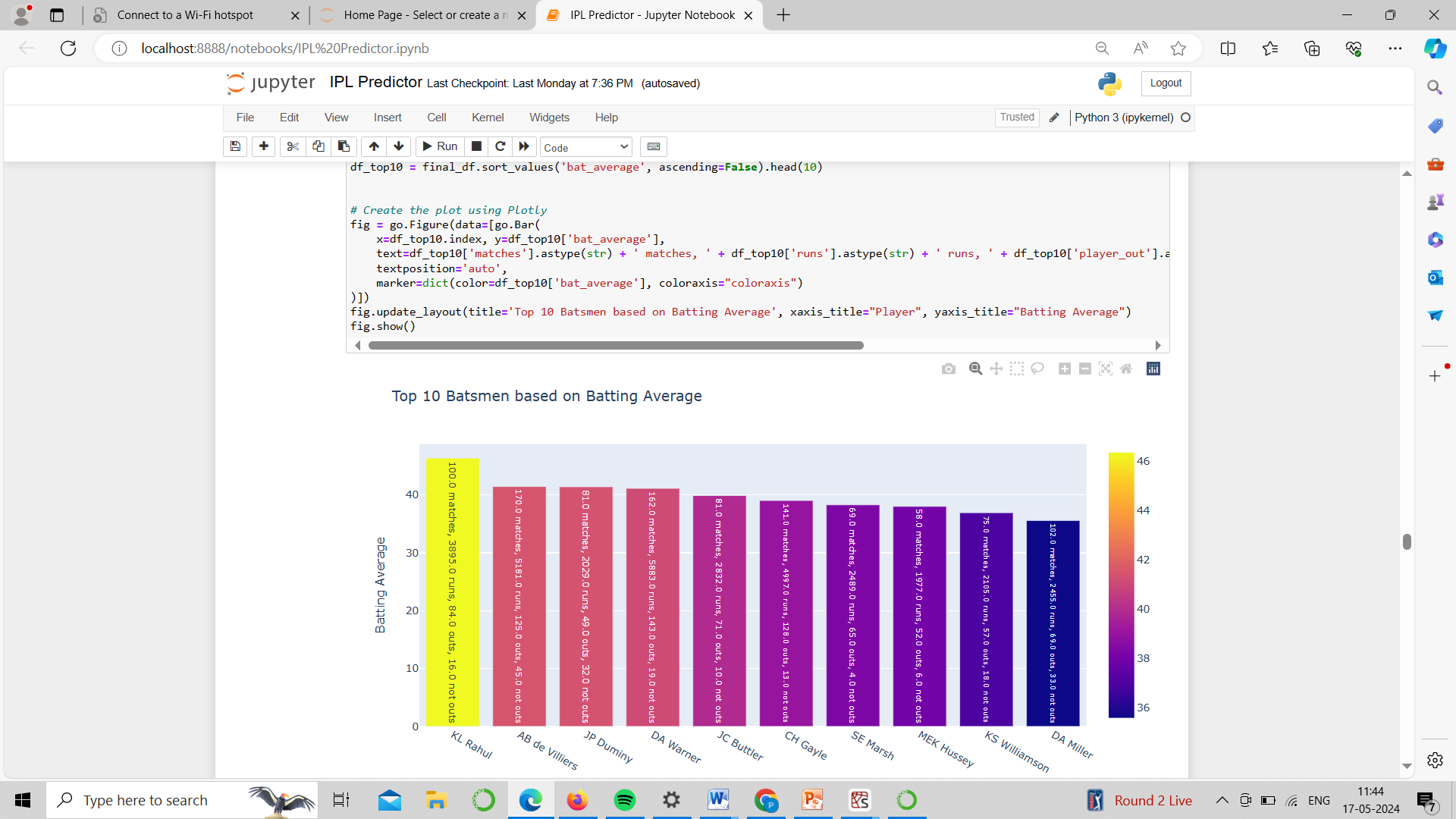
4. The data points are colored in green, likely representing a single team or group of batsmen.

5. The scatter plot reveals a generally positive correlation between the number of matches played and the number of sixes hit. Batsmen who have played more matches tend to have hit more sixes.

