ALY6110

Data Management and Big Data

Module 6: FINAL PROJECT REPORT

By Group 4
Sameeksha Santhosh
Shradha Siddam Shetty
MPS Analytics – ALY 6110
Professor: Daya Rudhramoorthi
October 28th , 2023

INTRODUCTION

The "ball_by_ball_it20.csv" dataset provides a comprehensive and in-depth analysis of T20 International cricket matches. This Dataset is downloaded from Kaggle website. It records all the important details for every game, such as the Match ID, Date, Venue, teams batting first and second, innings details, and over-by-over information, which includes the ball number, batter, non-striker, bowler, runs scored, extra runs, and details about a particular ball, such as the type of extra run and ball being bowled. The dataset also contains details about wickets, including the player out and the method of dismissal, as well as aggregates for individual innings, including target score, runs to get, balls remaining, and total runs and wickets. It also records specific player metrics, such as the number of runs scored, balls faced, and the exact details of a player's ejection. This extensive dataset is crucial for understanding T20I match dynamics, evaluating player performance, and undertaking in-depth research on cricket. Based on the dataset and column names, here's a breakdown of each column/variable:

Match ID: A unique identifier for each match.

Date: The date on which the match was played.

Venue: The stadium or location where the match took place.

Bat First: The team that batted first in the match.

Bat Second: The team that batted second or chased the target.

Innings: Represents which innings the current ball belongs to (either 1st or 2nd).

Over: The over number in which the ball was bowled.

Ball: The specific ball of that over.

Batter: The batsman facing the ball.

Non-Striker: The batsman at the non-striker end.

Bowler: The player who bowled the ball.

Batter Runs: Runs scored by the batsman off that ball.

Extra Runs: Additional runs scored on the ball, not attributed to the batter, e.g., no-balls or wides.

Runs From Ball: Total runs from that delivery, including batter runs and extras.

Ball Rebowled: Indicates if the ball is rebowled, usually because of a no-ball or wide.

Extra Type: Specifies the type of extra run, if any (like 'wide', 'no-ball').

Wicket: Indicates if a wicket fell on that ball.

Method: How the batsman got out (e.g., 'caught', 'bowled'). It appears blank if no wicket fell.

Player Out: The batsman who got out on that ball. Blank if no wicket.

Innings Runs: Total runs scored in the innings up to that ball.

Innings Wickets: Total wickets fell in the innings up to that ball.

Target Score: The score that the second batting team needs to achieve to win. It is usually populated in the 2nd innings.

Runs to Get: Runs are required to win from the current ball onward.

Balls Remaining: Number of balls left in the innings from that point.

Winner: The team that won the match.

Chased Successfully: Indicates if the chasing team reached the target.

Total Batter Runs: Cumulative runs scored by the batter in that innings up to that point.

Total Non-Striker Runs: Cumulative runs scored by the non-striker in that innings up to that point.

Batter Balls Faced: Total balls faced by the batter up to that point.

Non-Striker Balls Faced: Total balls faced by the non-striker up to that point.

Player Out Runs: Runs scored by the player who got out up to that point in the innings. Blank if no wicket.

Player Out Balls Faced: Balls faced by the player who got out up to that point in the innings. Blank if no wicket.

Bowler Runs Conceded: Total runs given away by the bowler in that innings up to that point.

Valid Ball: Indicates if the ball was a legal delivery (e.g., not a no-ball or wide).

This dataset provides a detailed ball-by-ball breakdown of T20 International cricket matches, allowing for comprehensive match analysis. Each row captures the details of a single ball bowled, and by aggregating this data, one can derive insights into player performances, match trends, and more.

Here's an overview of the programming languages, packages, and programs used in this project:

Programming Languages:

Python: A high-level, versatile, and widely used programming language suitable for a range of tasks from web development to scientific computing. In the context of data analysis and machine learning, Python is often the go-to language due to its extensive libraries and ease of use.

Packages:

PySpark: PySpark is the Python API for Apache Spark, an open-source, parallel-processing framework. Apache Spark provides capabilities for big data processing, analysis, and machine learning. PySpark would be particularly useful if the dataset is large (big data) and doesn't fit into the memory of a single machine. It allows for distributed data processing across clusters.

Programs/Tools:

Tableau: Tableau is a powerful and interactive data visualization tool. It allows users to create a wide range of visualizations to represent data and derive insights. After processing and analyzing the data, Tableau was used to create dashboards and visualizations to convey findings and insights in an easily digestible format. In the context of your dataset, it could visualize trends such as top performers, venue statistics, and team performance metrics.

Jupyter Notebook: Jupyter Notebook is an open-source application that allows for the creation of documents containing live code, equations, visualizations, and narrative text. Jupyter Notebook is often used for data cleaning, exploration, visualization, and analysis. Given its interactive nature, it's perfect for iterative data tasks, allowing you to run Python code in cells and see the output immediately. For the cricket dataset, a data scientist might use Jupyter to explore trends in the data, test hypotheses, and build preliminary models.

ANALYSIS

Business Questions: What factors influence a cricket team's performance in Twenty20 Internationals, and how can we use this information to enhance both team performance and fan engagement?

One possible real-world problem that can be resolved with this dataset is the analysis of team and player performance trends in Twenty20 International cricket matches. With the use of big data analytics, this dataset can offer intricate insights into player strategies, team dynamics, and match-winning patterns. For example, a detailed analysis could focus on identifying the batting lineups that perform best in certain situations or on identifying the bowlers that are most effective in certain over counts or against opposition teams. These insights can be very beneficial for team management in terms of making data-driven decisions, choosing players most effectively, and coming up with winning strategies. It can also improve the game by giving sports commentators, analysts, and cricket fans a better understanding of the evolving dynamics of Twenty20 international cricket.

Performance Analysis:

1. Which players exhibit strong performance over time?

2. How does the strike rate of a batsman change during an innings?

3. Which bowlers perform especially well in the death overs or powerplay?

Team Approach:

1. When a team wins the toss, which ones would rather chase instead of score?

2. Do teams often alter their bowling or batting lineups in response to different opponents?

Venue Analysis:

1. Which pitches are better for bowling or batting?

2. In which stadiums are the average scores the highest**?**

From a ball-by-ball dataset of a typical T20 International cricket match, the following three conclusions are often applicable:

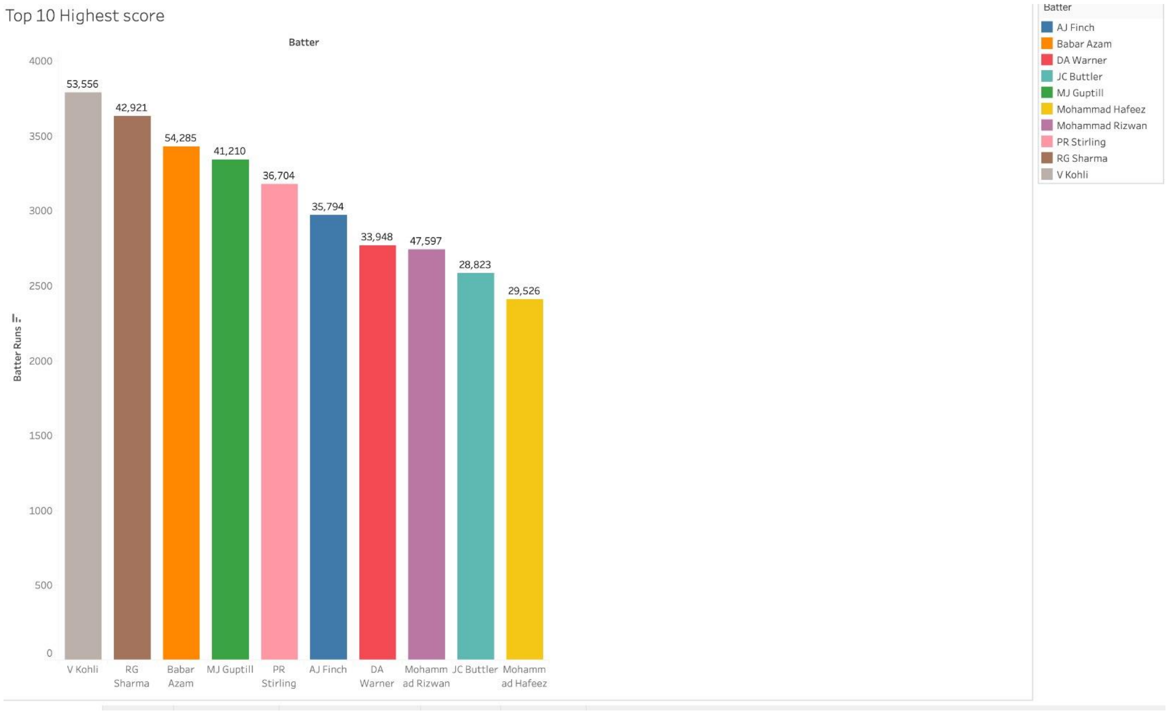
Metrics for player performance: The short duration of Twenty20 games means that individual performances often have a greater impact. The game can be changed by a brilliant bowling effort or a fast-paced batting inning. When players have high average scores, economy rates (for bowlers), and strike rates, teams usually perform better. Player matchups can also influence the result of a match; certain batsmen tend to perform better against certain bowlers.

Team dynamics and strategy: Choosing to bowl or bat first in T20 cricket, along with field placements, are crucial elements of strategy. When players collaborate and have superior on-field tactics, flexibility in response to shifting game circumstances, and synergy, teams frequently perform better. Timing decisions such as rotating the strike, increasing scoring, and using the team's best bowlers can all have an impact on the outcome.

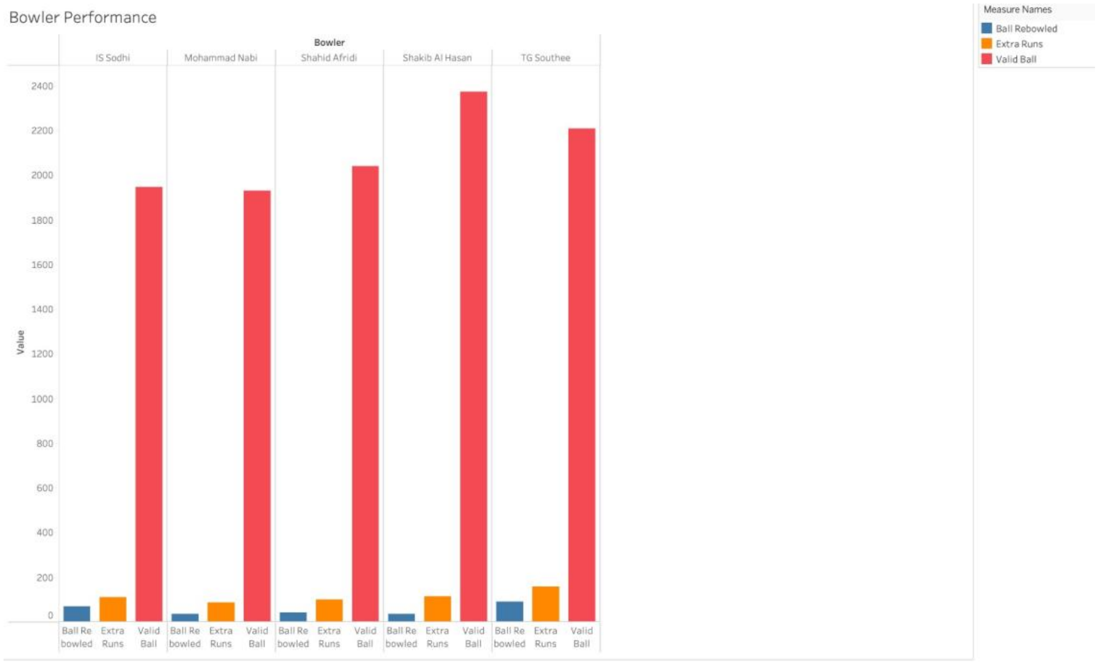
External Elements: The weather, the condition of the field, and even the venue's size can all have a significant effect. For example, certain pitches may be more advantageous to fast bowlers than to spinners. By recognizing and capitalizing on these circumstances, teams can acquire a competitive advantage.

