St. Clair College @ Ace Acumen

CAPSTONE PROJECT

Cricket Score Prediction

STUDENTS NAME: Mayur Donda (0795322)
Jeelkumar Jiyani (0805063)
Krupa Koladiya (0806550)
Isha Jivani (0801943)

INSTRUCTOR: Sarama Shehmir & Francisco Guevara

(i) Problem formulation and data sources

Abstract

Sports forecasting is one of the goals of sports analysis, which tries to help decision makers gain an advantage over competitors. Sports data analysis is very common. The International Cricket Council (ICC) data analysis results are commonly used in cricket. The difficulty of this activity is determined by the collecting of historical data, the acquisition of data for future occurrences, the expertise necessary to understand the obtained data, and a variety of other factors.

Cricket is India's most popular sport, and it is played in all regions in various formats such as T20, ODI, and Test. The Indian Premier League (IPL), a national cricket league, features players from Indian regional clubs, the national squad, and international teams.

When watching a match in the IPL, viewers make their own predictions. They base their forecasts on the facts they have available and calculate the score using numerous statistics and records. As a result, there is a sizable market for algorithms that predict the best team score, which is more important. We will offer predictions for every IPL match that has already occurred. In this method, machine learning techniques are utilized to predict the outcomes of the matches.

Problem Statement

The main objective of this study is to forecast the projected of1st innings score of the team in IPL cricket match in 2023.
Introducing a web application that uses machine learning technology to predict the scores of the most popular IPL (Indian Premier League) cricket matches with greater accuracy than the methods currently in use.

We are going to use different dataset to extract the relevant data of the IPL matches. We are going to apply different EDA process, cleaning process to make the data useful for the problem statement. After that, we have to perform different steps to reach the final model. Then, in this research, utilizing different machine learning algorithms and we are going to suggest a model for predicting the score of the IPL matches.

At the end, our proposed system consists a model that has one part in which we can predict the of score. In this
the score prediction is done with the help of different machine learning algorithm.

Data Sources

- We did a lot research to collect the IPL match data from the different resources.

- At the end, we manage to collect the 3 data set which are preferable for this topic. That 3 data sets are as below:

(a) https://data.world/maven356/indian-premiere-leaguelatest/workspace/data-dictionary

(b) https://cricsheet.org/matches/

(c) https://www.kaggle.com/datasets/patrickb1912/ipl-complete-dataset-20082020

The dataset under consideration is one that was compiled from several sources, and it requires cleaning, transformation, and understanding of each variable in order to become useable.

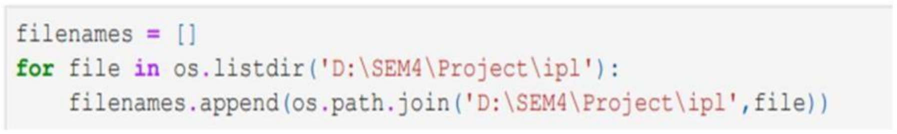
The data from the website must first be extract, and then the raw dataset must be cleaned using Python, Tableau and MS Excel techniques. We will learn which factors are crucial and how they affect the score of the team in IPL through this project.

Exploratory Data Analysis

- Data exploration, commonly referred to as exploratory data analysis (EDA), is a step in the data analysis process when various techniques are employed to better understand the dataset being used.

- **Data Extraction:-**

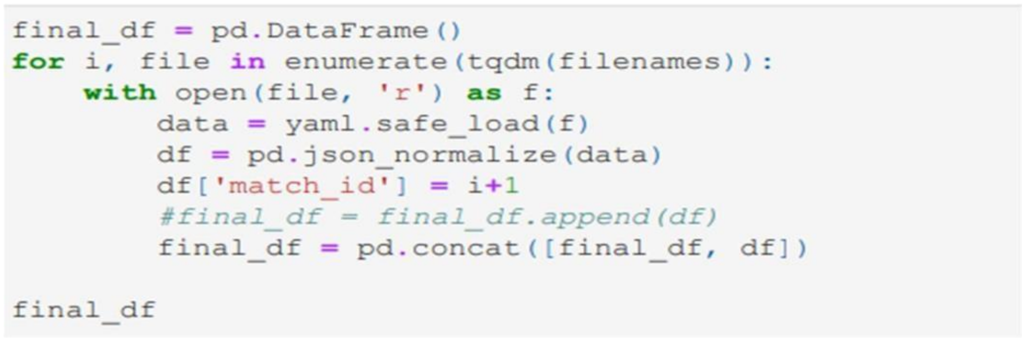
- To starts with, we imported necessary libraries(for this example numpy, matplotlib, yaml, os, and pickle) and loaded the yaml file dataset.



```
filenames = []
for file in os.listdir('D:\SEM4\Project\ipl'):
    filenames.append(os.path.join('D:\SEM4\Project\ipl',file))
```

(1.1)

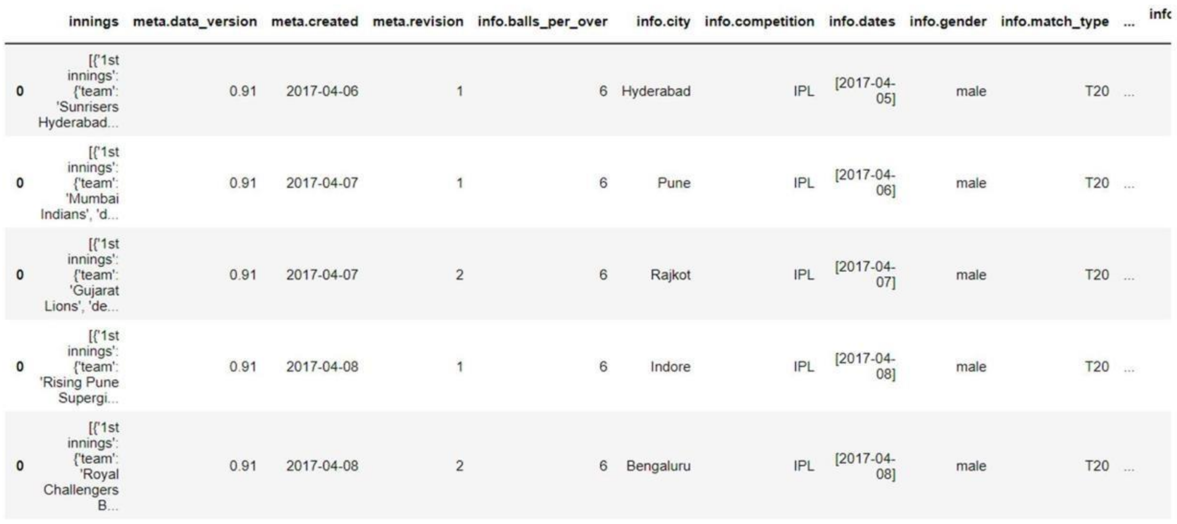
- Importing the new match id columns and loading the yaml data file into the data frame.



```
final_df = pd.DataFrame()
for i, file in enumerate(tqdm(filenames)):
    with open(file, 'r') as f:
        data = yaml.safe_load(f)
        df = pd.json_normalize(data)
        df['match_id'] = i+1
        #final_df = final_df.append(df)
        final_df = pd.concat([final_df, df])

final_df
```

(1.2)

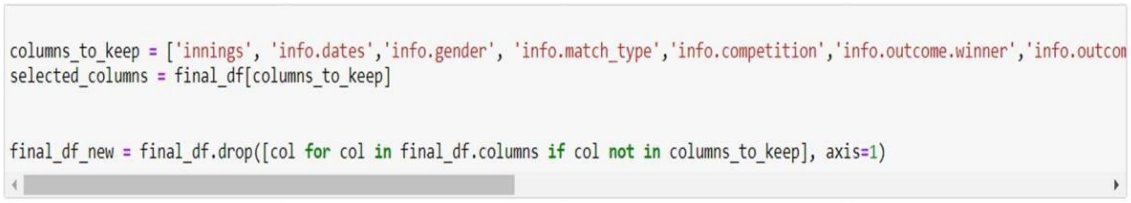
- So, we have the following data frame(couldn't capture complete data in screenshot):



	innings	meta.data_version	meta.created	meta.revision	info.balls_per_over	info.city	info.competition	info.dates	info.gender	info.match_type	...	inf
0	[{'1st innings': {'team': 'Sunrisers Hyderabad...	0.91	2017-04-06	1	6	Hyderabad	IPL	[2017-04-05]	male	T20	...	
0	[{'1st innings': {'team': 'Mumbai Indians', 'd...	0.91	2017-04-07	1	6	Pune	IPL	[2017-04-06]	male	T20	...	
0	[{'1st innings': {'team': 'Gujarat Lions', 'de...	0.91	2017-04-07	2	6	Rajkot	IPL	[2017-04-07]	male	T20	...	
0	[{'1st innings': {'team': 'Rising Pune Supergi...	0.91	2017-04-08	1	6	Indore	IPL	[2017-04-08]	male	T20	...	
0	[{'1st innings': {'team': 'Royal Challengers B...	0.91	2017-04-08	2	6	Bengaluru	IPL	[2017-04-08]	male	T20	...	

(1.3)

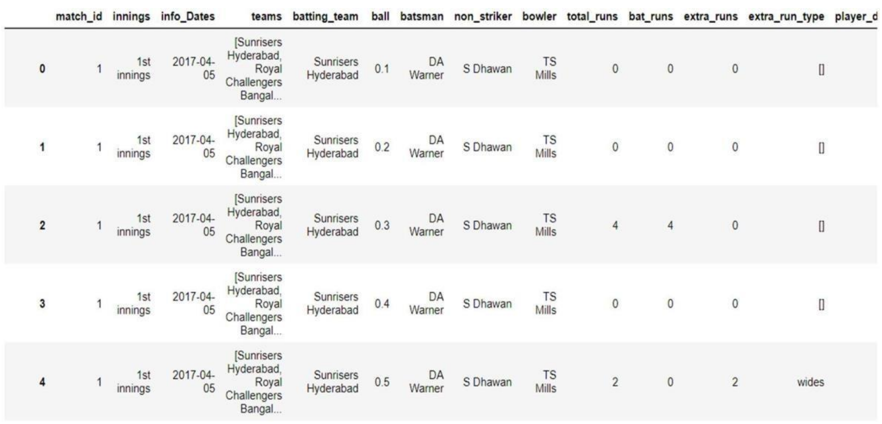
- Dropping other columns and selecting the necessary ones.



```
columns_to_keep = ['innings', 'info.dates','info.gender', 'info.match_type','info.competition','info.outcome.winner','info.outcom
selected_columns = final_df[columns_to_keep]


final_df_new = final_df.drop([col for col in final_df.columns if col not in columns_to_keep], axis=1)
```

(1.4)

- Analyzing a new data frame that has 16 columns and 950 rows overall. In total, 950 IPL matches were played between the years 2008 and 2022, and all the data from those matches is shown in this dataset. Under the innings columns, the ball-by-ball analysis is presented in json directory format.
- Examining the innings columns data, from which we're creating a new dataset for our project.
- The innings column displays information about the bowler, the batsman, the run, etc. for each match's entire innings, ball by ball.
- creating a new dataframe called delivery data in this data we are going to create new columns deliveries, batsman, total run, etc. also we collect the match id and date information from the *matc*h dataset.
- So, this is our final dataset.