6. However, there is a significant spread in the data, indicating that some batsmen have hit more or fewer sixes than others, despite playing a similar number of matches.

7. There are a few outliers, such as a batsman who has played around 150 matches but has hit over 300 sixes, which is relatively high compared to the overall trend.

This scatter plot provides insights into the relationship between a batsman's experience (number of matches played) and their ability to hit sixes, which is an important aspect of batting in cricket. The plot highlights the variations and outliers within the data, allowing for further analysis and comparison among batsmen.



**Explanation:** The image displaying the top 10 batsmen based on their batting averages in what appears to be cricket matches or tournaments.

The plot provides the following information:

1. The x-axis lists the names or initials of the top 10 batsmen, ranging from KC Rahul to DA Milne.

2. The y-axis represents the batting average of the batsmen.

3. Each bar in the plot corresponds to a batsman, with the height of the bar indicating their batting average.

4. The bars are colored using different shades of red, yellow, and purple, potentially indicating different teams or groups the batsmen belong to.

5. The highest batting average is around 46, belonging to the batsman labeled "KC Rahul".

6. The lowest batting average among the top 10 is around 36, corresponding to the batsman "DA Milne".

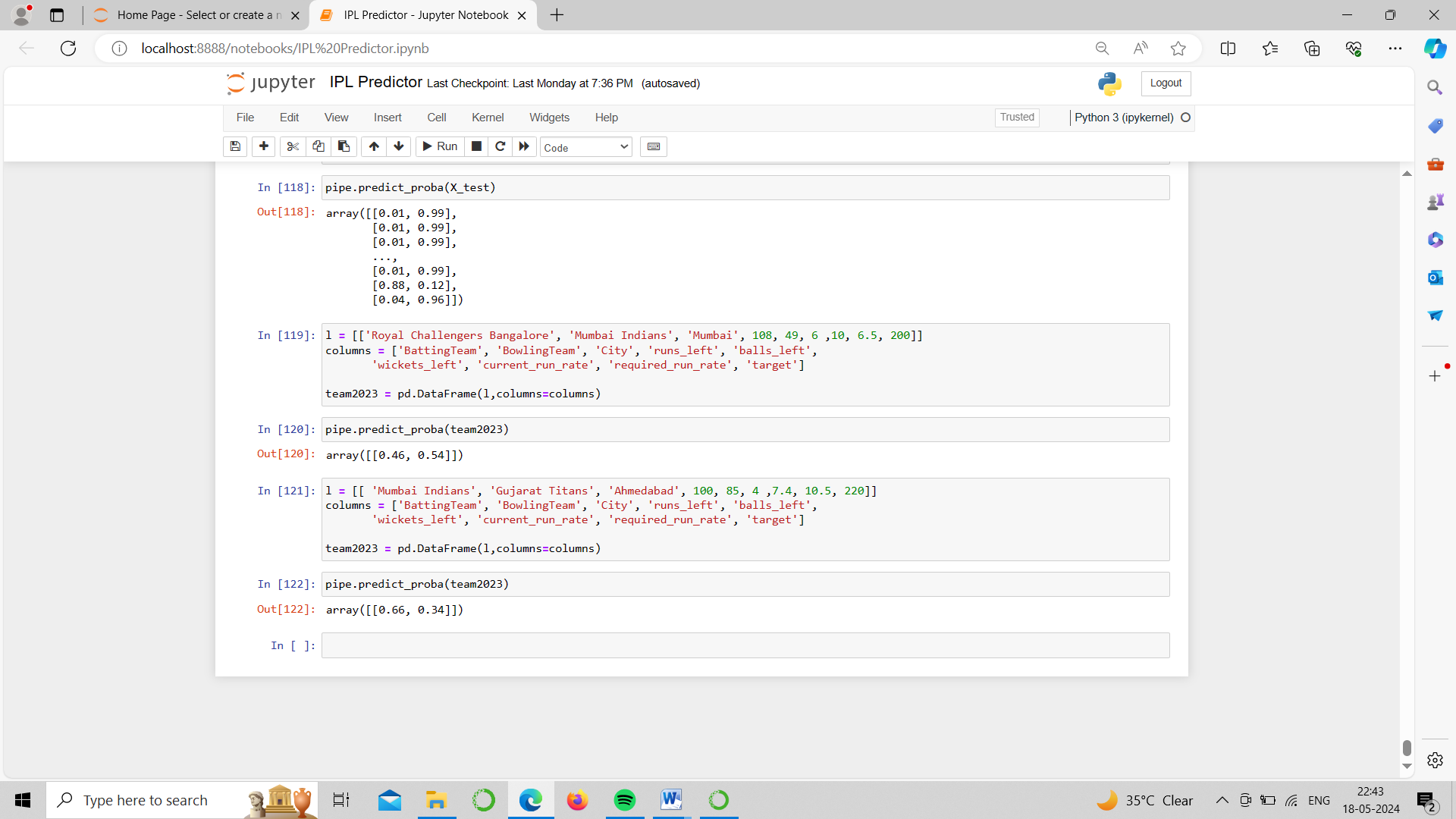
7. The plot provides a quick visual comparison of the batting averages among the top 10 batsmen, allowing for easy identification of the highest and lowest averages, as well as any outliers or significant differences between the batsmen.