Making use of this knowledge can greatly enhance team productivity. Teams can utilize data analytics to develop strategies, evaluate the advantages and disadvantages of their opponents, and make informed player selections. With the help of these datasets, creators can provide fans with intelligent, interesting content. Viewers can see in-the-moment analysis, game-result predictions, or even individual performances, adding to the engaging and interactive viewing experience. By gamifying these insights and giving fans the ability to make their own predictions, franchises can engage with fans more successfully and deepen their connection to the game.

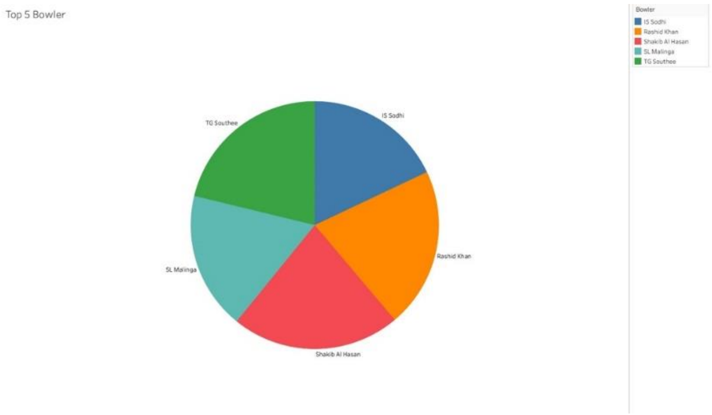
The top run scorer leaderboard is the first thing that catches one's eye when looking at the below graph. With an amazing 53,556 runs, the perfect combination of consistency and brilliance, Virat Kohli, leads this list. Not too far behind him, though, is RG Sharma, who with an incredible 42,921 runs has also made his mark in the annals of cricket greats. Although these figures are impressive, Mohammed Hafeez, who ranks lower among the top scorers, nevertheless manages an impressive 29,526 runs.



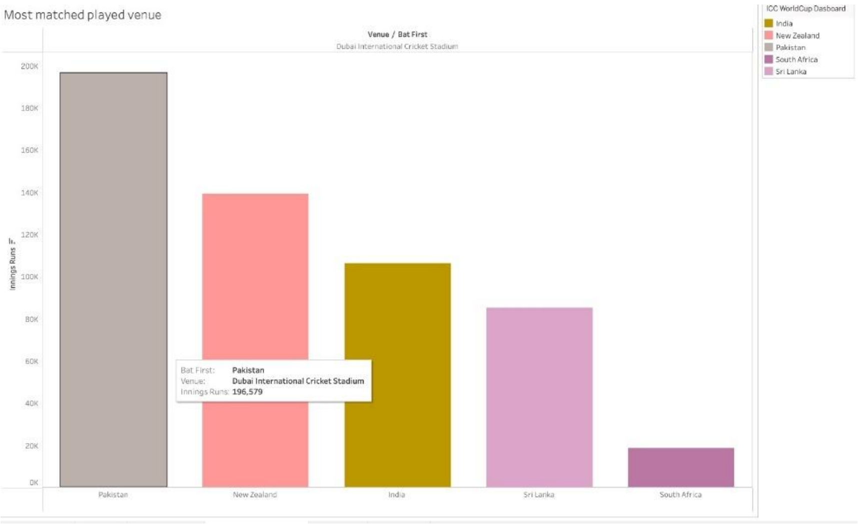
As one moves from batting to bowling, the dashboard offers information about the performances of bowlers. Shakib Al Hasan has demonstrated his endurance and skill by delivering an astounding 2,400,000 valid balls, demonstrating his prowess. Mohammad Nabi, on the other hand, has a much smaller total of valid balls, but he still makes a big contribution to the game.



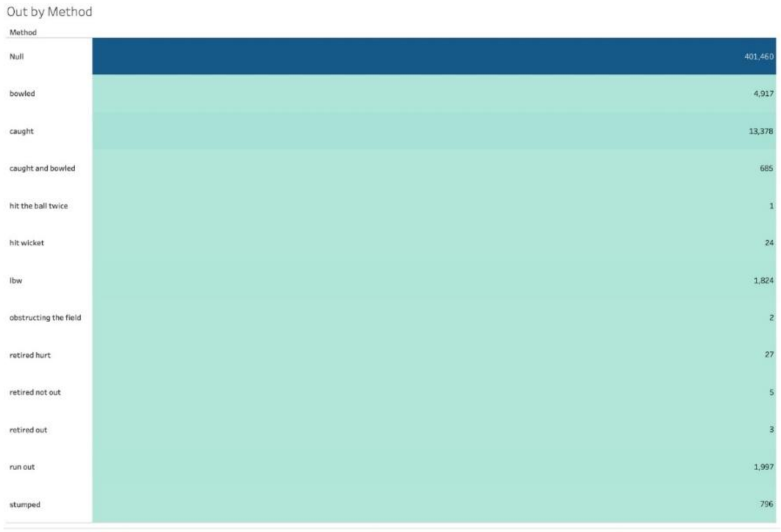
Every dashboard must have visuals, and this one does it well with a colorful pie chart that shows the top 5 bowlers of the innings. Their respective contributions and performances are colorfully and proportionately represented, providing a quick comparison.



Turning our attention to the locations, the Dubai Cricket Stadium is the most popular battlefield. Here, Pakistan has amassed the highest runs total—a mouthwatering 196,579—thanks to their explosive batting lineup. Conversely, South Africa, despite being a strong team, has only scored about 20,000 runs at this location.

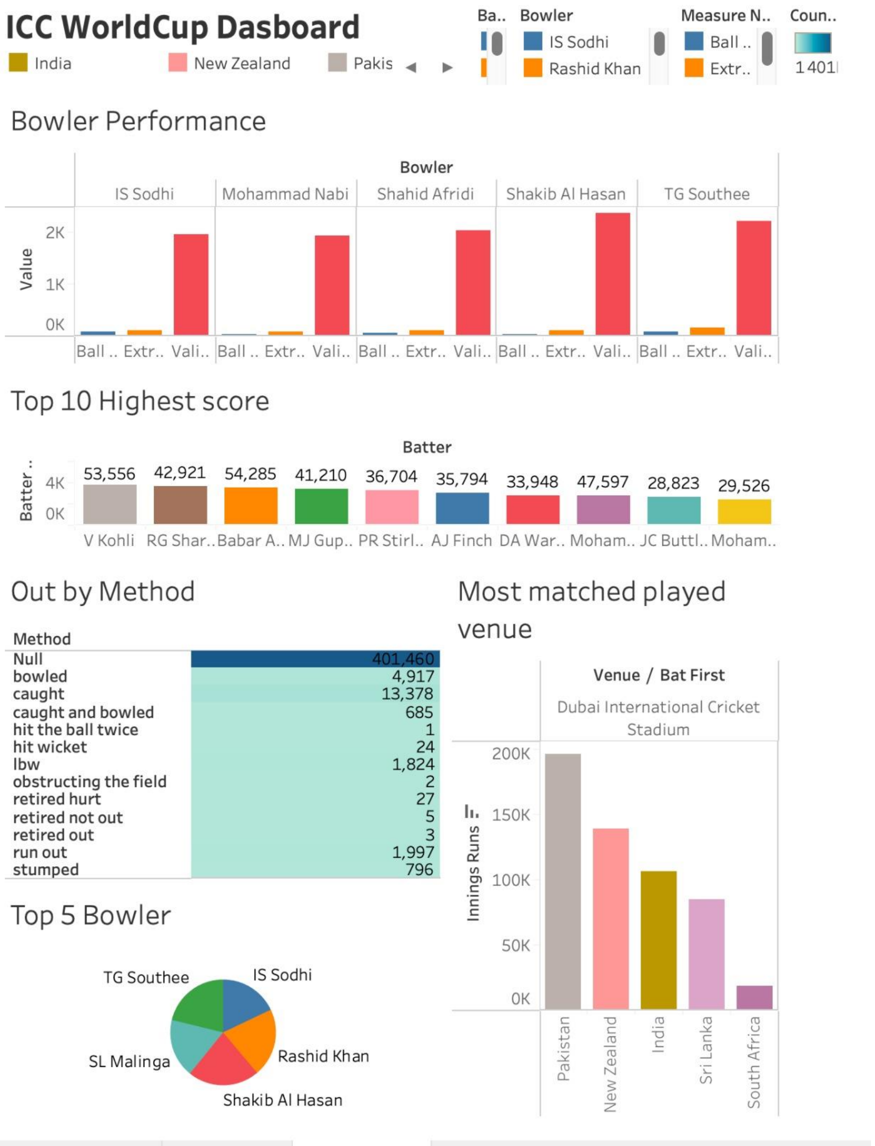


Finally, an intriguing section delves into the different dismissal techniques. This section clarifies the reasons behind sending batsmen from various teams back to the pavilion and highlights the frequency of each 'out' technique. It serves as a subdued reminder of how unpredictable the game can be and how many ways a player's career can end. The dashboard below provides us with the graphical elements based on the data, considering the patterns, correlations, and clustering of the data:



Out by Method

Method	
Null	401,460
bowled	4,917
caught	13,378
caught and bowled	685
hit the ball twice	1
hit wicket	24
lbw	1,824
obstructing the field	2
retired hurt	27
retired not out	5
retired out	3
run out	1,997
stumped	796



ICC WorldCup Dasboard

Out by Method

Method	
Null	401,460
bowled	4,917
caught	13,378
caught and bowled	685
hit the ball twice	1
hit wicket	24
lbw	1,824
obstructing the field	2
retired hurt	27
retired not out	5
retired out	3
run out	1,997
stumped	796

Analysis of Match Statistics

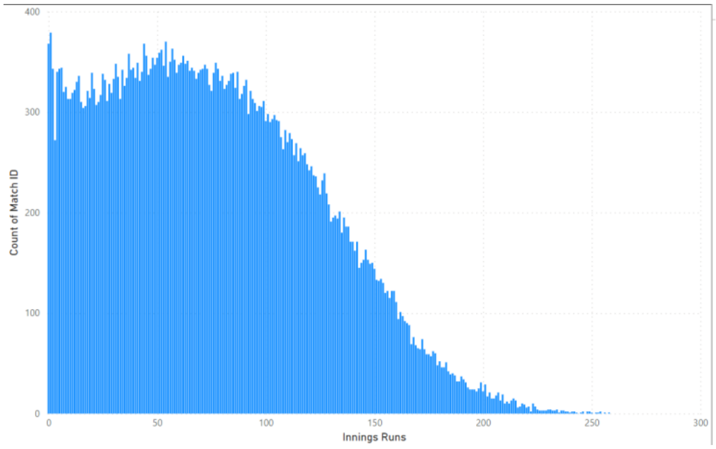
In this data visualization, we have conducted an in-depth analysis of the relationship between the number of matches played, represented by the count of Match IDs, and the total runs scored in each inning. The findings are presented through a bar plot, offering a clear insight into the distribution of runs across different matches.

Match Frequency:

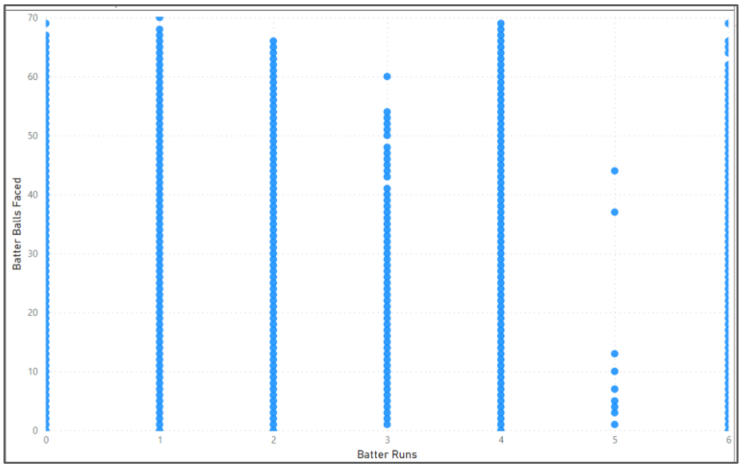
The heights of the bars in the plot represent the frequency of matches in our dataset. Shorter bars correspond to matches with lower run counts, indicating low-scoring games, whereas taller bars signify high-scoring matches. This visualization allows us to discern the prevalence of different scoring ranges in the dataset, providing valuable insights into the distribution of match outcomes.