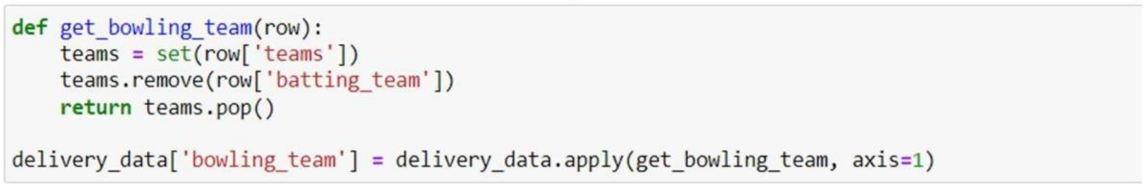


	match_id	innings	info_Dates	teams	batting_team	ball	batsman	non_striker	bowler	total_runs	bat_runs	extra_runs	extra_run_type	player_d
0	1	1st innings	2017-04-05	[Sunrisers Hyderabad, Royal Challengers Bangal...	Sunrisers Hyderabad	0.1	DA Warner	S Dhawan	TS Mills	0	0	0	[]	
1	1	1st innings	2017-04-05	[Sunrisers Hyderabad, Royal Challengers Bangal...	Sunrisers Hyderabad	0.2	DA Warner	S Dhawan	TS Mills	0	0	0	[]	
2	1	1st innings	2017-04-05	[Sunrisers Hyderabad, Royal Challengers Bangal...	Sunrisers Hyderabad	0.3	DA Warner	S Dhawan	TS Mills	4	4	0	[]	
3	1	1st innings	2017-04-05	[Sunrisers Hyderabad, Royal Challengers Bangal...	Sunrisers Hyderabad	0.4	DA Warner	S Dhawan	TS Mills	0	0	0	[]	
4	1	1st innings	2017-04-05	[Sunrisers Hyderabad, Royal Challengers Bangal...	Sunrisers Hyderabad	0.5	DA Warner	S Dhawan	TS Mills	2	0	2	wides	

(1.5)

- Then, adding columns for bowling teams.

(1.6)



```python
def get_bowling_team(row):
    teams = set(row['teams'])
    teams.remove(row['batting_team'])
    return teams.pop()

delivery_data['bowling_team'] = delivery_data.apply(get_bowling_team, axis=1)
```

- **Understanding the variables,**

- We have a number of 225954 rows and 18 columns for our dataset.

- As we can see, we got these variables:

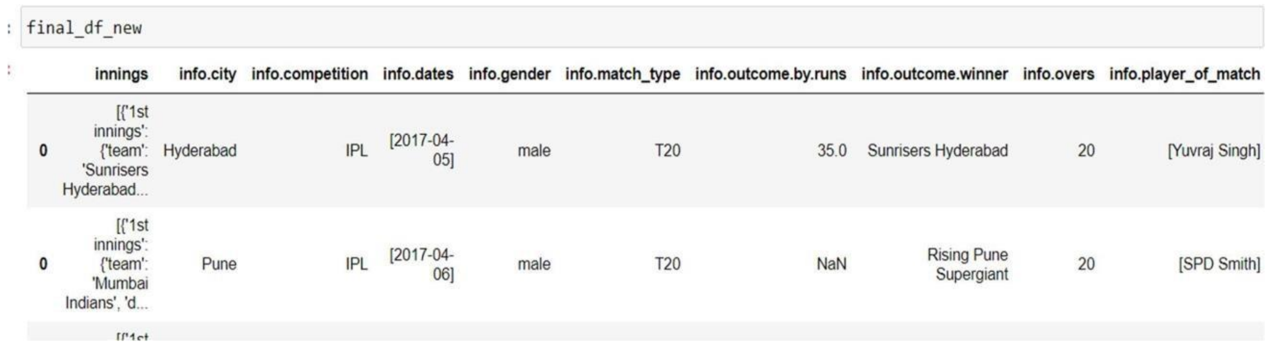
1. **match_id**: Distinct ID assigned to a Match (referred from Matches Table)
2. **Innings**: 1 – First Innings, 2 – Second Innings, and super over
3. **info_Dates**: date of the match has been played.
4. **batting_team**: Team Batting currently.
5. **bowling_team**: Team Bowling currently.
6. **Ball**: Current Ball and over.
7. **Batsman**: Name of the Batsman on Strike
8. **Non-Striker**: Name of the Batsman at the non-striker end
9. **Bowler**: Name of the Bowler for the Over in progress
10. **bat_runs**: Runs scored by the Batsman.
11. **Extra_Runs**: extra run is given by the bowling team.
12. **extra_run_type**: Types of the extra run
13. **Total_Runs**: Extra Runs + Runs scored by the Batsman.
14. **Player_Dismissed**: Name of the dismissed player (if any)
15. **wicket_type**: Reason for the dismissal of the player (if any)
16. **Fielder:** name of the Fielder responsible for dismissal (if any)
17. **City**: a city where a match has been played
18. **venue**: ground name where the match has been played

- **Cleaning our dataset,**

- Data Cleaning is the process of detecting and correcting inaccurate records from the database.

a. Removing unwanted variables:

- During the data extraction process, a number of unnecessary columns from the raw data set were deleted. The raw data collection has 827 columns and 950 rows overall. After analyzing all the columns, we only keep the columns for innings, "info.dates," "info.gender," "info.match type," "info.competition," "info.outcome.winner," "info.outcome.by.runs," "info.overs," "info.player of match," "info.teams," "info.toss.decision," "info.toss.winner," "info.umpires," "info.venue," "info.city," and "match_id".



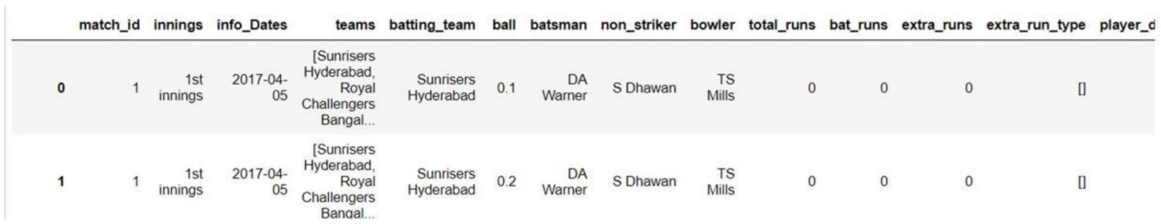
```
final_df_new
```

	innings	info.city	info.competition	info.dates	info.gender	info.match_type	info.outcome.by.runs	info.outcome.winner	info.overs	info.player_of_match
0	[{'1st innings': {'team': 'Sunrisers Hyderabad...	Hyderabad	IPL	[2017-04-05]	male	T20	35.0	Sunrisers Hyderabad	20	[Yuvraj Singh]
0	[{'1st innings': {'team': 'Mumbai Indians', 'd...	Pune	IPL	[2017-04-06]	male	T20	NaN	Rising Pune Supergiant	20	[SPD Smith]
	[{'1st									

(1.7)

- However, we are going to use the inning dataset which is JSON format and after extract the inning dataset we have total total 225954 row and 18 columns for this dataset

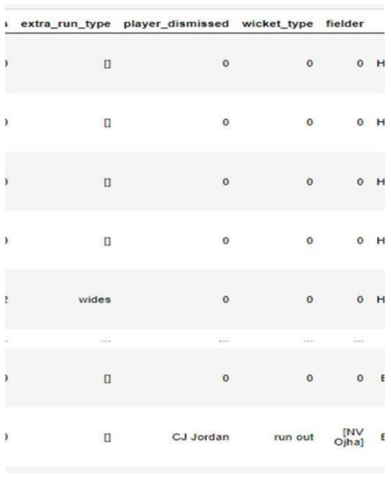
(1.8)



	match_id	innings	info_Dates	teams	batting_team	ball	batsman	non_striker	bowler	total_runs	bat_runs	extra_runs	extra_run_type	player_d
0	1	1st innings	2017-04-05	[Sunrisers Hyderabad, Royal Challengers Bangal...	Sunrisers Hyderabad	0.1	DA Warner	S Dhawan	TS Mills	0	0	0	[]	
1	1	1st innings	2017-04-05	[Sunrisers Hyderabad, Royal Challengers Bangal...	Sunrisers Hyderabad	0.2	DA Warner	S Dhawan	TS Mills	0	0	0	[]	

b. Replacing the value:

- We removed the [', and '] from the fielder text value and replaced the [] for wicket type columns with 0.



	extra_run_type	player_dismissed	wicket_type	fielder	
)	[]	0	0	0	H
)	[]	0	0	0	H
)	[]	0	0	0	H
)	[]	0	0	0	H
2	wides	0	0	0	H
...	...	...	...	...	
)	[]	0	0	0	E
)	[]	CJ Jordan	run out	[NV Ojha]	E

(1.9)

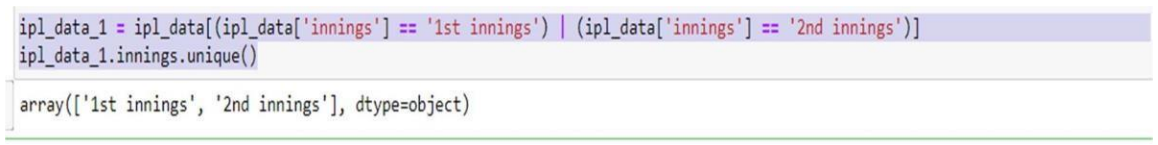
c. Duplicate data:

- There is no duplication value in this dataset.

d. Irrelevant and Incorrect Data:

- Removing unnecessary data from the innings column, such as super over rows. We only have data for the first and second innings after the data has been removed. After EDA, we will eliminate the second innings' data since we will only be working with the first innings' data.