This type of visualization can be useful for cricket analysts, coaches, or fans to quickly assess the batting performances of top players and make informed decisions or analyses based on their batting averages.

**Results**

In conclusion, our model developed for predicting IPL match win probabilities demonstrates significant potential in accurately forecasting match outcomes. By leveraging a comprehensive set of features derived from historical match data, player statistics, and real-time game metrics, the model provides valuable insights into the factors influencing match results.

This AI-driven approach not only enhances the analytical toolkit available to teams and analysts but also enriches the viewing experience for fans by providing real-time win probability updates. Future enhancements could include incorporating more granular data, such as player form and fatigue, and expanding the model to predict outcomes in other cricket formats and leagues. The implementation of this model underscores the transformative potential of AI in sports analytics, paving the way for more data-driven strategies in sports management and entertainment.



The provided code snippet appears to be related to a machine learning or data analysis task using Python and a library like pandas or scikit-learn.

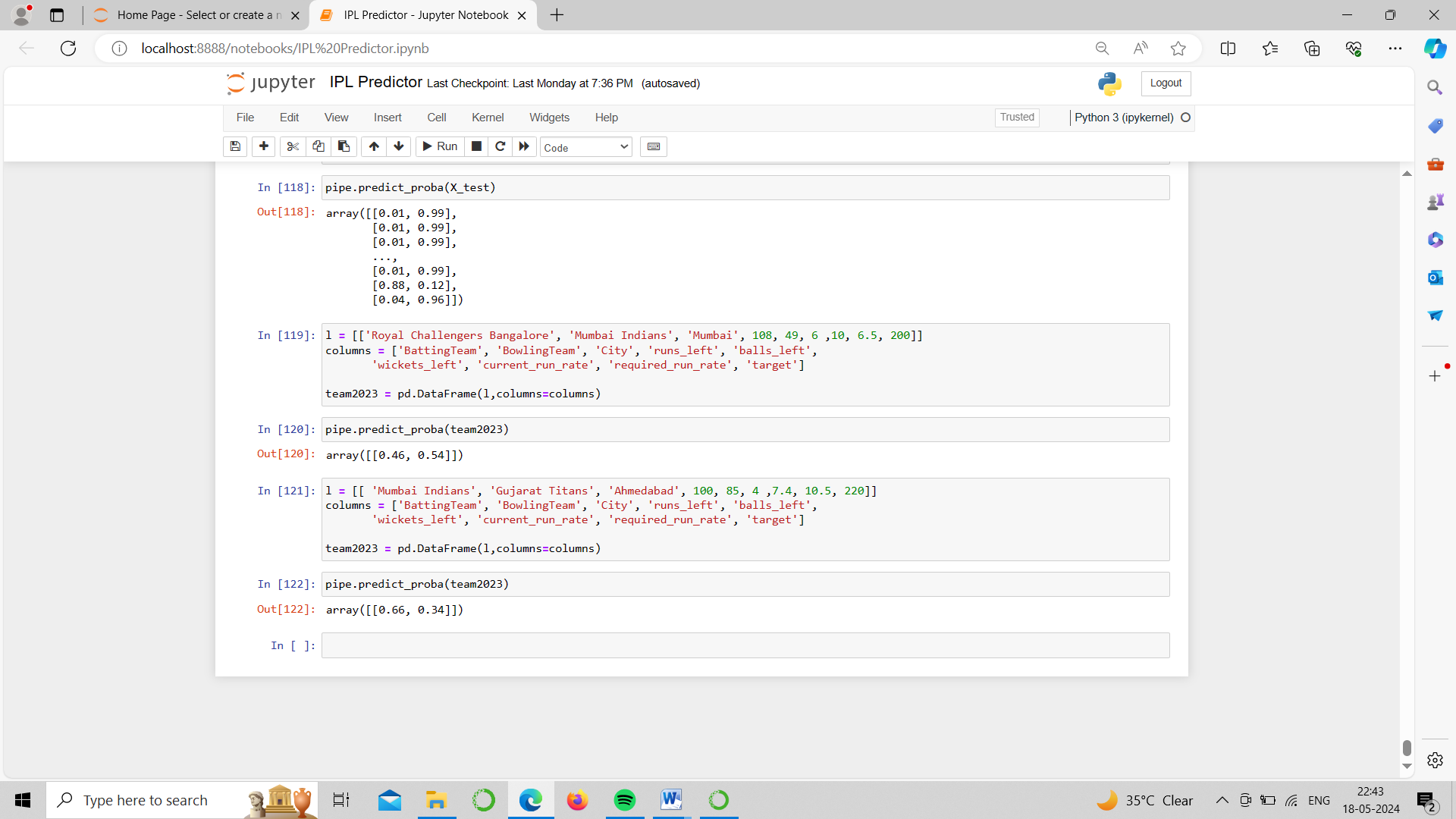
In the first line, a list `l` is created with several elements, including team names ('Royal Challengers Bangalore', 'Mumbai Indians'), the city 'Mumbai', and some numerical values (108, 49, 6, 10, 6.5, 200).