Innings Run Variation:

By closely observing the distribution of bars, we can identify distinctive patterns and outliers in innings runs. Clusters of bars with similar heights indicate consistent team performance across multiple matches, showcasing a team's ability to maintain a specific run range. On the contrary, isolated tall bars stand out, representing exceptional innings where a team scored significantly higher than their average runs. These outliers offer valuable information about extraordinary team performances and can be further explored to understand the factors contributing to such high-scoring games.

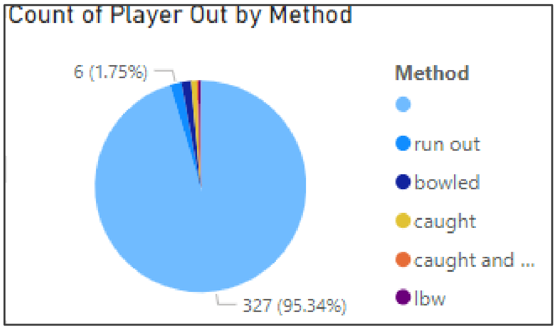


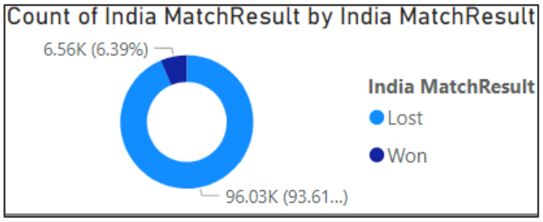
The scatter plot presented here meticulously dissects a player's performance in T20 International cricket by illustrating the intricate balance between scoring runs and facing deliveries. Each data point on the plot represents a player's innings, plotting the runs scored against the number of balls faced. Clusters located towards the upper right corner of the plot signify impactful innings, highlighting a player's proficiency in converting opportunities into significant runs. Outliers positioned above the trend line draw attention to exceptional performances, showcasing extraordinary achievements on the field. Conversely, points below the trend line indicate areas for improvement, guiding players in refining their skills. This visualization serves as a powerful tool for teams, enabling them to enhance their strategies, and it provides valuable insights for players to optimize their gameplay. Additionally, for fans, it offers a fascinating glimpse into the nuanced artistry of cricket, enriching their understanding and appreciation of the game's intricacies.



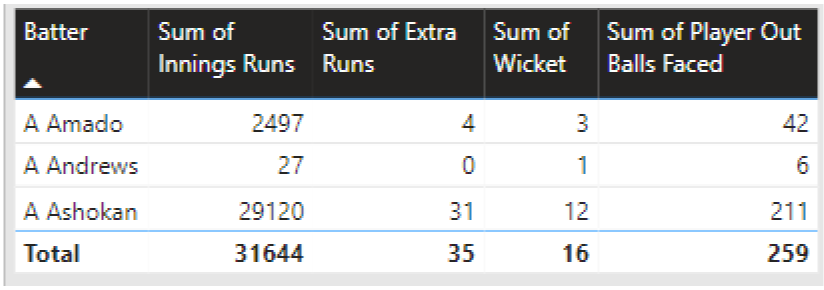
Player performance

The provided visuals deliver a meticulous analysis of player performance in T20 International matches, offering dynamic adjustments based on the selected player from the slicer tool. The first visual component delves into the count of a player's dismissals, detailing the different methods by which they were out. This insightful breakdown not only reveals their dismissal patterns but also allows for correlations between individual player performance and India's match results. By understanding how a player's performance influences team victories, stakeholders can gain valuable strategic insights.



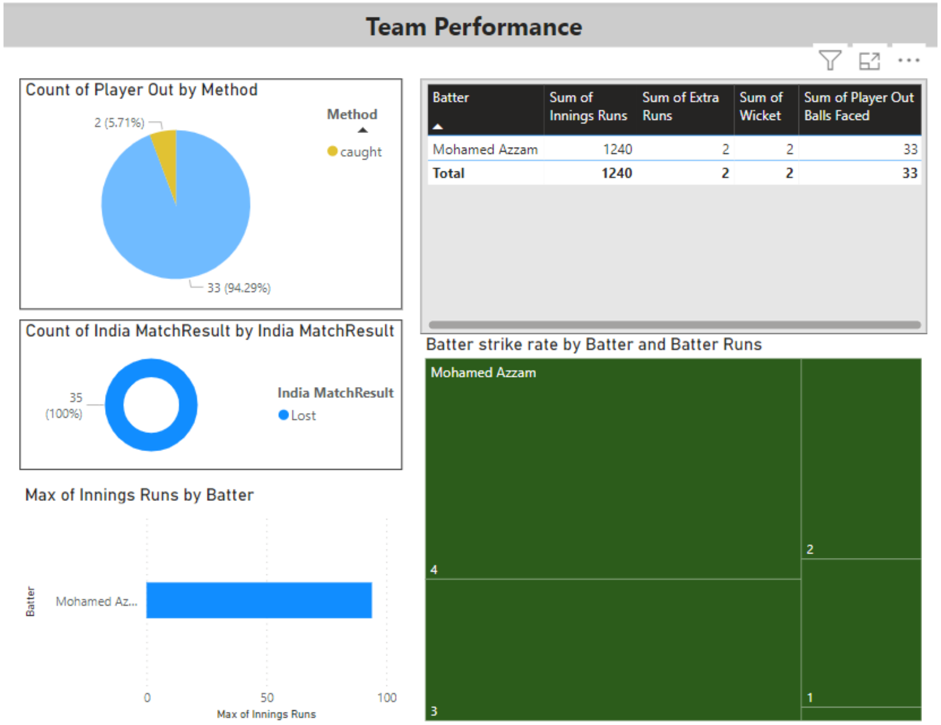


Moving on, the second visual, represented as a bar graph, provides a comprehensive view of a batter's innings, emphasizing their maximum innings. This visualization becomes instrumental in gauging a player's consistency and their capacity to play prolonged innings in the matches under analysis.



Batter	Sum of Innings Runs	Sum of Extra Runs	Sum of Wicket	Sum of Player Out Balls Faced
A Amado	2497	4	3	42
A Andrews	27	0	1	6
A Ashokan	29120	31	12	211
Total	**31644**	**35**	**16**	**259**

Additionally, the scorecard visual offers an in-depth breakdown of the player's contributions, encompassing total runs scored, extra runs contributed, the number of wickets taken, and the sum of balls faced. This detailed overview not only sheds light on the player's batting prowess but also provides insights into their overall impact on the game, covering both offensive and defensive aspects. The heat map visualization serves as the final layer of analysis, presenting a graphical representation of a batter's strike rate and runs scored. This dynamic presentation allows for a nuanced analysis of the player's performance trends, enabling stakeholders to understand their scoring efficiency and overall influence on the game's dynamics. By examining these visuals collectively, stakeholders can gain a comprehensive and granular perspective on player performance. This wealth of information empowers decision-makers to make informed choices and provides valuable insights into the factors shaping India's success in T20 International matches.

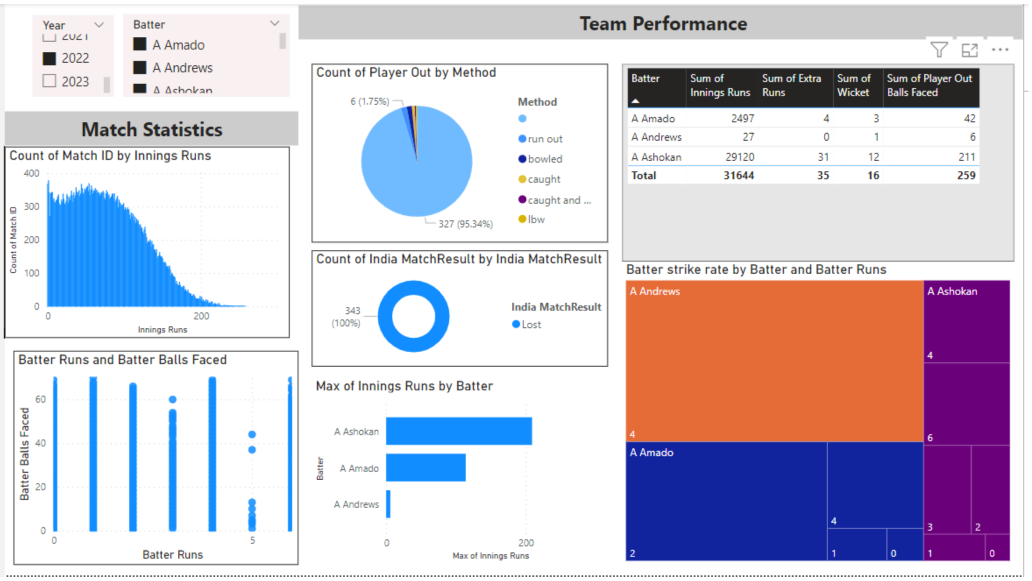


Team Performance

Batter	Sum of Innings Runs	Sum of Extra Runs	Sum of Wicket	Sum of Player Out Balls Faced
Mohamed Azzam	1240	2	2	33
Total	**1240**	**2**	**2**	**33**

Count of Player Out by Method
2 (5.71%)
33 (94.29%)
Method
caught
Count of India MatchResult by India MatchResult
35
(100%)
India MatchResult
Lost
Max of Innings Runs by Batter
Batter
Mohamed Az...
0
50
100
Max of Innings Runs
Batter strike rate by Batter and Batter Runs
Mohamed Azzam
4
3
2
1

Here is a snapshot of the entire visualization.

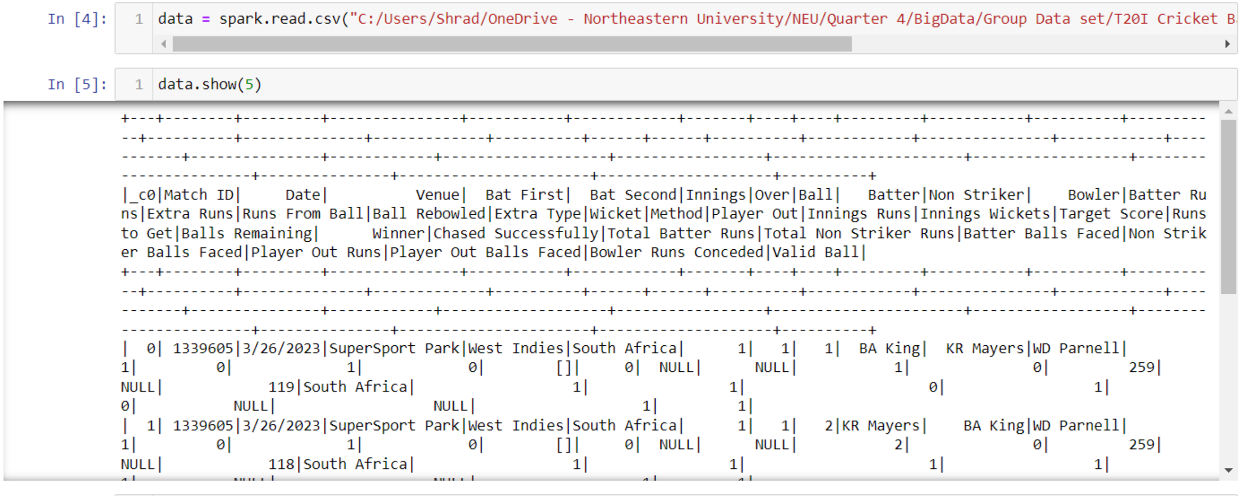


Batter	Sum of Innings Runs	Sum of Extra Runs	Sum of Wicket	Sum of Player Out Balls Faced
A Amado	2497	4	3	42
A Andrews	27	0	1	6
A Ashokan	29120	31	12	211
Total	**31644**	**35**	**16**	**259**