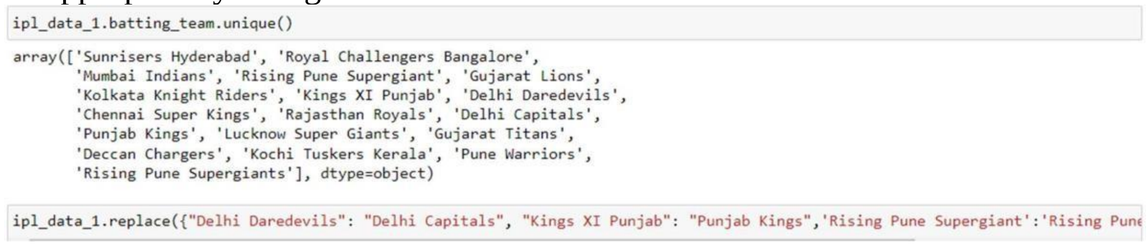
(1.10)



```
ipl_data_1 = ipl_data[(ipl_data['innings'] == '1st innings') | (ipl_data['innings'] == '2nd innings')]
ipl_data_1.innings.unique()
```

```
array(['1st innings', '2nd innings'], dtype=object)
```

- Few team names are misdescribed, too. We correctly and appropriately change the name of their teams.



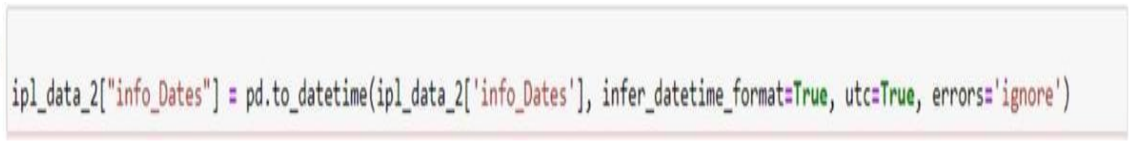
```
ipl_data_1.batting_team.unique()
```

```
array(['Sunrisers Hyderabad', 'Royal Challengers Bangalore',
       'Mumbai Indians', 'Rising Pune Supergiant', 'Gujarat Lions',
       'Kolkata Knight Riders', 'Kings XI Punjab', 'Delhi Daredevils',
       'Chennai Super Kings', 'Rajasthan Royals', 'Delhi Capitals',
       'Punjab Kings', 'Lucknow Super Giants', 'Gujarat Titans',
       'Deccan Chargers', 'Kochi Tuskers Kerala', 'Pune Warriors',
       'Rising Pune Supergiants'], dtype=object)
```

```
ipl_data_1.replace({"Delhi Daredevils": "Delhi Capitals", "Kings XI Punjab": "Punjab Kings",'Rising Pune Supergiant':'Rising Pune
```

- convert the data column to date format.

(1.11)



```
ipl_data_2["info_Dates"] = pd.to_datetime(ipl_data_2['info_Dates'], infer_datetime_format=True, utc=True, errors='ignore')
```

- **Categorical data:**

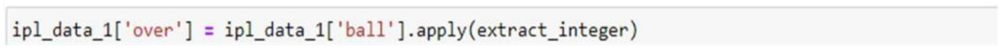
- We have a total of 13 columns in a categorical format some of going to be removed and for some columns, we are going performed data encoding.

- **Outliers:**

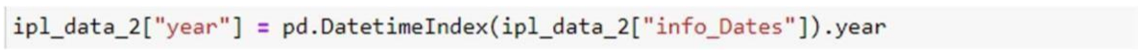
- This dataset does not have an outliner. To perform outliner, however, we must first complete feature engineering.

- **Feature Engineering and selection:**

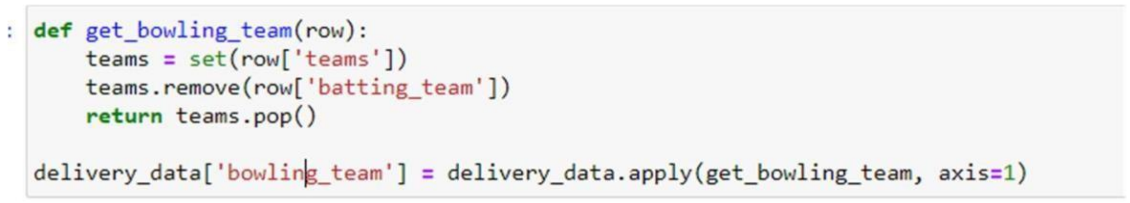
- We introduce new features such as year, Is_wicket, match_id, bowling_team, and over.



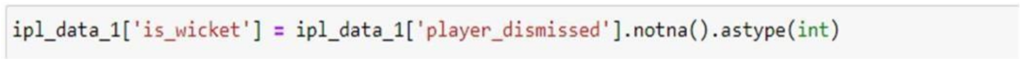
```
ipl_data_1['over'] = ipl_data_1['ball'].apply(extract_integer)
```



```
ipl_data_2["year"] = pd.DatetimeIndex(ipl_data_2["info_Dates"]).year
```



```
def get_bowling_team(row):
    teams = set(row['teams'])
    teams.remove(row['batting_team'])
    return teams.pop()

delivery_data['bowling_team'] = delivery_data.apply(get_bowling_team, axis=1)
```



```
ipl_data_1['is_wicket'] = ipl_data_1['player_dismissed'].notna().astype(int)
```

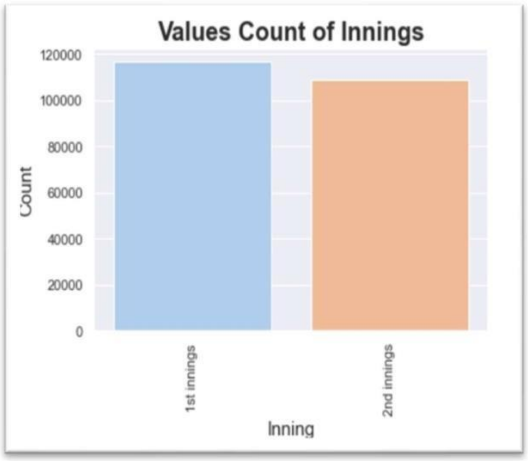
(1.12)

- we will introduce new features such as total_score, net run rate and last 5 overs runs for feature selection process.

Analyzing relationships between variables

- **Exploring the inning feature:**

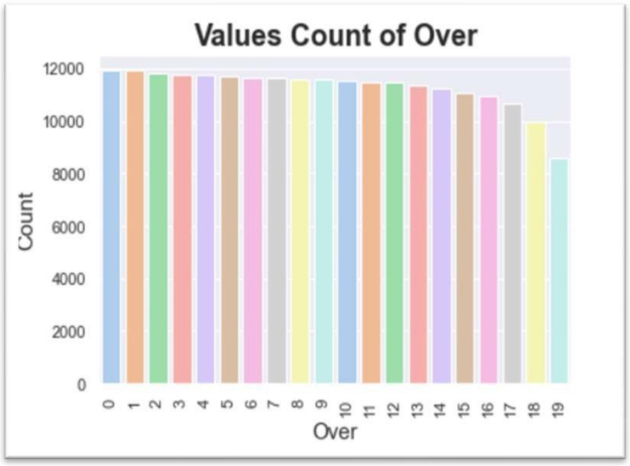
- There are two innings in the IPL, and as the first inning typically lasts longer than the second, the match may not have been completed due to rain or another factor.



(1.13)

- **Exploring the ball feature:**

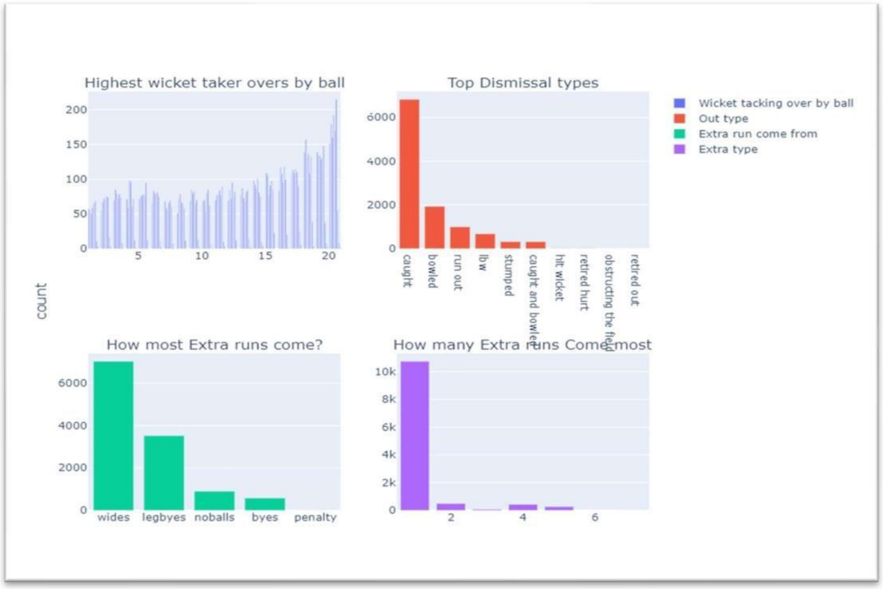
- Created a new column called over, and use it to extract over numbers from ball columns.
- Over start from 0 to 19 it here it has 20 overs matches most of the time matches were finished last over.



(1.14)

- **Exploring wicket_type, extra_run_type, extra_run, and, dismissal_type:**

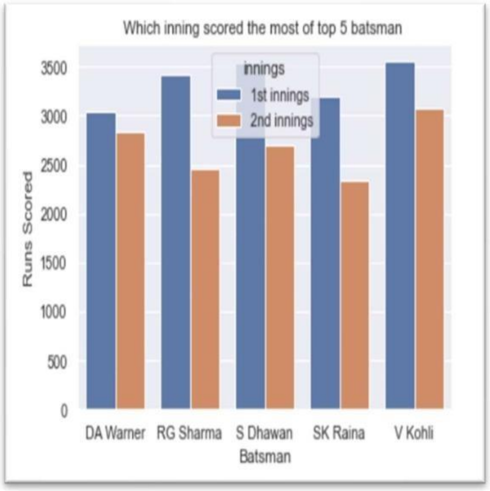
- Most extra runs are given as white balls, and most runs come from a single run. Most of the dismissals are caught out, and the last over's highest wicket was taken.