The next line defines the column names for a pandas DataFrame, including 'BattingTeam', 'BowlingTeam', 'City', 'runs\_left', 'balls\_left', 'wickets\_left', 'current\_run\_rate', 'required\_run\_rate', and 'target'.

Then, a pandas DataFrame `team2023` is created using the `l` list and the defined column names.

In the following line, the `predict\_proba` method is called on an object `pipe`, which likely represents a fitted machine learning model or pipeline. This method is typically used to obtain the predicted probabilities or class probabilities for the input data in `team2023`.

The output of `pipe.predict\_proba(team2023)` is an array with a single element, `[[0.46, 0.54]]`, which could represent the predicted probabilities for a binary classification problem or a multi-class classification problem with two classes.



The code snippet appears to be related to a machine learning or data analysis task involving cricket match data.

In the first line, a list `l` is created with several elements, including team names ('Mumbai Indians', 'Gujarat Titans'), the city 'Ahmedabad', and some numerical values (100, 85, 4, 7.4, 10.5, 220).

The next line defines the column names for a pandas DataFrame, including 'BattingTeam', 'BowlingTeam', 'City', 'runs\_left', 'balls\_left', 'wickets\_left', 'current\_run\_rate', 'required\_run\_rate', and 'target'.

Then, a pandas DataFrame `team2023` is created using the `l` list and the defined column names.

In the following line, the `predict\_proba` method is called on an object `pipe`, which likely represents a fitted machine learning model or pipeline. This method is typically used to obtain the predicted probabilities or class probabilities for the input data in `team2023`.

The output of `pipe.predict\_proba(team2023)` is an array with a single element, `[[0.66, 0.34]]`, which could represent the predicted probabilities for a binary classification problem or a multi-class classification problem with two classes.

**References**

* **Dataset** <https://github.com/simranjeet97/IPL2023_WinningPrediction_EDA_Dashboard/tree/main/IPL2023_Data>
* **Project Idea**

<https://www.geeksforgeeks.org/machine-learning-projects/>

* **Google**

[https://www.google.com](https://www.google.com/)

* **Geeks 4 Geeks**

<https://www.geeksforgeeks.org/>