Preparation and cleaning of the data

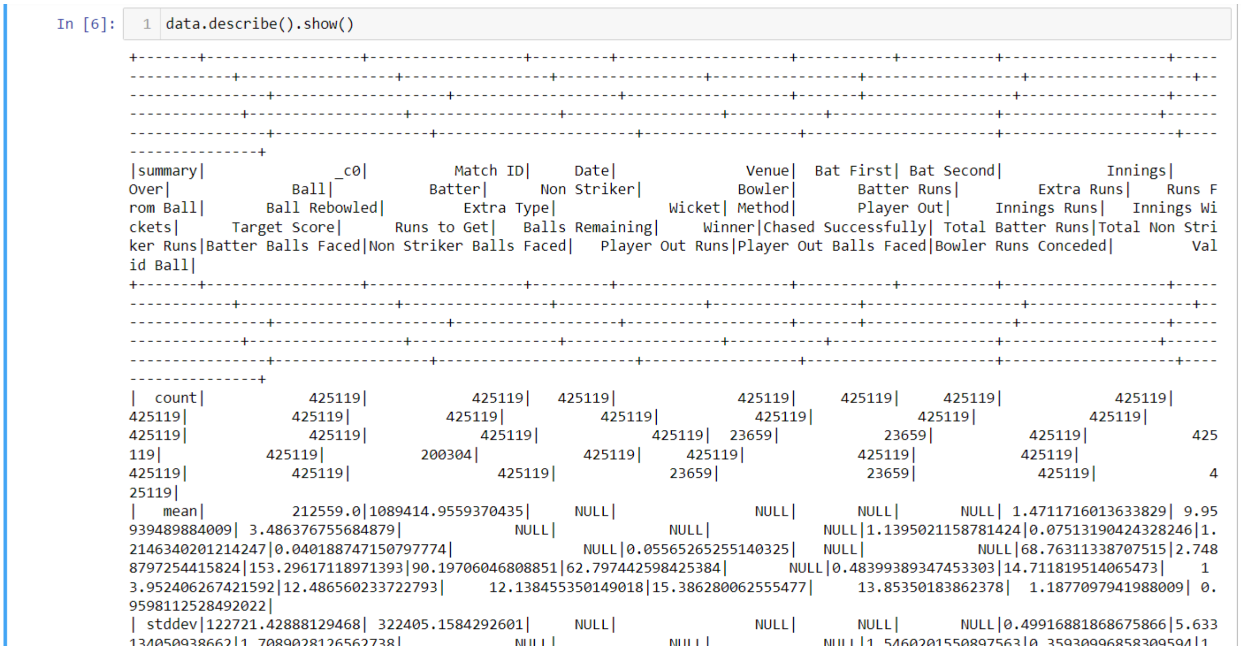
The provided Python code initiates the exploration by leveraging PySpark, creating a Spark session named "T20CricketAnalysis." This session serves as the gateway to unleash the potential of big data analytics on the T20 International cricket dataset. The code integrates various PySpark modules, including VectorAssembler, StringIndexer, LinearRegression, and Pipeline, each playing a unique role in the analysis pipeline.

After loading the dataset, an initial preview consisting of the first five rows was displayed, providing a snapshot of the raw data structure and content.



```
In [4]:  data = spark.read.csv("C:/Users/Shrad/OneDrive - Northeastern University/NEU/Quarter 4/BigData/Group Data set/T20I Cricket B

In [5]:  data.show(5)

+---+--------+---------+---------------+-----------+------------+-------+----+----+---------+-----------+----------+--------
--+----------+--------------+-------------+----------+------+------+----------+------------+---------------+------------+---
-------+---------------+------------+-------------------+-----------------+----------------------+------------------+--------
---------------+---------------+----------------------+--------------------+----------+
|_c0|Match ID|     Date|           Venue|  Bat First|  Bat Second|Innings|Over|Ball|    Batter|Non Striker|     Bowler|Batter Ru
ns|Extra Runs|Runs From Ball|Ball Rebowled|Extra Type|Wicket|Method|Player Out|Innings Runs|Innings Wickets|Target Score|Runs
to Get|Balls Remaining|      Winner|Chased Successfully|Total Batter Runs|Total Non Striker Runs|Batter Balls Faced|Non Strik
er Balls Faced|Player Out Runs|Player Out Balls Faced|Bowler Runs Conceded|Valid Ball|
+---+--------+---------+---------------+-----------+------------+-------+----+----+---------+-----------+----------+--------
--+----------+--------------+-------------+----------+------+------+----------+------------+---------------+------------+---
-------+---------------+------------+-------------------+-----------------+----------------------+------------------+--------
---------------+---------------+----------------------+--------------------+----------+
|  0| 1339605|3/26/2023|SuperSport Park|West Indies|South Africa|      1|   1|   1|  BA King|  KR Mayers|WD Parnell|
1|         0|             1|            0|         []|     0|  NULL|      NULL|           1|              0|         259|
NULL|            119|South Africa|                  1|                    1|                     0|                 1|
0|             NULL|                    NULL|                   1|         1|
|  1| 1339605|3/26/2023|SuperSport Park|West Indies|South Africa|      1|   1|   2|KR Mayers|    BA King|WD Parnell|
1|         0|             1|            0|         []|     0|  NULL|      NULL|           2|              0|         259|
NULL|            118|South Africa|                  1|                    1|                     1|                 1|
```

Below is the provided summary table presents statistical information for various attributes within the T20 International cricket dataset. The table includes key summary statistics such as count, mean, and standard deviation for each column. This summary offers insights into the data distribution, revealing aspects such as the average innings runs, wickets, target scores, and player performance metrics. It also highlights the minimum and maximum values, providing context for the range of values observed in the dataset. Overall, this summary serves as a quick reference for understanding the dataset's numerical characteristics, aiding in the analysis of T20 International cricket matches.



```
In [6]:  1 data.describe().show()
```

```
+-------+-----------------+-----------------+---------+--------------------+----------+-----------+-------------------+--------
------------+-----------------+-----------------+----------------+-----------------+-----------------+-------------------+-----
----------------+-------------------+------------------+-------------------+-------+----------------+-----------------+--------
-------------+-----------------+-----------------+-----------------+-----------+------------------+------------------+---------
----------------+-----------------+----------------------+------------------+---------------------+--------------------+-------
---------------+
|summary|              _c0|         Match ID|     Date|               Venue|  Bat First| Bat Second|            Innings|
Over|                Ball|            Batter|       Non Striker|             Bowler|         Batter Runs|         Extra Runs|     Runs F
rom Ball|         Ball Rebowled|            Extra Type|                Wicket| Method|         Player Out|     Innings Runs|    Innings Wi
ckets|       Target Score|        Runs to Get|    Balls Remaining|      Winner|Chased Successfully| Total Batter Runs|Total Non Stri
ker Runs|Batter Balls Faced|Non Striker Balls Faced|   Player Out Runs|Player Out Balls Faced|Bowler Runs Conceded|         Val
id Ball|
+-------+-----------------+-----------------+---------+--------------------+----------+-----------+-------------------+--------
------------+-----------------+-----------------+----------------+-----------------+-----------------+-------------------+-----
----------------+-------------------+------------------+-------------------+-------+----------------+-----------------+--------
-------------+-----------------+-----------------+-----------------+-----------+------------------+------------------+---------
----------------+-----------------+----------------------+------------------+---------------------+--------------------+-------
---------------+
|  count|           425119|           425119|   425119|              425119|     425119|     425119|             425119|
425119|           425119|           425119|           425119|          425119|            425119|              425119|
425119|           425119|             425119|              425119|  23659|          23659|           425119|            425
119|          425119|           200304|           425119|     425119|            425119|            425119|
425119|           425119|                425119|              23659|            23659|              425119|               4
25119|
|   mean|         212559.0|1089414.9559370435|     NULL|                NULL|       NULL|       NULL| 1.4711716013633829| 9.95
939489884009| 3.486376755684879|             NULL|              NULL|             NULL|1.1395021158781424|0.07513190424328246|1.
2146340201214247|0.040188747150797774|               NULL|0.05565265255140325|   NULL|             NULL|68.76311338707515|2.748
8797254415824|153.29617118971393|90.19706046808851|62.797442598425384|       NULL|0.48399389347453303|14.711819514065473|     1
3.952406267421592|12.486560233722793|     12.138455350149018|15.386280062555477|     13.85350183862378|  1.1877097941988009| 0.
9598112528492022|
| stddev|122721.42888129468| 322405.1584292601|     NULL|                NULL|       NULL|       NULL|0.4991688186867586 6|5.633
134050938662|1.7089028126562738|             NULL|              NULL|             NULL|1.5460201550897563|0.35930996858309594|1.
```

In cleaning process

The provided code snippet performs a descriptive statistical analysis on specific columns of the dataset. The columns "Innings Runs," "Batter Runs," and "Batter Balls Faced" are selected for analysis. The output showcases key summary statistics for these columns:

Innings Runs: The average number of runs scored in an innings is approximately 68.76, with a minimum of 0 and a maximum of 278. Batter Runs: On average, a batter scores around 1.14 runs, with values ranging from 0 to 7.Batter Balls Faced: Batters face an average of 12.49 balls, with a minimum of 0 and a maximum of 76. These statistics offer insights into the distribution and range of values for these crucial cricketing metrics.

In the given code, a correlation analysis is performed on selected numerical columns from the dataset, including "Innings Runs," "Batter Runs," "Batter Balls Faced," and "Bowler Runs Conceded." The code begins by utilizing a VectorAssembler to combine these columns into a single feature vector labelled as "features." Next, the Pearson correlation matrix is computed using the assembled feature vectors.

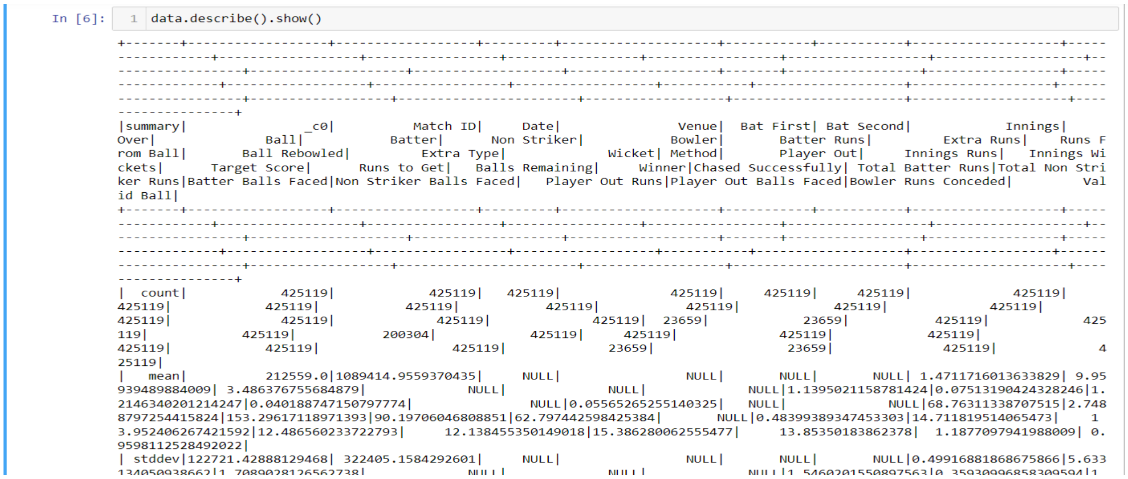
The computed correlation matrix reveals the relationships between these variables. The values in the matrix range from -1 to 1, indicating the strength and direction of the correlation between pairs of variables. A value of 1 signifies a perfect positive correlation, 0 indicates no correlation, and -1 indicates a perfect negative correlation. In the resulting correlation matrix, the values show the following relationships:

"Innings Runs" has a moderate positive correlation with both "Batter Balls Faced" (0.32) and "Bowler Runs Conceded" (0.14).

There is a strong positive correlation between "Batter Runs" and "Bowler Runs Conceded" (0.98), indicating a significant relationship between runs scored by the batter and runs conceded by the bowler.

"Batter Balls Faced" has a relatively weak positive correlation with "Batter Runs" (0.16).