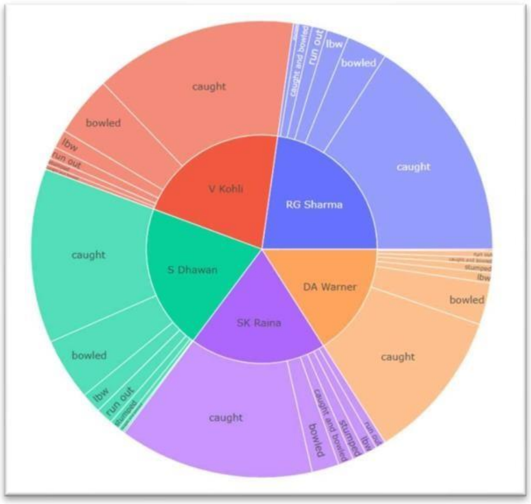


(1.15)

- **Exploring batsman and bat_run feature:**

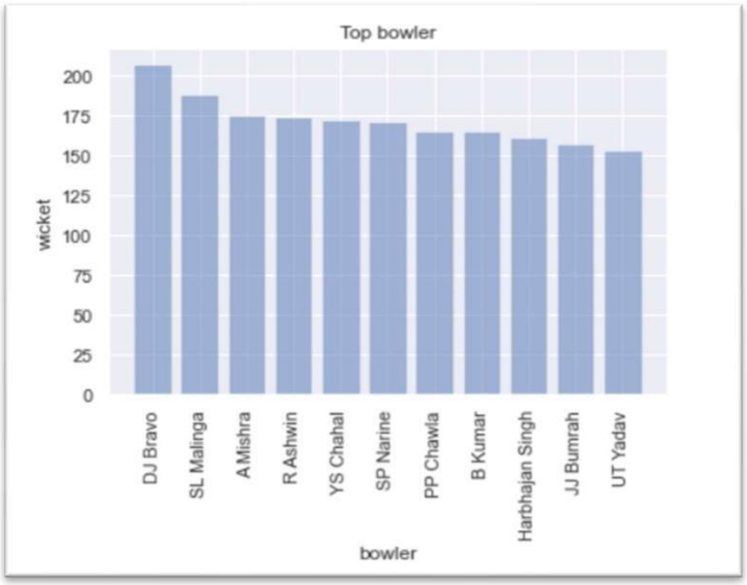
- In the IPL, Virat Kohli has scored the most runs overall and, in both innings, combined. In comparison to the second innings, Rohit Sharma scores more runs in the first innings. Most of the batsmen are out as caught and followed by bowled wickets.





- **Exploring bowler feature:**

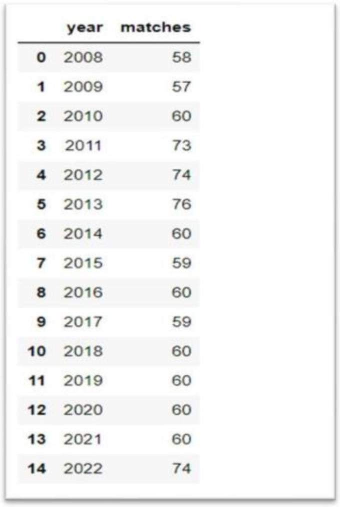
- The IPL competition has 473 bowlers in all and the most wickets taken by DJ Bravo.



(1.16)

- **Exploring date feature:**

- The highest match played 76 games in 2013 and averaged 60 games per season in all editions of IPL.

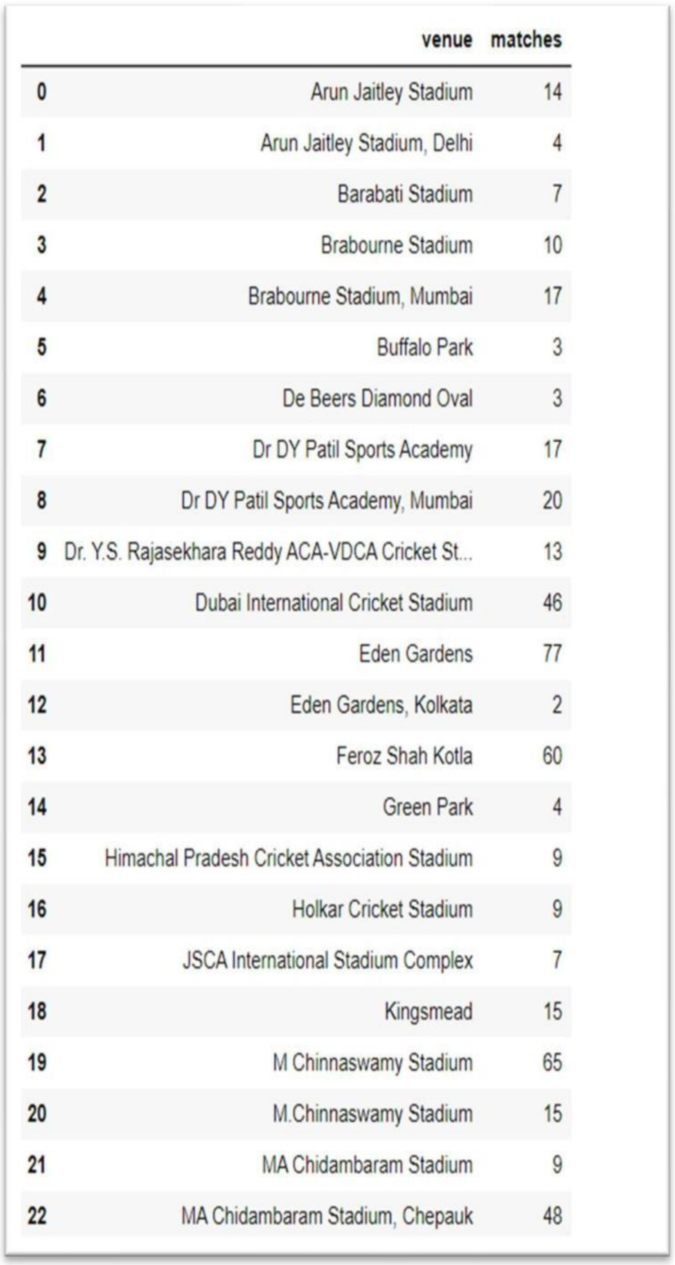


	year	matches
0	2008	58
1	2009	57
2	2010	60
3	2011	73
4	2012	74
5	2013	76
6	2014	60
7	2015	59
8	2016	60
9	2017	59
10	2018	60
11	2019	60
12	2020	60
13	2021	60
14	2022	74

(1.17)

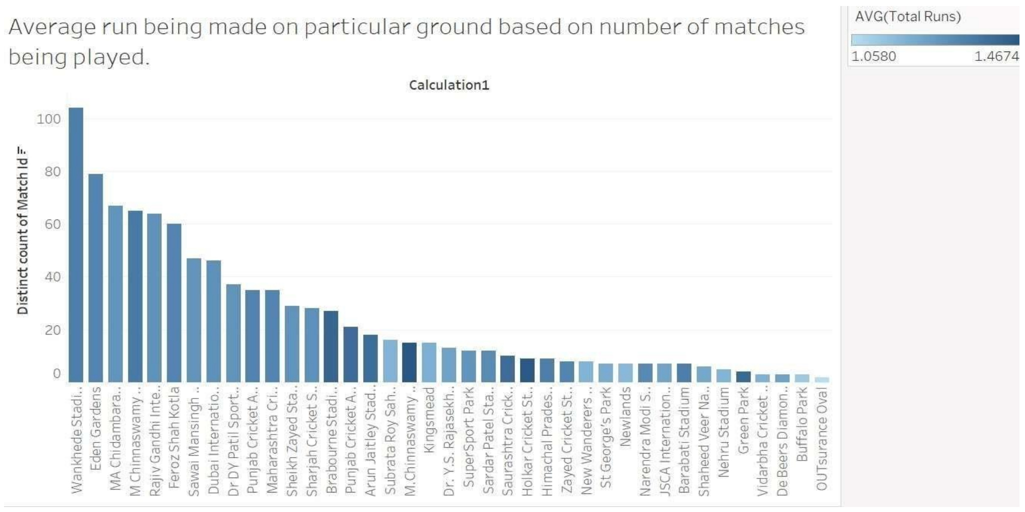
- **Exploring venue feature:**

(1.18)



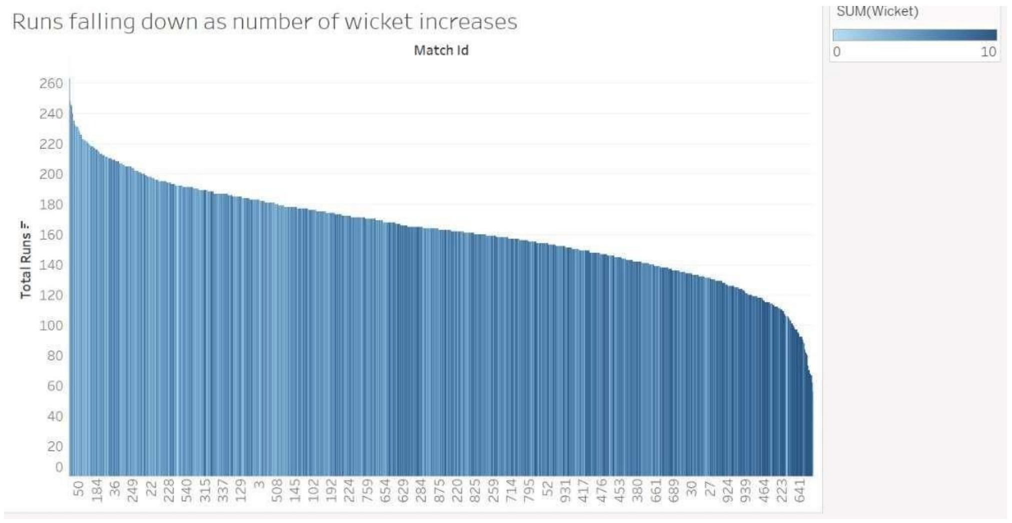
	venue	matches
0	Arun Jaitley Stadium	14
1	Arun Jaitley Stadium, Delhi	4
2	Barabati Stadium	7
3	Brabourne Stadium	10
4	Brabourne Stadium, Mumbai	17
5	Buffalo Park	3
6	De Beers Diamond Oval	3
7	Dr DY Patil Sports Academy	17
8	Dr DY Patil Sports Academy, Mumbai	20
9	Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket St...	13
10	Dubai International Cricket Stadium	46
11	Eden Gardens	77
12	Eden Gardens, Kolkata	2
13	Feroz Shah Kotla	60
14	Green Park	4
15	Himachal Pradesh Cricket Association Stadium	9
16	Holkar Cricket Stadium	9
17	JSCA International Stadium Complex	7
18	Kingsmead	15
19	M Chinnaswamy Stadium	65
20	M.Chinnaswamy Stadium	15
21	MA Chidambaram Stadium	9
22	MA Chidambaram Stadium, Chepauk	48