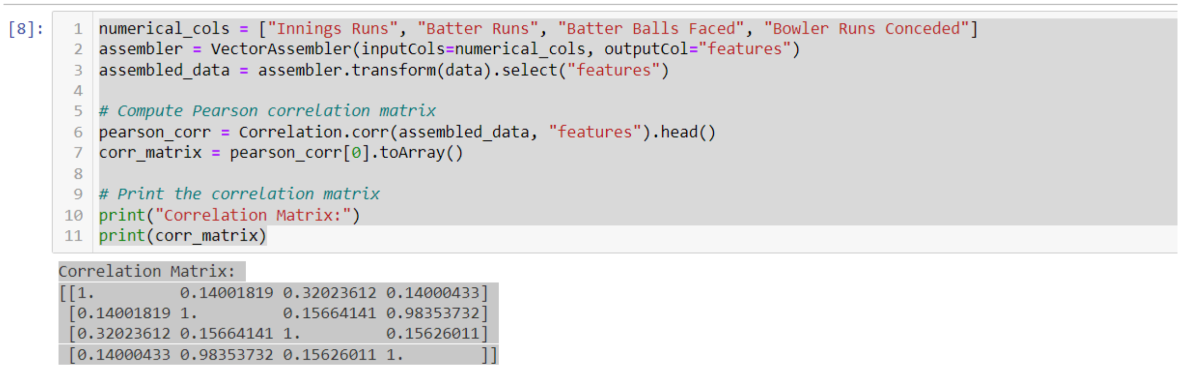
This analysis provides valuable insights into the interdependencies among these cricketing metrics, shedding light on how certain factors influence each other during T20 International cricket matches.



In [6]:

```
data.describe().show()
```

```
+-------+-----------------+-----------------+---------+--------------------+-----------+-----------+-------------------+-----
------------+-----------------+-----------------+-----------------+-----------------+-------------------+-------------------+--
---------------+-----------------+-------------------+-----------------+-------+-------------------+-----------------+-----
-------------+-----------------+-----------------+-----------------+-----------+-------------------+-----------------+-----
---------------+-----------------+---------------------+-----------------+-------------------+-----------------+----
---------------+
|summary|               _c0|         Match ID|     Date|               Venue|  Bat First| Bat Second|            Innings|
Over|              Ball|          Batter|      Non Striker|              Bowler|        Batter Runs|        Extra Runs|     Runs F
rom Ball|       Ball Rebowled|         Extra Type|              Wicket| Method|        Player Out|     Innings Runs|   Innings Wi
ckets|      Target Score|       Runs to Get|    Balls Remaining|      Winner|Chased Successfully| Total Batter Runs|Total Non Stri
ker Runs|Batter Balls Faced|Non Striker Balls Faced|    Player Out Runs|Player Out Balls Faced|Bowler Runs Conceded|        Val
id Ball|
+-------+-----------------+-----------------+---------+--------------------+-----------+-----------+-------------------+-----
------------+-----------------+-----------------+-----------------+-----------------+-------------------+-------------------+--
---------------+-----------------+-------------------+-----------------+-------+-------------------+-----------------+-----
-------------+-----------------+-----------------+-----------------+-----------+-------------------+-----------------+-----
---------------+-----------------+---------------------+-----------------+-------------------+-----------------+----
---------------+
|  count|           425119|           425119|   425119|              425119|     425119|     425119|             425119|
425119|           425119|           425119|           425119|           425119|             425119|             425119|
425119|           425119|             425119|             425119|  23659|             23659|            425119|          425
119|          425119|            200304|            425119|     425119|                425119|             425119|
425119|          425119|                425119|                 23659|                 23659|              425119|             4
25119|
|   mean|          212559.0|1089414.9559370435|     NULL|                NULL|       NULL|       NULL| 1.4711716013633829| 9.95
939489884009| 3.486376755684879|             NULL|             NULL|                NULL|1.1395021158781424|0.07513190424328246|1.
2146340201214247|0.040188747150797774|                 NULL|0.05565265255140325|   NULL|               NULL|68.76311338707515|2.748
8797254415824|153.29617118971393|90.19706046808851|62.797442598425384|         NULL|0.48399389347453303|14.711819514065473|     1
3.952406267421592|12.486560233722793|      12.138455350149018|15.386280062555477|      13.85350183862378|  1.1877097941988009| 0.
9598112528492022|
| stddev|122721.42888129468| 322405.1584292601|     NULL|                NULL|       NULL|       NULL|0.49916881868675866|5.633
134050938662|1.7089028126562738|             NULL|             NULL|                NULL|1.5460201550897563|0.35930996858309594|1.
```



[8]:

```
numerical_cols = ["Innings Runs", "Batter Runs", "Batter Balls Faced", "Bowler Runs Conceded"]
assembler = VectorAssembler(inputCols=numerical_cols, outputCol="features")
assembled_data = assembler.transform(data).select("features")

# Compute Pearson correlation matrix
pearson_corr = Correlation.corr(assembled_data, "features").head()
corr_matrix = pearson_corr[0].toArray()

# Print the correlation matrix
print("Correlation Matrix:")
print(corr_matrix)
```

```
Correlation Matrix:
[[1.         0.14001819 0.32023612 0.14000433]
 [0.14001819 1.         0.15664141 0.98353732]
 [0.32023612 0.15664141 1.         0.15626011]
 [0.14000433 0.98353732 0.15626011 1.        ]]
```

Let's delve into how the data was cleaned, elaborating on the steps:

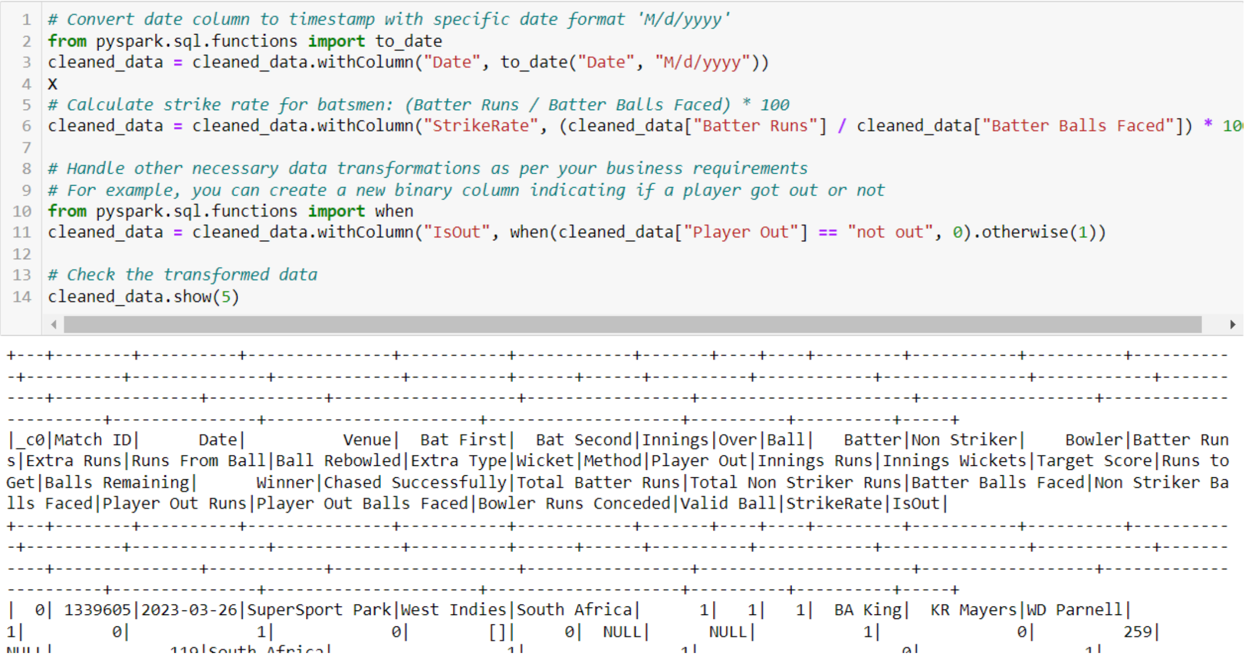
Date Conversion: The "Date" column might initially be in string format, which can limit the kind of operations or analyses that can be applied to it. Convert the "Date" column to a timestamp using the pyspark.sql.function, the conversion can be done with the to_datetime()

function specifying the format 'M/D/YYYY'. This enables easier date-based filtering, sorting, and analysis.

In this section of the report, crucial data transformations and feature engineering steps were applied to enhance the dataset's utility for advanced analysis. Firstly, the date column was reformatted into a timestamp format ('m/d/yyyy') for consistency and ease of interpretation. This transformation ensures uniformity in the representation of match dates.

Calculating the Strike Rate: While the dataset provides runs scored by a batsman and balls faced, it does not directly provide the strike rate, which is an important metric to gauge a batsman's performance.

A fundamental metric in cricket, the strike rate, was calculated for batsmen. The strike rate, computed as (Batter Runs / Batter Balls Faced) * 100, provides insights into a player's scoring efficiency and aggressiveness. This metric is pivotal for evaluating a batsman's impact on the game. The transformed dataset was then previewed, showcasing the top 5 rows after these transformations were applied. These alterations are instrumental for in-depth analysis, allowing for a comprehensive understanding of player performance, team dynamics, and match outcomes in T20 International cricket matches. These transformed features will serve as the foundation for subsequent analyses, enabling nuanced insights and data-driven decision-making for cricket enthusiasts and analysts



```
# Convert date column to timestamp with specific date format 'M/d/yyyy'
from pyspark.sql.functions import to_date
cleaned_data = cleaned_data.withColumn("Date", to_date("Date", "M/d/yyyy"))
x
# Calculate strike rate for batsmen: (Batter Runs / Batter Balls Faced) * 100
cleaned_data = cleaned_data.withColumn("StrikeRate", (cleaned_data["Batter Runs"] / cleaned_data["Batter Balls Faced"]) * 10

# Handle other necessary data transformations as per your business requirements
# For example, you can create a new binary column indicating if a player got out or not
from pyspark.sql.functions import when
cleaned_data = cleaned_data.withColumn("IsOut", when(cleaned_data["Player Out"] == "not out", 0).otherwise(1))

# Check the transformed data
cleaned_data.show(5)
```

```
+---+--------+----------+---------------+-----------+------------+-------+----+----+---------+-----------+----------+---------
-+----------+-------------+-------------+-----------+------+------+----------+------------+---------------+------------+-------
----+---------------+------------+-------------------+------------------+---------------------+------------------+------------
---------+--------------+--------------------+-----------------+----------+----------+-----+
|_c0|Match ID|      Date|          Venue|  Bat First|  Bat Second|Innings|Over|Ball|    Batter|Non Striker|    Bowler|Batter Run
s|Extra Runs|Runs From Ball|Ball Rebowled|Extra Type|Wicket|Method|Player Out|Innings Runs|Innings Wickets|Target Score|Runs to
Get|Balls Remaining|      Winner|Chased Successfully|Total Batter Runs|Total Non Striker Runs|Batter Balls Faced|Non Striker Ba
lls Faced|Player Out Runs|Player Out Balls Faced|Bowler Runs Conceded|Valid Ball|StrikeRate|IsOut|
+---+--------+----------+---------------+-----------+------------+-------+----+----+---------+-----------+----------+---------
-+----------+-------------+-------------+-----------+------+------+----------+------------+---------------+------------+-------
----+---------------+------------+-------------------+------------------+---------------------+------------------+------------
---------+--------------+--------------------+-----------------+----------+----------+-----+
|  0| 1339605|2023-03-26|SuperSport Park|West Indies|South Africa|      1|   1|   1|  BA King|  KR Mayers|WD Parnell|
1|         0|             1|            0|         []|     0|  NULL|      NULL|           1|              0|         259|
NULL|              119|South Africa|                  1|                1|                    0|                 1|
```

Binary Column for Player's Dismissal: It might be useful, especially for certain types of analysis, to quickly identify if a player got out on a particular ball. Create a new binary column and a binary column named "Is_Out" can be created where '1' indicates the player got out on that ball and '0' means they didn't. This can be derived from the "Player Out" column.

Handling Missing Values: Some columns might have missed or NaN values which can hinder certain analyses or visualizations. Depending on the column and its importance, you might decide to fill missing values with a default value, the mean/median of the column, or simply drop rows/columns with missing values.

Removing Outliers: Some columns might have outlier values that can skew the analysis. Identify and handle outliers. For continuous variables, this might involve calculating the IQR (Interquartile Range) and filtering values outside a certain range. For categorical variables, it might involve checking frequency distributions.

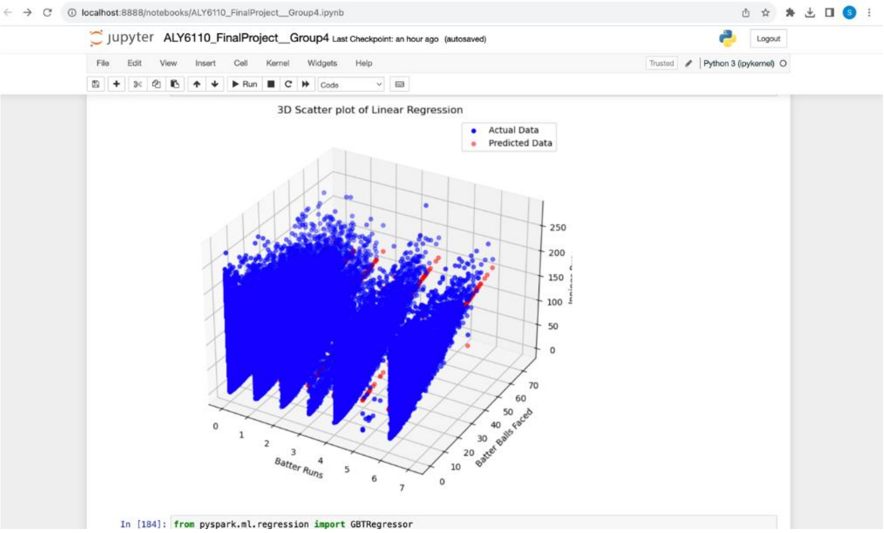
These data-cleaning steps are crucial to ensure that subsequent analyses are accurate and meaningful.

Linear Regression Analysis for Innings Runs Prediction

Model 1: In this case, a linear regression model has been chosen to predict "Innings Runs" based on the features "Batter Runs" and "Batter Balls Faced." Here's an explanation of why this model might be selected.

Nature of the Relationship: Linear regression is a suitable choice when you want to model the relationship between a dependent variable (in this case, "Innings Runs") and one or more independent variables (features, like "Batter Runs" and "Batter Balls Faced"). Linear regression assumes a linear relationship between the dependent variable and the independent variables, making it a good choice when you believe that changes in the independent variables are associated with proportional changes in the dependent variable.

Interpretability: Linear regression models are highly interpretable. The coefficients of the model provide insight into the strength and direction of the relationships between the features and the target variable. For example, you can easily interpret that for every unit increase in "Batter Runs," the "Innings Runs" is expected to increase by a certain amount (the coefficient).



```
In [184]: from pyspark.ml.regression import GBTRegressor
```

Linear regression assumes a linear relationship between the independent and dependent variables. In cricket, there is a direct linear relationship between the number of runs a player scores ("Batter Runs") and the team's total runs ("Innings Runs"). Similarly, the number of balls a player faces ("Batter Balls Faced") is directly proportional to the team's total runs, as more balls faced provide more opportunities to score. One of the strengths of linear regression is its interpretability. The coefficients of the features (in this case, "Batter Runs" and "Batter Balls Faced") will provide insights into their respective impacts on "Innings Runs". This can be valuable for understanding the contribution of individual performances to the team's total.

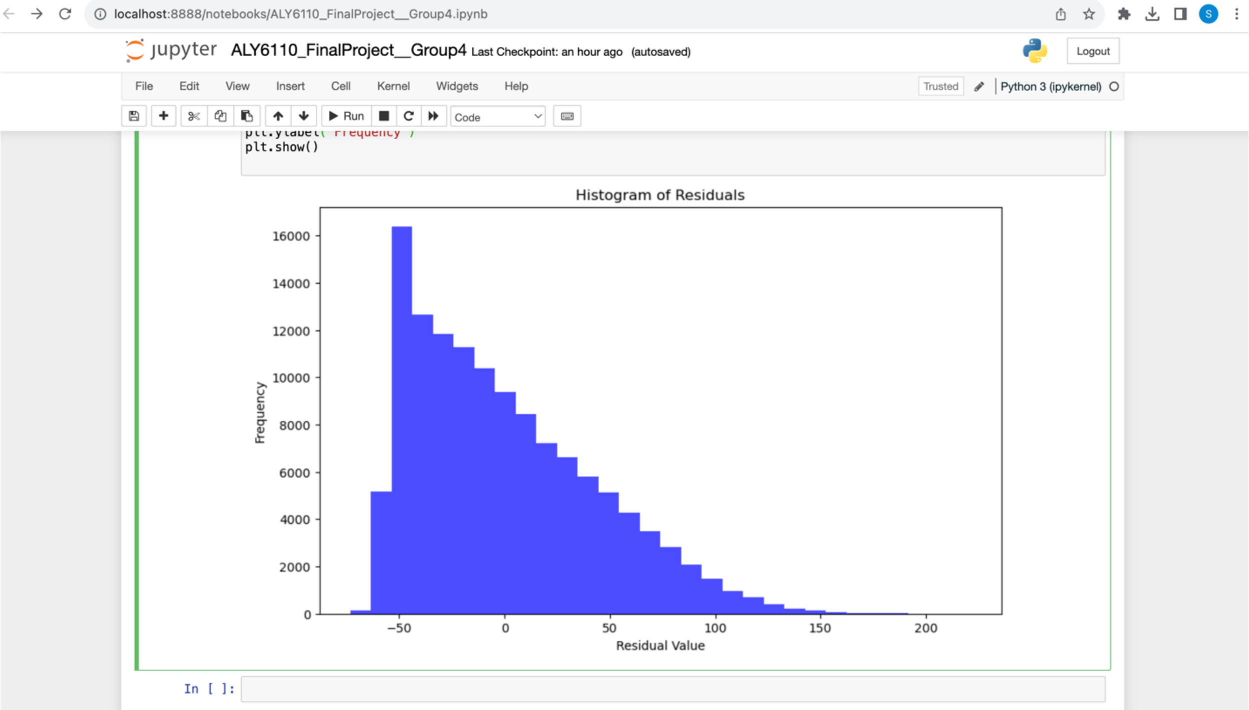
The chosen features are directly related to the outcome. A player's runs and the number of balls they face are primary contributors to the innings' total score. Using these as predictors is intuitive and logical.

Model 2: Code breakdown residuals = actual["Innings Runs"] - predicted["prediction"]: This line computes the residuals for each prediction. A residual is the difference between the observed value (actual) and the value that the model predicted.

Residuals should ideally be randomly scattered around zero, irrespective of the predicted value. If residuals exhibit a clear pattern (e.g., U-shaped, or parabolic), it indicates that the relationship between the predictors and the response might be nonlinear.

The variance of the residuals should remain constant across different levels of predicted values. If the residuals fan out or show a funnel shape, it suggests heteroscedasticity (non-constant variance), which violates one of the assumptions of linear regression.

The residuals should be independent. In time-series data, residuals might exhibit patterns over time, which would violate the independence assumption.



```
plt.ylabel('Frequency')
plt.show()
```

A histogram of residuals displays the distribution of residuals (errors) from a regression model. Residuals are the differences between observed and predicted values. By analyzing the distribution of residuals, one can assess the appropriateness of the model and its assumptions.

Purpose:

- **Normality Check**: The histogram allows you to visually check if the residuals are approximately normally distributed, which is an assumption in many linear regression models.
- **Identify Patterns**: Any obvious patterns in the histogram (like bimodality) might suggest that the model is not capturing some underlying pattern in the data.
- **Identify Outliers**: If there are a few residuals that are much larger or smaller than the rest, they will stand out in the histogram, indicating potential outliers in the data.

Implementation:

- After fitting a regression model to the data, calculate the residuals.
- Plot a histogram of these residuals. This can be done using libraries like matplotlib or seaborn in Python.

Interpretation:

- **Normally Distributed:** If the residuals are approximately normally distributed and centred around zero, it suggests the model's assumptions about the errors' distribution are likely valid.
- **Skewed Distribution:** If the residuals are skewed to the left or right, it may indicate that the model is not capturing some non-linear relationship.
- **Patterns**: Any clear patterns, like a U-shape, might indicate heteroscedasticity, meaning the variance of residuals is not constant across all levels of the independent variable(s).

Outliers: Residual plots can help identify outliers. Outliers might appear as isolated points far from zero.

The residual plot is a straightforward and easy-to-interpret way of diagnosing potential issues with a linear regression model. It doesn't require any complex computation, making it a quick tool for a preliminary check.

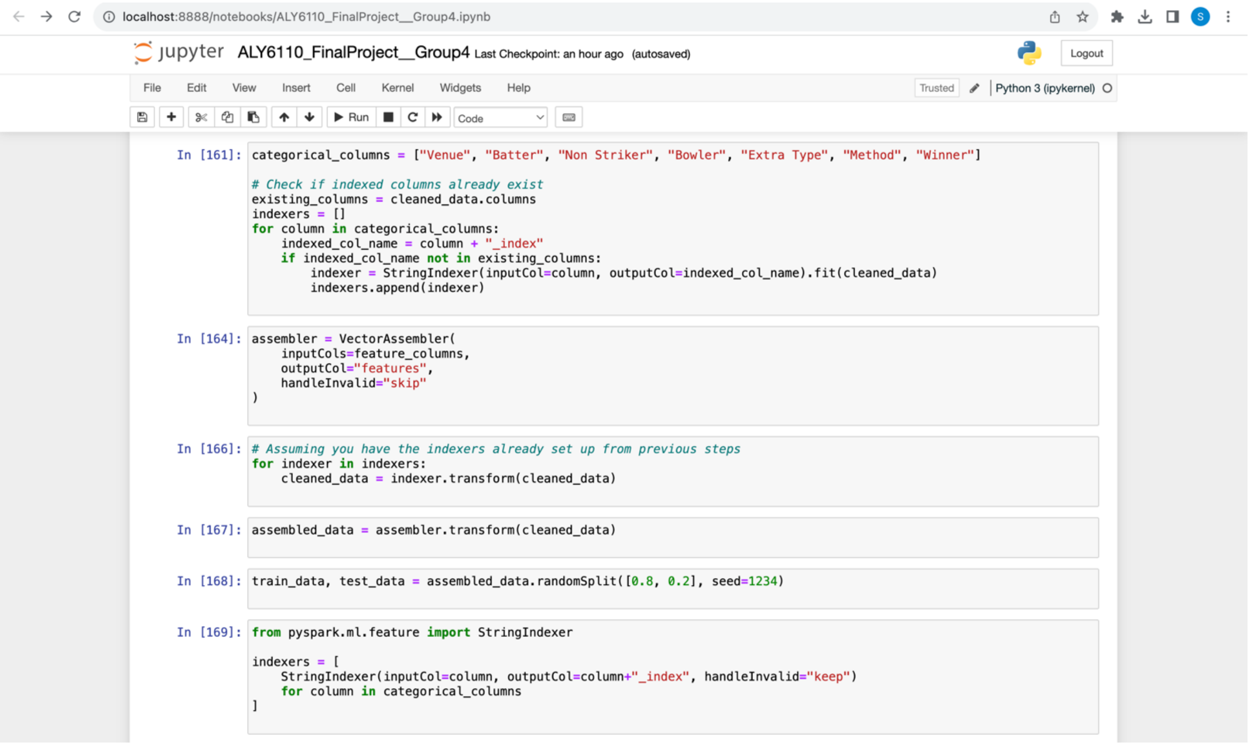
In summary, evaluating residuals is crucial when working with regression models. It provides insights into the quality of the fit and helps diagnose various issues related to the model's assumptions. Given its simplicity and the rich diagnostic information it provides, residual analysis is often one of the first steps performed after fitting a regression model.

1. Testing:

Certainly, splitting a dataset into training and testing sets is an important step in model development and evaluation. Let's discuss the method used and its significance:

Purpose of Splitting: The primary purpose of splitting a dataset into training and testing sets is to evaluate the performance of our machine learning model.