St. Clair College @ Ace Acumen



(1.19)

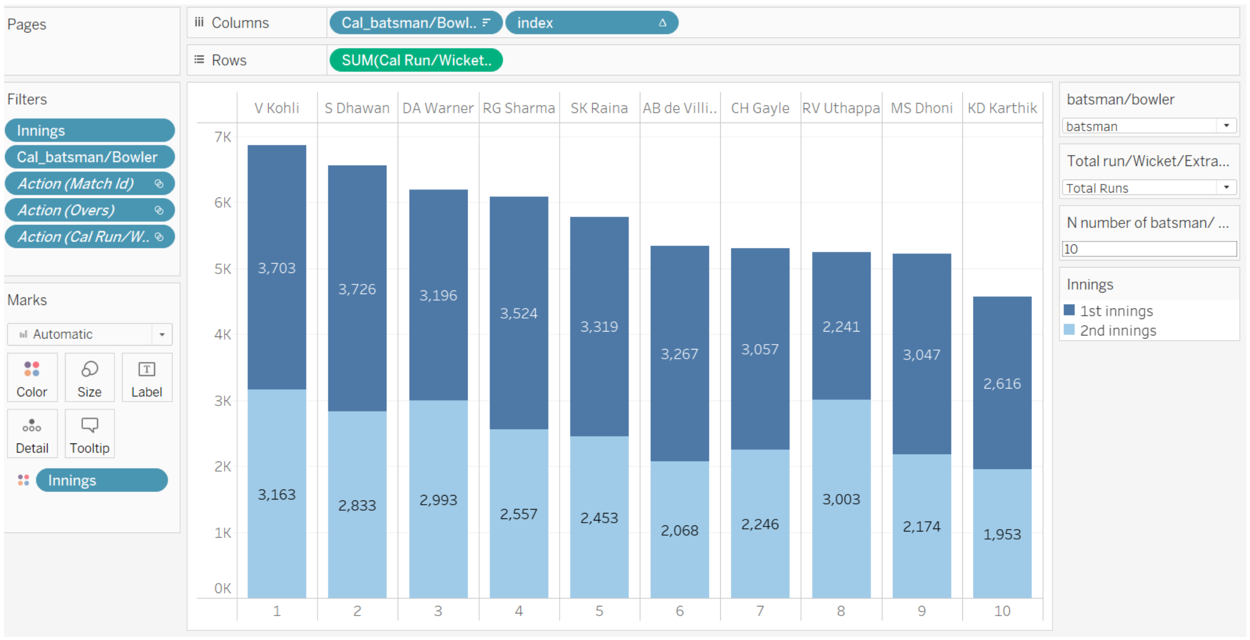
(1.20)



WEEK4
Tableau Dashboard

1). This chart provides a comprehensive worksheet that displays the performance of batsmen in an innings. The worksheet shows the Total Run, Extra Run, and Wickets taken by each batsman in the innings, as well as the runs given by the bowlers and the wickets taken by them. The data is presented in the form of a bar chart, showcasing the top 10 batsmen with the highest runs. This information is useful for fans and analysts alike, as it allows them to compare the performance of different batsmen and identify key players.

To present the data in a clear and concise manner, the worksheet also provides information on the overall performance of the bowlers. This information is essential in understanding the contribution of the bowlers to the game and the impact they have had on the outcome. The worksheet provides a complete picture of the performance of batsmen and bowlers in an innings, making it a valuable tool for anyone looking to gain insight into the game.


Pages

Columns: Cal_batsman/Bowl.. | index

Rows: SUM(Cal Run/Wicket..

Filters: Innings, Cal_batsman/Bowler, Action (Match Id), Action (Overs), Action (Cal Run/W..

Marks: Automatic | Innings

batsman/bowler: batsman

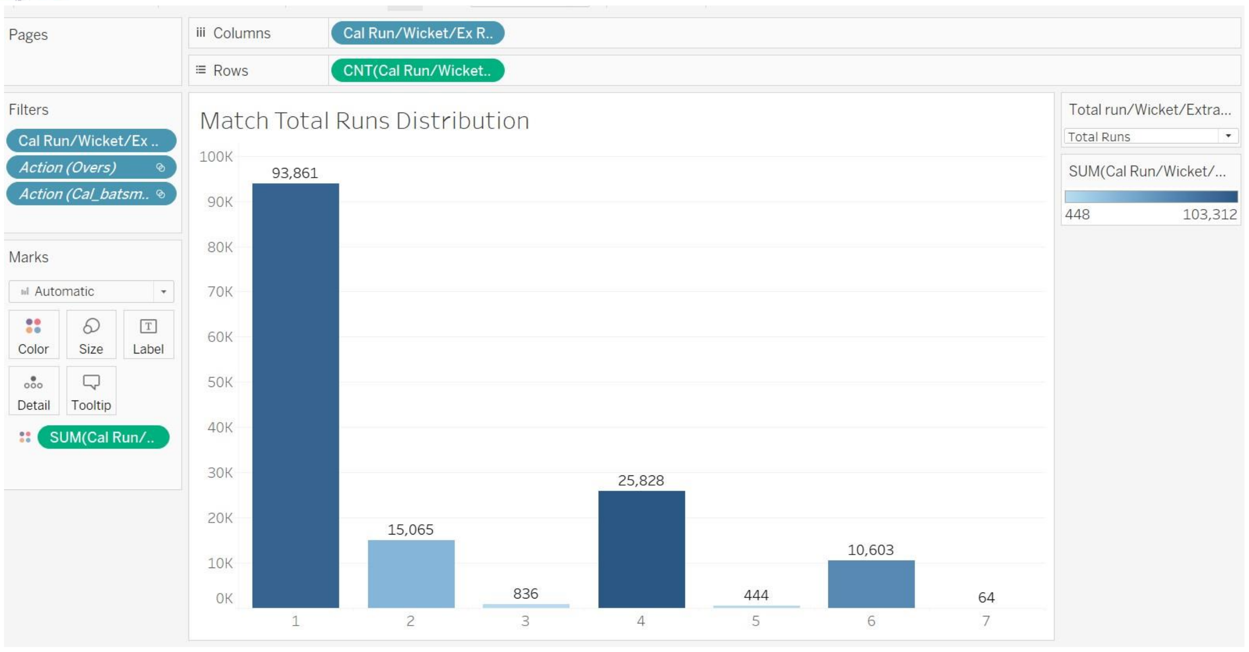
Total run/Wicket/Extra...: Total Runs

N number of batsman/ ...: 10

Innings: 🟦 1st innings, ▨ 2nd innings

(1.20)

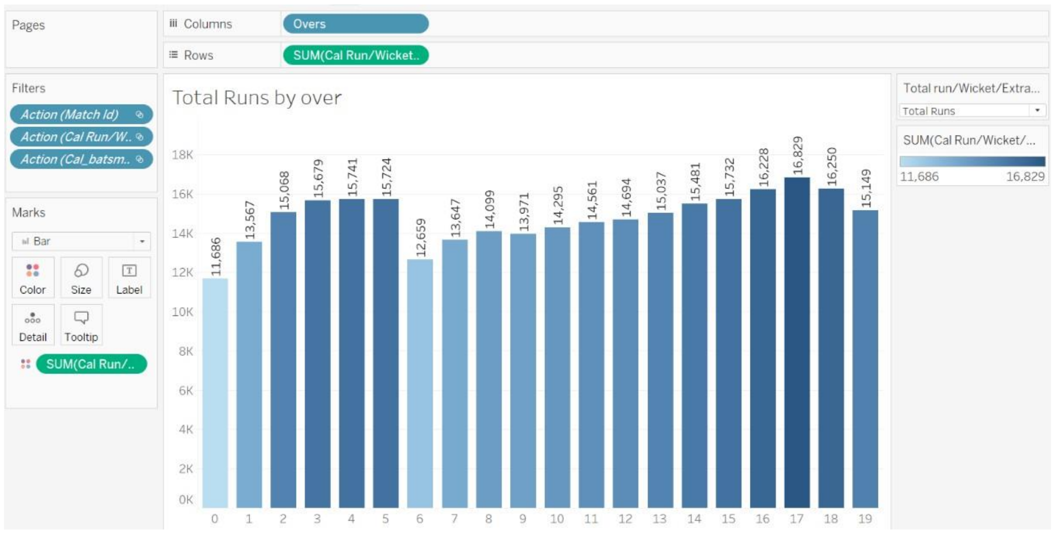
2). The Bar chart provides a comprehensive worksheet that displays the performance of batsmen in an innings. It shows the run distribution for each batsman in terms of singles, fours, sixes, and other parameters. This information allows us to check the Total Run and Extra Run made by each batsman. The worksheet also displays the wickets taken by the bowler. This information provides a comprehensive picture of the performance of each batsman in the innings and is valuable in understanding their impact on the game.