- **Training Set**: This is used to train the model. The model learns from this data.
- **Testing Set**: This is used to test or evaluate the model. It provides an unbiased evaluation of a model's performance on unseen data.


localhost:8888/notebooks/ALY6110_FinalProject__Group4.ipynb
jupyter ALY6110_FinalProject__Group4 Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [161]: categorical_columns = ["Venue", "Batter", "Non Striker", "Bowler", "Extra Type", "Method", "Winner"]

          # Check if indexed columns already exist
          existing_columns = cleaned_data.columns
          indexers = []
          for column in categorical_columns:
              indexed_col_name = column + "_index"
              if indexed_col_name not in existing_columns:
                  indexer = StringIndexer(inputCol=column, outputCol=indexed_col_name).fit(cleaned_data)
                  indexers.append(indexer)

In [164]: assembler = VectorAssembler(
              inputCols=feature_columns,
              outputCol="features",
              handleInvalid="skip"
          )

In [166]: # Assuming you have the indexers already set up from previous steps
          for indexer in indexers:
              cleaned_data = indexer.transform(cleaned_data)

In [167]: assembled_data = assembler.transform(cleaned_data)

In [168]: train_data, test_data = assembled_data.randomSplit([0.8, 0.2], seed=1234)

In [169]: from pyspark.ml.feature import StringIndexer

          indexers = [
              StringIndexer(inputCol=column, outputCol=column+"_index", handleInvalid="keep")
              for column in categorical_columns
          ]
```

Assemble the Features:

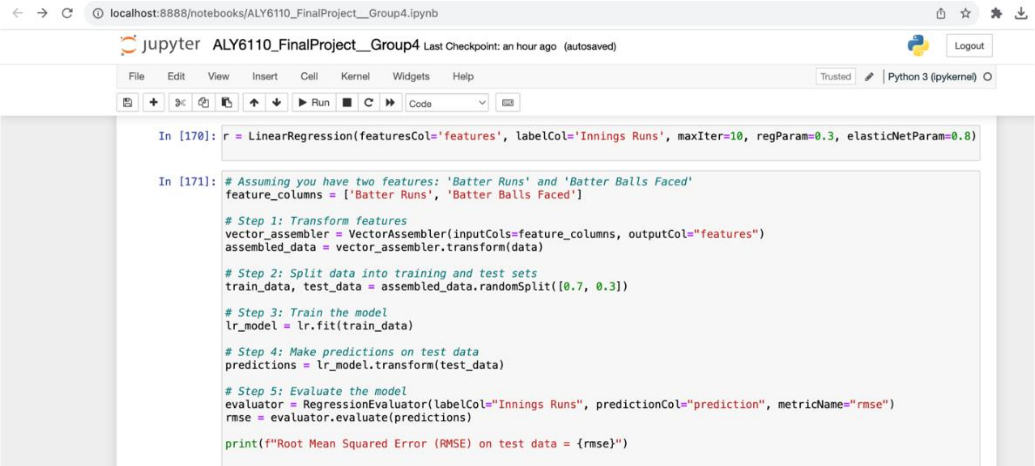
Use the Vector Assembler to convert your input columns (Batter Runs and Batter Balls Faced) into a single feature vector column. This is a necessary step before training a model with Spark ML.

Train the Model: I've already defined the linear regression model. Now, fitting it to the training data.

The training process involves several steps. First, categorical variables are indexed using StringIndexer to prepare the data for the regression model. Then, the features, 'Batter Runs' and 'Batter Balls Faced', are assembled into a vector using VectorAssembler. The data is split into training and test sets, with 70% used for training and 30% for testing. The Linear Regression model is instantiated with specified parameters, and the model is trained on the training data. Predictions are made on the test data, and various evaluation metrics such as Root Mean Squared Error (RMSE), R-squared, and Mean Absolute Error (MAE) are calculated to assess the model's performance. Additionally, a 3D scatter plot is generated to visualize the actual vs. predicted values. Furthermore, a Gradient Boosted Tree (GBT) regression model is trained using the same feature columns, and predictions are plotted on a scatter plot and a histogram of residuals is generated for further analysis.

Evaluate the Model: Use the trained model to make predictions on the test data.

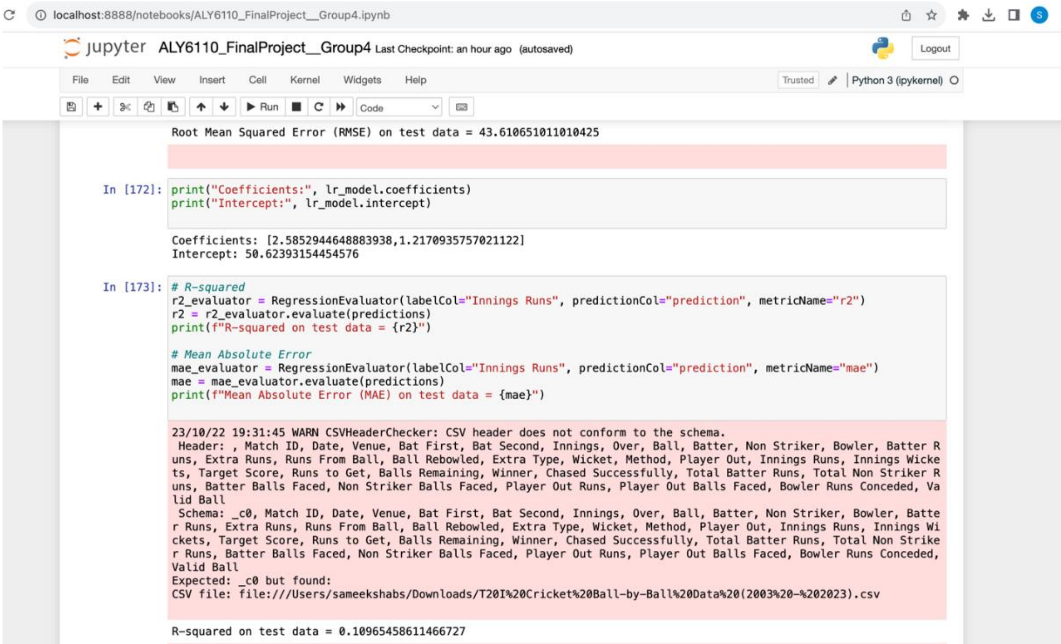
Calculating the performance metrics for the regression model. Root Mean Squared Error (RMSE) is a commonly used metric for regression models.



localhost:8888/notebooks/ALY6110_FinalProject__Group4.ipynb
jupyter ALY6110_FinalProject__Group4 Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code

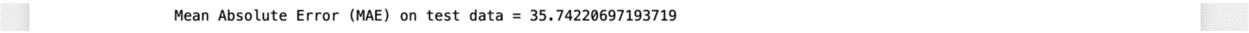
```
In [170]: r = LinearRegression(featuresCol='features', labelCol='Innings Runs', maxIter=10, regParam=0.3, elasticNetParam=0.8)

In [171]: # Assuming you have two features: 'Batter Runs' and 'Batter Balls Faced'
          feature_columns = ['Batter Runs', 'Batter Balls Faced']

          # Step 1: Transform features
          vector_assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
          assembled_data = vector_assembler.transform(data)

          # Step 2: Split data into training and test sets
          train_data, test_data = assembled_data.randomSplit([0.7, 0.3])

          # Step 3: Train the model
          lr_model = lr.fit(train_data)

          # Step 4: Make predictions on test data
          predictions = lr_model.transform(test_data)

          # Step 5: Evaluate the model
          evaluator = RegressionEvaluator(labelCol="Innings Runs", predictionCol="prediction", metricName="rmse")
          rmse = evaluator.evaluate(predictions)

          print(f"Root Mean Squared Error (RMSE) on test data = {rmse}")
```

Inspect the Model Coefficients:



localhost:8888/notebooks/ALY6110_FinalProject__Group4.ipynb
jupyter ALY6110_FinalProject__Group4 Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code

```
          Root Mean Squared Error (RMSE) on test data = 43.610651011010425

In [172]: print("Coefficients:", lr_model.coefficients)
          print("Intercept:", lr_model.intercept)

          Coefficients: [2.5852944648883938,1.2170935757021122]
          Intercept: 50.62393154454576

In [173]: # R-squared
          r2_evaluator = RegressionEvaluator(labelCol="Innings Runs", predictionCol="prediction", metricName="r2")
          r2 = r2_evaluator.evaluate(predictions)
          print(f"R-squared on test data = {r2}")

          # Mean Absolute Error
          mae_evaluator = RegressionEvaluator(labelCol="Innings Runs", predictionCol="prediction", metricName="mae")
          mae = mae_evaluator.evaluate(predictions)
          print(f"Mean Absolute Error (MAE) on test data = {mae}")

          23/10/22 19:31:45 WARN CSVHeaderChecker: CSV header does not conform to the schema.
           Header: , Match ID, Date, Venue, Bat First, Bat Second, Innings, Over, Ball, Batter, Non Striker, Bowler, Batter R
          uns, Extra Runs, Runs From Ball, Ball Rebowled, Extra Type, Wicket, Method, Player Out, Innings Runs, Innings Wicke
          ts, Target Score, Runs to Get, Balls Remaining, Winner, Chased Successfully, Total Batter Runs, Total Non Striker R
          uns, Batter Balls Faced, Non Striker Balls Faced, Player Out Runs, Player Out Balls Faced, Bowler Runs Conceded, Va
          lid Ball
           Schema: _c0, Match ID, Date, Venue, Bat First, Bat Second, Innings, Over, Ball, Batter, Non Striker, Bowler, Batte
          r Runs, Extra Runs, Runs From Ball, Ball Rebowled, Extra Type, Wicket, Method, Player Out, Innings Runs, Innings Wi
          ckets, Target Score, Runs to Get, Balls Remaining, Winner, Chased Successfully, Total Batter Runs, Total Non Strike
          r Runs, Batter Balls Faced, Non Striker Balls Faced, Player Out Runs, Player Out Balls Faced, Bowler Runs Conceded,
          Valid Ball
          Expected: _c0 but found:
          CSV file: file:///Users/sameekshabs/Downloads/T20I%20Cricket%20Ball-by-Ball%20Data%20(2003%20-%202023).csv

          R-squared on test data = 0.10965458611466727
```



```
Mean Absolute Error (MAE) on test data = 35.74220697193719
```

This table appears to show the results of a regression prediction for a variable called "Innings Runs". Here's a summary of the data:

Columns:

- Innings Runs: This is the actual value of "Innings Runs" for each observation in the dataset.
- Prediction: This column shows the predicted value of "Innings Runs" generated by the regression model.
- Residual: This represents the difference between the actual value and the predicted value, calculated as

Residual = Innings Runs − prediction

residual=Innings Runs−prediction.

Interpretation:

The residuals provide insights into the model's accuracy. A residual of zero indicates that the prediction was perfect for that observation. A positive residual means the model under-predicted, and a negative residual means the model over-predicted.

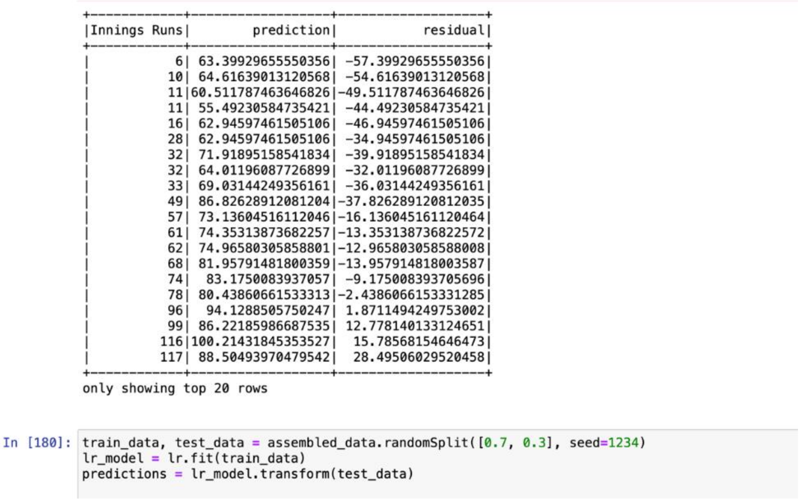
For example, in the first row, the actual value was 6, but the model predicted 63.4, resulting in a large negative residual of -57.4. This indicates a significant over-prediction by the model.

Similarly, in the last row, the actual value was 117, and the model predicted 88.5, leading to a positive residual of 28.5, indicating under-prediction.

Model Performance:

By looking at the table, it's evident that for some observations, the model's predictions are quite off from the actual values (as evidenced by the high residuals). For others, the predictions are closer to the actual values.

This kind of table can help diagnose specific instances where the model performs poorly or well. However, for a comprehensive evaluation of model performance, we would typically look at summary statistics or visualizations of the residuals and use metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-squared, etc.



```
+------------+------------------+------------------+
|Innings Runs|        prediction|          residual|
+------------+------------------+------------------+
|           6| 63.39929655550356| -57.39929655550356|
|          10| 64.61639013120568| -54.61639013120568|
|          11|60.511787463646826|-49.511787463646826|
|          11| 55.49230584735421| -44.49230584735421|
|          16| 62.94597461505106| -46.94597461505106|
|          28| 62.94597461505106| -34.94597461505106|
|          32| 71.91895158541834| -39.91895158541834|
|          32| 64.01196087726899| -32.01196087726899|
|          33| 69.03144249356161| -36.03144249356161|
|          49| 86.82628912081204|-37.826289120812035|
|          57| 73.13604516112046|-16.136045161120464|
|          61| 74.35313873682257|-13.353138736822572|
|          62| 74.96580305858801|-12.965803058588008|
|          68| 81.95791481800359|-13.957914818003587|
|          74|  83.1750083937057| -9.175008393705696|
|          78| 80.43860661533313|-2.4386066153331285|
|          96|  94.1288505750247| 1.8711494249753002|
|          99| 86.22185986687535| 12.778140133124651|
|         116|100.21431845353527|  15.78568154646473|
|         117| 88.50493970479542|  28.49506029520458|
+------------+------------------+------------------+
only showing top 20 rows
```

```
In [180]: train_data, test_data = assembled_data.randomSplit([0.7, 0.3], seed=1234)
          lr_model = lr.fit(train_data)
          predictions = lr_model.transform(test_data)
```

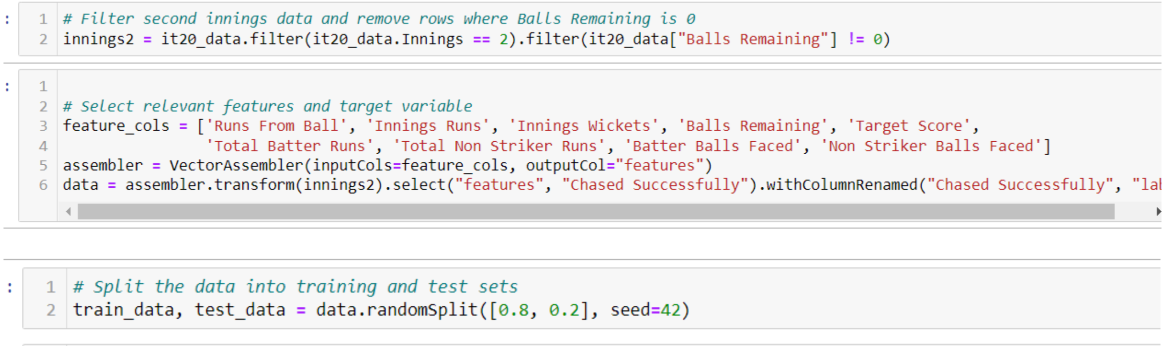
A simple linear regression model is thus produced. In summary, this table shows each regression model prediction and how it differs from the actual values, depending on your needs. Such data analysis aids in the assessment and improvement of models.

Logistic Regression for Second Innings Chasing Success Prediction

The objective of this analysis was to develop a predictive model for determining the success of chasing a target in the context of cricket matches. To achieve this, we employed a Logistic Regression algorithm using relevant match and innings-related features.

Data Preprocessing:

The dataset was preprocessed by selecting specific features deemed essential for predicting the outcome. These features included 'Runs From Ball', 'Innings Runs', 'Innings Wickets', 'Balls Remaining', 'Target Score', 'Total Batter Runs', 'Total Non-Striker Runs', 'Batter Balls Faced', and 'Non-Striker Balls Faced'. These features were combined into a vector using the Vector Assembler, and the resulting dataset was split into training and test sets (80% training, 20% test) to facilitate model evaluation.



```
# Filter second innings data and remove rows where Balls Remaining is 0
innings2 = it20_data.filter(it20_data.Innings == 2).filter(it20_data["Balls Remaining"] != 0)
```

```

# Select relevant features and target variable
feature_cols = ['Runs From Ball', 'Innings Runs', 'Innings Wickets', 'Balls Remaining', 'Target Score',
                'Total Batter Runs', 'Total Non Striker Runs', 'Batter Balls Faced', 'Non Striker Balls Faced']
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data = assembler.transform(innings2).select("features", "Chased Successfully").withColumnRenamed("Chased Successfully", "lal
```

```
# Split the data into training and test sets
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)
```

Model Training and Evaluation:

The second innings data is filtered and rows where 'Balls Remaining' is 0 are removed. Relevant features and the target variable ('Chased Successfully') are selected. The features are assembled into a vector using VectorAssembler, and the data is split into training and test sets with an 80-20 split ratio. A Logistic Regression model is instantiated with specified parameters, and the model is trained on the training data. The trained model is used to make predictions on the test data, and the performance of the model is evaluated based on its ability to predict whether a target score is chased successfully or not.

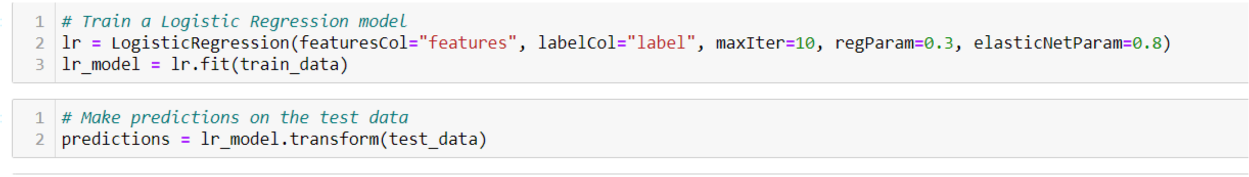
A Logistic Regression model was trained on the training dataset with the following parameters:

Maximum Iterations: 10

Regularization Parameter (regParam): 0.3

Elastic Net Parameter (elasticNetParam): 0.8

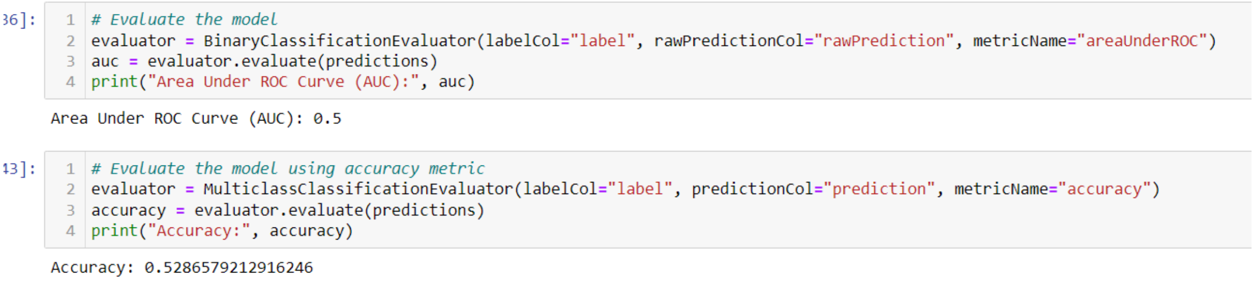
The trained model was then used to make predictions on the test dataset. To evaluate the model's performance, the Area Under the ROC Curve (AUC) metric was employed. AUC measures the ability of the model to distinguish between positive and negative classes, with a value of 0.5 indicating random chance.



```
# Train a Logistic Regression model
lr = LogisticRegression(featuresCol="features", labelCol="label", maxIter=10, regParam=0.3, elasticNetParam=0.8)
lr_model = lr.fit(train_data)
```

```
# Make predictions on the test data
predictions = lr_model.transform(test_data)
```

Results:



```
# Evaluate the model
evaluator = BinaryClassificationEvaluator(labelCol="label", rawPredictionCol="rawPrediction", metricName="areaUnderROC")
auc = evaluator.evaluate(predictions)
print("Area Under ROC Curve (AUC):", auc)
```

```
Area Under ROC Curve (AUC): 0.5
```

```
# Evaluate the model using accuracy metric
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.5286579212916246
```

Upon evaluating the model, the following results were obtained:

Area Under ROC Curve (AUC): 0.5

Accuracy: 52.87%

The AUC value of 0.5 indicates that the model performs no better than random chance, suggesting that the selected features and the Logistic Regression algorithm, as configured, are not effective in predicting whether a team can successfully chase a target in a cricket match.

The Logistic Regression model developed in this analysis, utilizing the chosen features, predictive power for determining the success of chasing a target in cricket matches. Further investigation, including feature engineering, exploring different algorithms, and potentially gathering additional data, may be necessary to build a more accurate predictive model for this scenario. It is essential to continually refine the model and consider domain-specific insights to improve its performance and usefulness in practical applications.

Conclusion

Model Assessment and Business Impact:

Model 1 (Linear Regression for Innings Runs Prediction):

Business Impact: This detailed analysis can aid team management in understanding player strengths and weaknesses, enabling strategic decisions about lineups and training. Moreover, it enhances fan engagement by providing deeper insights into players' contributions, fostering a more informed and engaged fan base.

Model 2 (Logistic Regression for Match Outcome Prediction):

Business Impact: Understanding the dynamics of successful chases is pivotal for both team strategy and fan engagement. Teams can strategize their chases better, leading to more victories. Fans, armed with insights into successful chase patterns, can engage in more informed discussions, enhancing their overall experience.

Choosing the Right Model:

Model Strengths: Model 1 excels in player-level analysis, allowing fine-tuning of individual performances. Model 2, on the other hand, provides a broader perspective, essential for strategic team decisions.

Holistic Approach: Combining insights from both models enables a comprehensive understanding. Team management can use player-specific insights (from Model 1) alongside overall match outcome patterns (from Model 2) to devise robust strategies.

Continuous Enhancement: Both models emphasize the iterative nature of model development. Regular updates and refinements, incorporating new data and features, are crucial. Continuous enhancement ensures models adapt to evolving player dynamics and match scenarios, maintaining their relevance and accuracy over time.

In conclusion, leveraging the strengths of both models provides a multi-faceted approach to enhancing team performance and fan engagement. By utilizing player-specific insights and overall match outcome patterns, teams can make data-driven decisions, leading to improved on-field performances. Simultaneously, fans benefit from a richer understanding of the game, fostering a more engaging and interactive fan experience. The iterative process of model enhancement ensures a dynamic and adaptive analytical framework, aligning with the evolving landscape of Twenty20 Internationals. In summary, Model 1's approach is more localized, examining individual predictions, while Model 2 adopts a broader perspective, addressing the model's overall predictive capacity. Both conclusions stress the iterative nature of model development and the need for continuous enhancement to ensure accurate and reliable predictions in their respective domains.

REFERENCES

1. T20I Cricket Ball-by-Ball Data (2003 - 2023). (2023, September 20). Kaggle. https://www.kaggle.com/datasets/jamiewelsh2/ball-by-ball-it20
2. LogisticRegression — PySpark 3.5.0 documentation. (n.d.). https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.classification.LogisticRegression.html
3. Jagdeesh. (2023). PySpark Logistic Regression – How to Build and Evaluate Logistic Regression Models using PySpark MLlib. Machine Learning Plus. https://www.machinelearningplus.com/pyspark/pyspark-logistic-regression/
4. Jagdeesh. (2023a). PySpark Linear Regression – How to Build and Evaluate Linear Regression Models using PySpark MLlib. Machine Learning Plus. https://www.machinelearningplus.com/pyspark/pyspark-linear-regression/
5. Wickramasinghe, I. (2022). Applications of Machine Learning in cricket: A systematic review. Machine Learning With Applications, 10, 100435. https://doi.org/10.1016/j.mlwa.2022.100435
6. https://public.tableau.com/views/sam_16974179697290/Dashboard1?:language=en-GB&publish=yes&:display_count=n&:origin=viz_share_link
7. https://app.powerbi.com/links/bho4aPp10Z?ctid=a8eec281-aaa3-4dae-ac9b-9a398b9215e7&pbi_source=linkShare