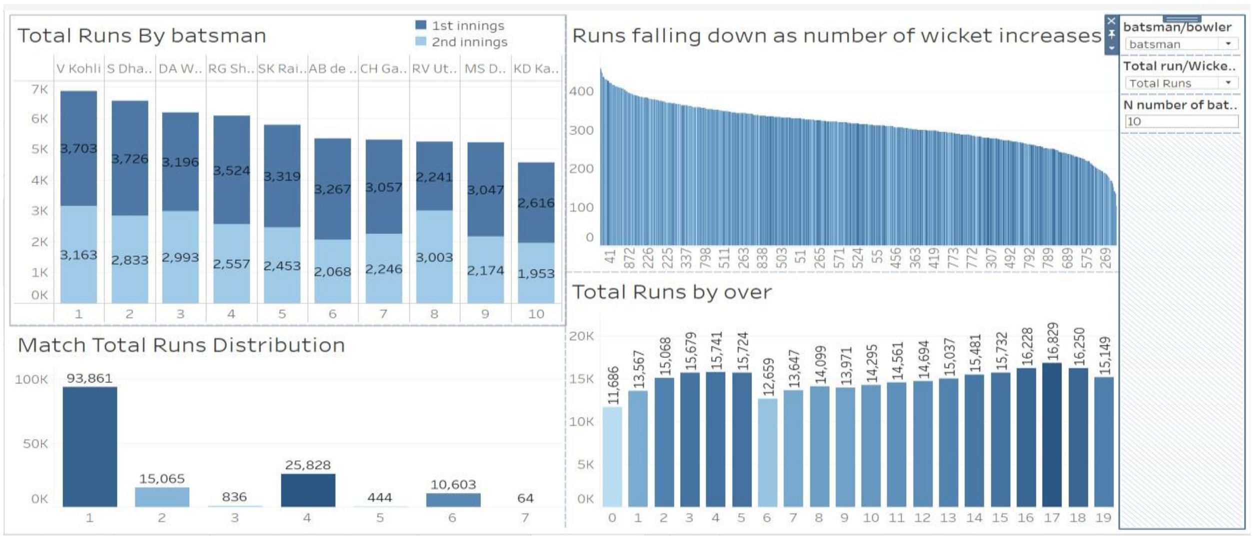
(1.21)

3). This Pictorial displays the performance of batsmen and bowlers in an innings over 20 overs. It shows the Total Run, Extra Run, and Wickets taken by each batsman and bowler over each of the 20 overs. This information provides a detailed picture of the performance of each player over each over, allowing us to see which overs they were most productive or made the most impact. The worksheet also summarizes the performance of each player by displaying the sum of all parameters for all 20 overs. This provides a useful overview of the player's form throughout the innings, allowing us to gain a better understanding of their overall performance.

(1.22)



4). Below interactive dashboard that provides a wealth of information about the performance of batsmen and bowlers in an innings. The dashboard is highly interactive, allowing users to evaluate various parameters based on their needs. It features filtered worksheets that enable users to quickly view specific columns and data points in the charts. This provides a flexible and responsive environment for exploring the performance of each player, and helps users gain a better understanding of the overall outcome of the game



(1.23)

DATA Cleaning

are performing data cleaning on the existing data set with the goal of using the cleaned data as the final data for modeling purposes.

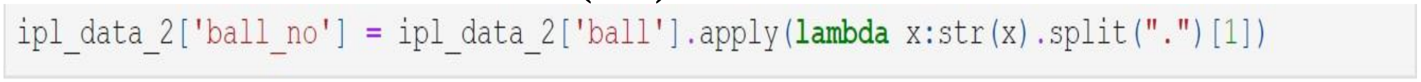
1. Eliminating the city name from the location
2. Substituting frequently occurring location names
3. Deleting information regarding the second inning.

Feature Engineering:

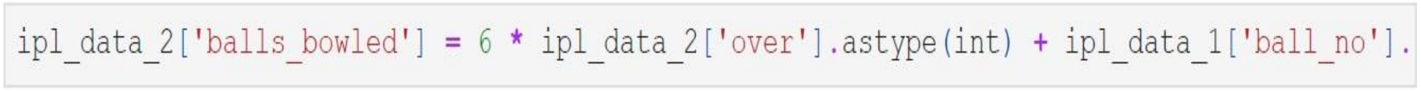
Adding new features to a dataset can enhance the capabilities of a model. These features provide the model with additional information and context, which can lead to improved accuracy and better predictions. When engineering new features, it is important to consider the correlation between the features and the target variable, as well as the interpretability of the features. It is also important to ensure that the new features are meaningful and relevant to the problem being solved. By carefully selecting and incorporating new features, the final dataset can be improved and optimized for modeling.

Ball_no: identifies which ball is being played in the current over.
balls_bowled: specifies the total number of balls that have been bowled in the firstinnings of the match.
balls_left: determines the number of balls remaining in the match, calculated by subtracting the number of balls bowled (balls_bowled) from 120.
wickets_left: indicates the number of wickets still available in the first innings, calculated by subtracting the current number of wickets (10 - current number of wickets).
current_score: shows the current score of the match.
current_runrate: reveals the current run rate of the match.
last_five: details the number of runs scored in the last 5 overs.
total_runs_x: display the final score of the match.

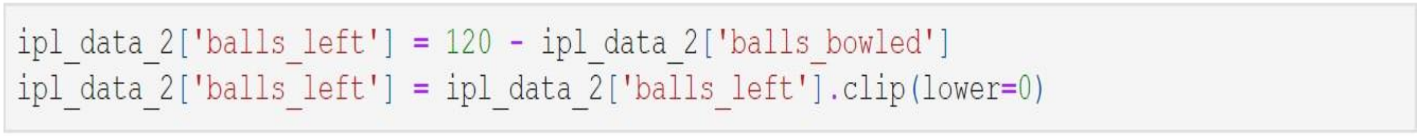
(1.24)



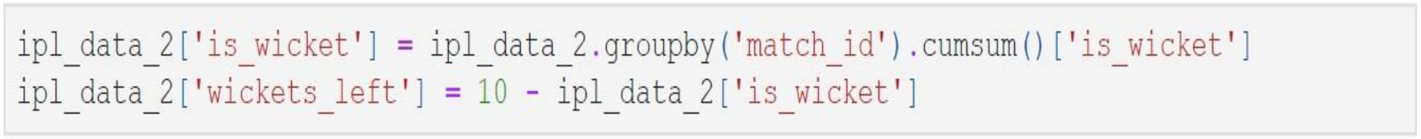
```
ipl_data_2['ball_no'] = ipl_data_2['ball'].apply(lambda x:str(x).split(".")[1])
```



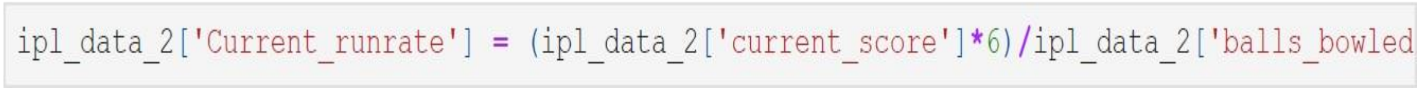
```
ipl_data_2['balls_bowled'] = 6 * ipl_data_2['over'].astype(int) + ipl_data_1['ball_no'].
```



```
ipl_data_2['balls_left'] = 120 - ipl_data_2['balls_bowled']
ipl_data_2['balls_left'] = ipl_data_2['balls_left'].clip(lower=0)
```



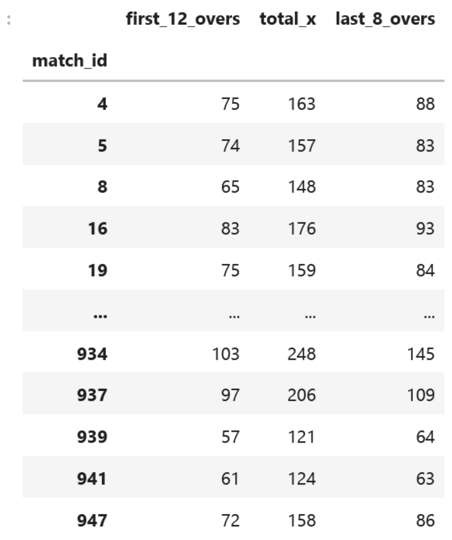
```
ipl_data_2['is_wicket'] = ipl_data_2.groupby('match_id').cumsum()['is_wicket']
ipl_data_2['wickets_left'] = 10 - ipl_data_2['is_wicket']
```



```
ipl_data_2['Current_runrate'] = (ipl_data_2['current_score']*6)/ipl_data_2['balls_bowled
```

Hypothesis Test

H(0):Runs scored in the first 12 overs of the first innings is equal to the runs scored in the last 8 overs H(1):Runs scored in the first 12 overs of the first innings is not equal to the runs scored in the last 8 overs

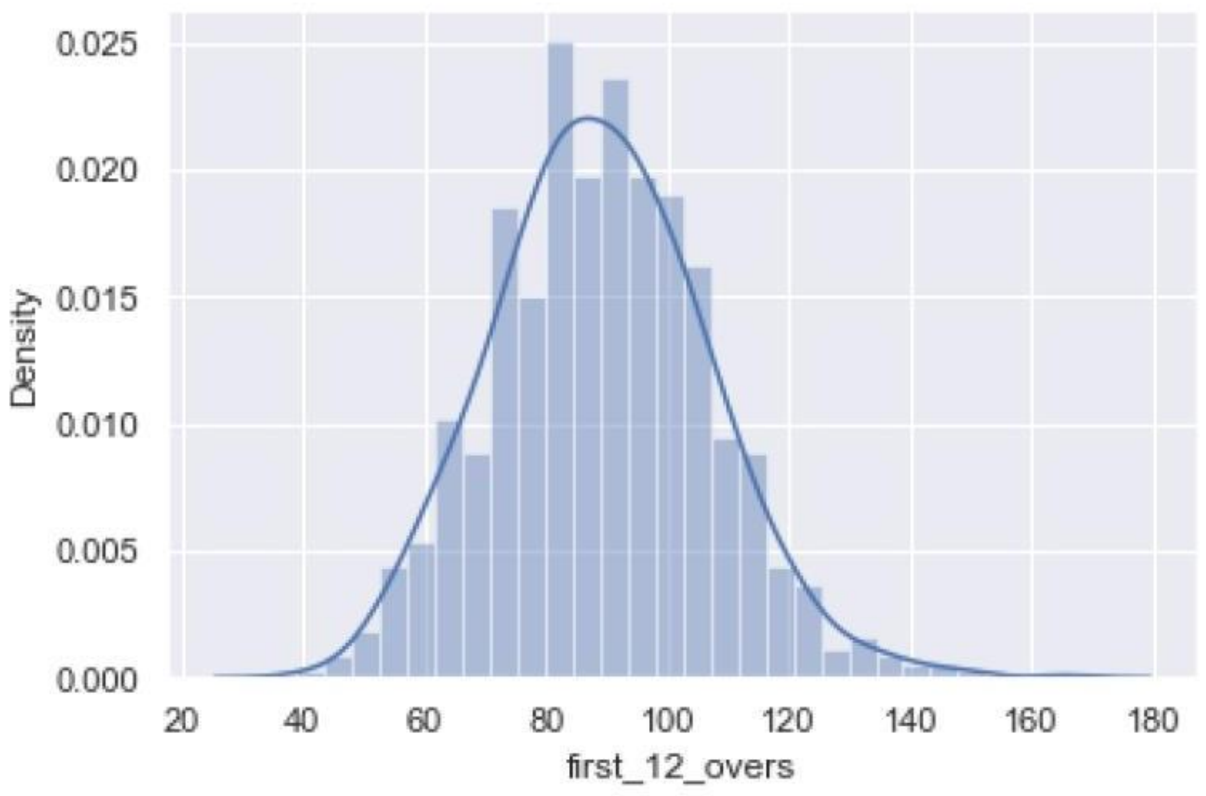


	first_12_overs	total_x	last_8_overs
match_id			
4	75	163	88
5	74	157	83
8	65	148	83
16	83	176	93
19	75	159	84
...	...	...	...
934	103	248	145
937	97	206	109
939	57	121	64
941	61	124	63
947	72	158	86

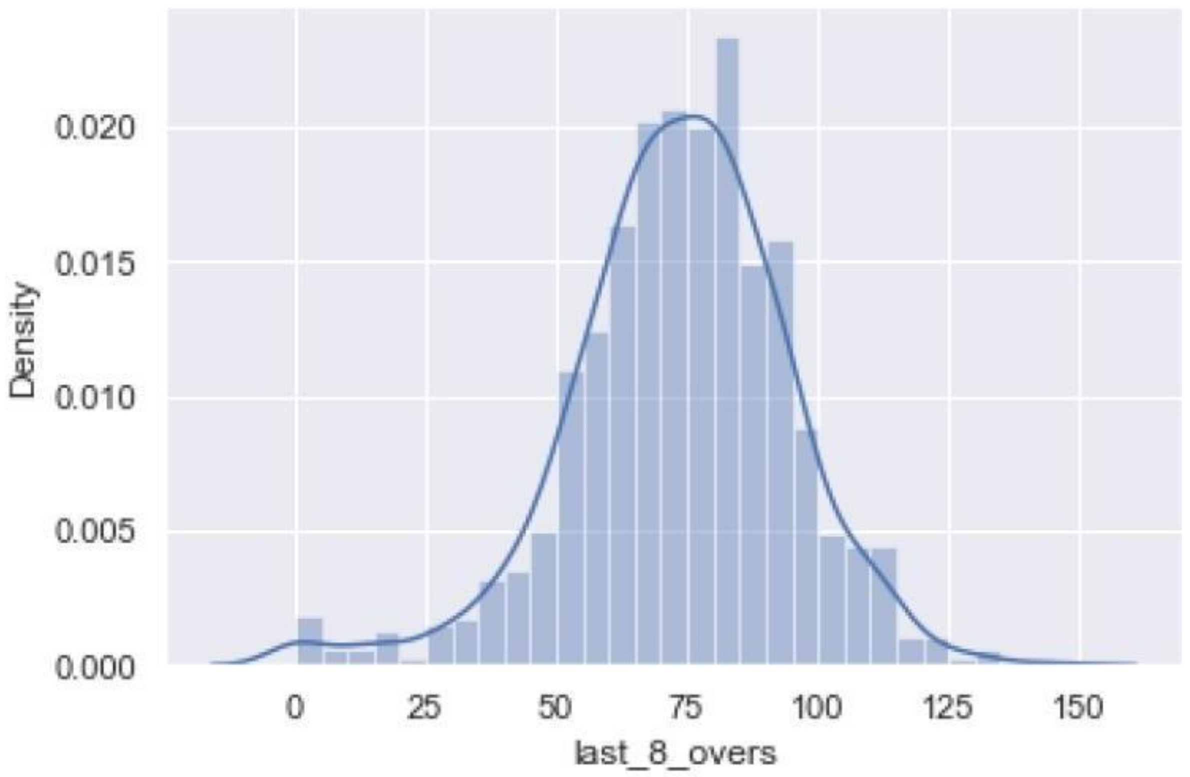
(1.25)

Normal distribution for first 12 over runs

(1.26)

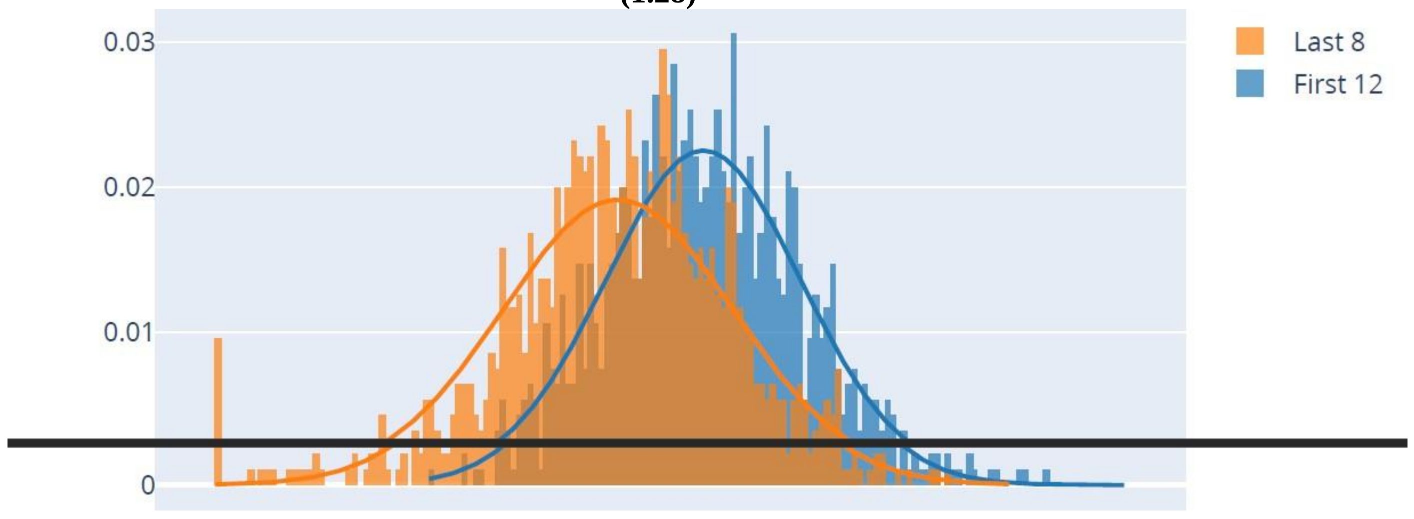


Normal distribution for last 8 over runs



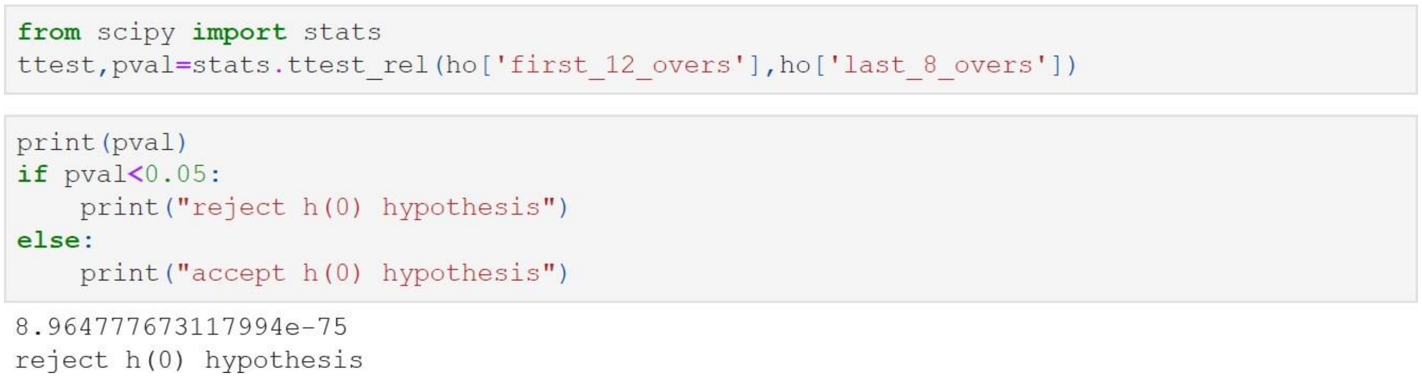
(1.27)

(1.28)



Both graphs show a normal bell-shaped curve, it can be concluded that the distribution of the data is normal. This type of distribution is recognizable by its symmetrical shape, with most data points centered around the mean and fewer on the tails.

T- Test :- A t-test is a type of inferential statistic which is used to determine if there is a significant difference between the means of two groups which may be related in certain features. We used here ttest_rel calculate to relation dataset.



```
from scipy import stats
ttest,pval=stats.ttest_rel(ho['first_12_overs'],ho['last_8_overs'])
```

```
print(pval)
if pval<0.05:
    print("reject h(0) hypothesis")
else:
    print("accept h(0) hypothesis")
```

```
8.964777673117994e-75
reject h(0) hypothesis
```

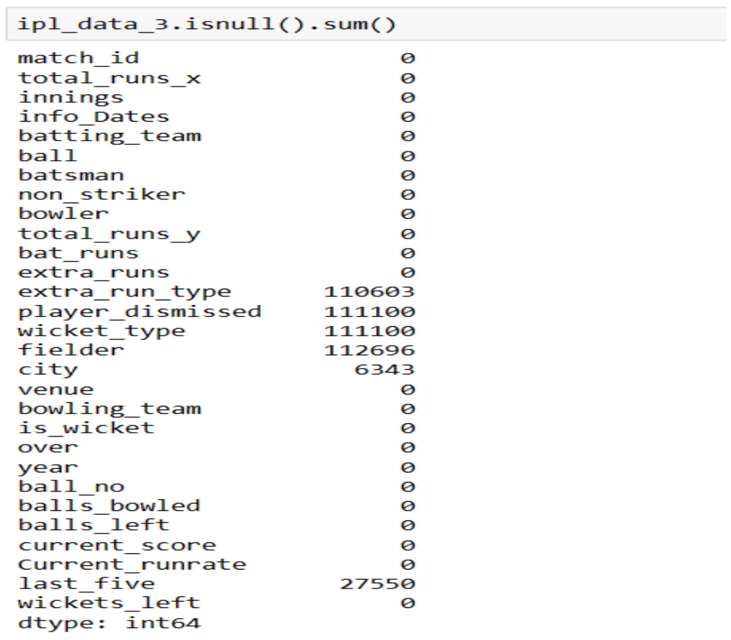
(1.29)

Based on the results of the statistical analysis, I reject the null hypothesis that the runs scored in the first 12 overs of the first innings are equal to the runs scored in the last 8 overs. The evidence supports the conclusion that there is a significant difference between the two sets of data

Reference: https://towardsdatascience.com/hypothesis-testing-in-machine-learning-using- python-a0dc89e169ce

WEEK 5

- In week 5, we checked for the null values in the data set. While that we found out that there is total 6343 value is null in city columns and we have total 27550 null value in last_five columns.

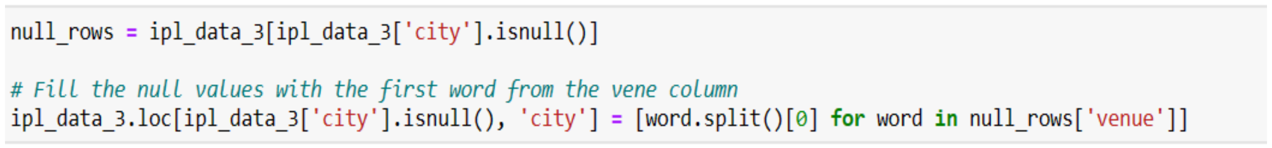


```
ipl_data_3.isnull().sum()
```

```
match_id                   0
total_runs_x               0
innings                    0
info_Dates                 0
batting_team               0
ball                       0
batsman                    0
non_striker                0
bowler                     0
total_runs_y               0
bat_runs                   0
extra_runs                 0
extra_run_type        110603
player_dismissed      111100
wicket_type           111100
fielder               112696
city                    6343
venue                      0
bowling_team               0
is_wicket                  0
over                       0
year                       0
ball_no                    0
balls_bowled               0
balls_left                 0
current_score              0
Current_runrate            0
last_five              27550
wickets_left               0
dtype: int64
```

(1.30)

- So, for that we input city name using venue name where first name is showing city name so we replace null city name from venue's first name.

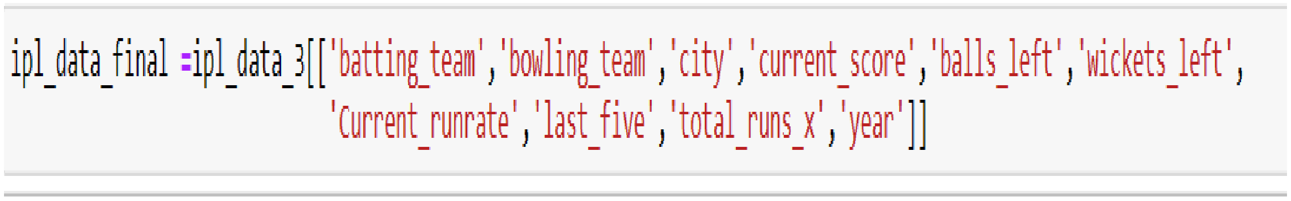


```
null_rows = ipl_data_3[ipl_data_3['city'].isnull()]

# Fill the null values with the first word from the vene column
ipl_data_3.loc[ipl_data_3['city'].isnull(), 'city'] = [word.split()[0] for word in null_rows['venue']]
```

(1.31)

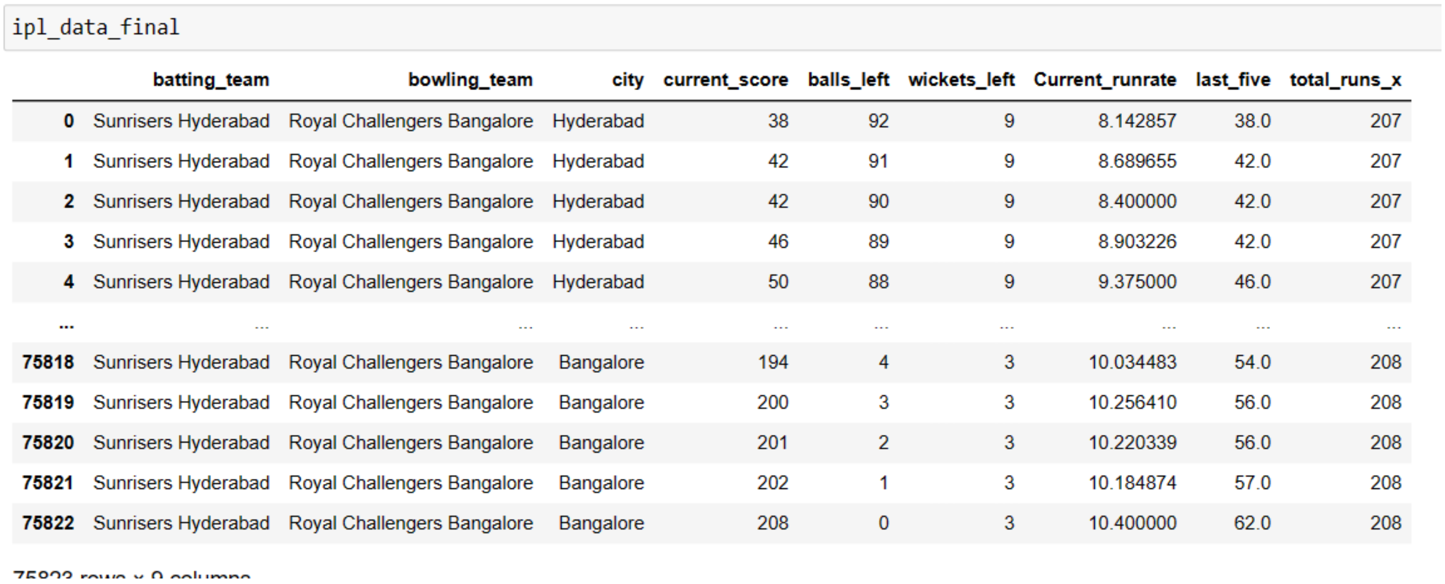
Selection of feature for the final dataset:

- So, we have different features in pur dataset such as, 'batting_team','bowling_team','city','current_score','balls_left','wickets_left','Current_runrate','last_five','total_runs_x.
- For, the final dataset slitting we select year feature.



```
ipl_data_final =ipl_data_3[['batting_team','bowling_team','city','current_score','balls_left','wickets_left',
                            'Current_runrate','last_five','total_runs_x','year']]
```

(1.32)



```
ipl_data_final
```

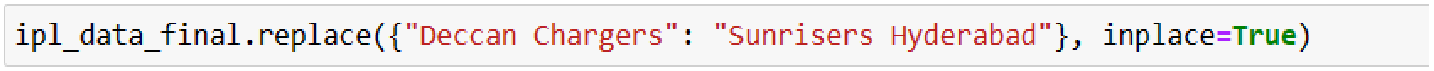
	batting_team	bowling_team	city	current_score	balls_left	wickets_left	Current_runrate	last_five	total_runs_x
0	Sunrisers Hyderabad	Royal Challengers Bangalore	Hyderabad	38	92	9	8.142857	38.0	207
1	Sunrisers Hyderabad	Royal Challengers Bangalore	Hyderabad	42	91	9	8.689655	42.0	207
2	Sunrisers Hyderabad	Royal Challengers Bangalore	Hyderabad	42	90	9	8.400000	42.0	207
3	Sunrisers Hyderabad	Royal Challengers Bangalore	Hyderabad	46	89	9	8.903226	42.0	207
4	Sunrisers Hyderabad	Royal Challengers Bangalore	Hyderabad	50	88	9	9.375000	46.0	207
...	...	...	...	...	...	...	...	...	...
75818	Sunrisers Hyderabad	Royal Challengers Bangalore	Bangalore	194	4	3	10.034483	54.0	208
75819	Sunrisers Hyderabad	Royal Challengers Bangalore	Bangalore	200	3	3	10.256410	56.0	208
75820	Sunrisers Hyderabad	Royal Challengers Bangalore	Bangalore	201	2	3	10.220339	56.0	208
75821	Sunrisers Hyderabad	Royal Challengers Bangalore	Bangalore	202	1	3	10.184874	57.0	208
75822	Sunrisers Hyderabad	Royal Challengers Bangalore	Bangalore	208	0	3	10.400000	62.0	208

75823 rows × 9 columns

(1.33)

- We do have 16 different teams in our dataset which are 'Sunrisers Hyderabad', ' Mumbai Indians', 'Gujarat Lions', 'Rising Pune Supergiants', 'Royal Challengers Bangalore', 'Kolkata Knight Riders', 'Delhi Capitals', 'Punjab Kings' , 'Rajasthan Royals', 'Chennai Super Kings', 'Lucknow Super Giants', 'Gujarat Titan-s', 'Deccan Chargers', 'Kochi Tuskers Kerala', 'Pune Warriors'

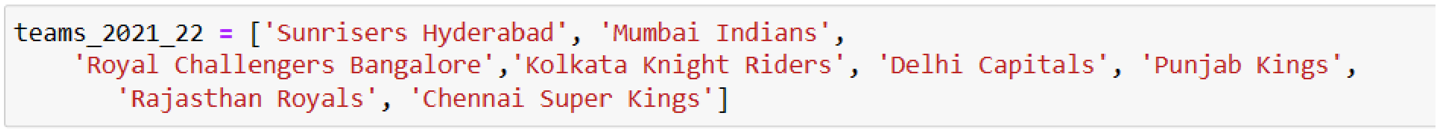
- We change the team's name of Deccan charger and give it a new name which is Sunrise Hyderabad.



```
ipl_data_final.replace({"Deccan Chargers": "Sunrisers Hyderabad"}, inplace=True)
```

(1.34)

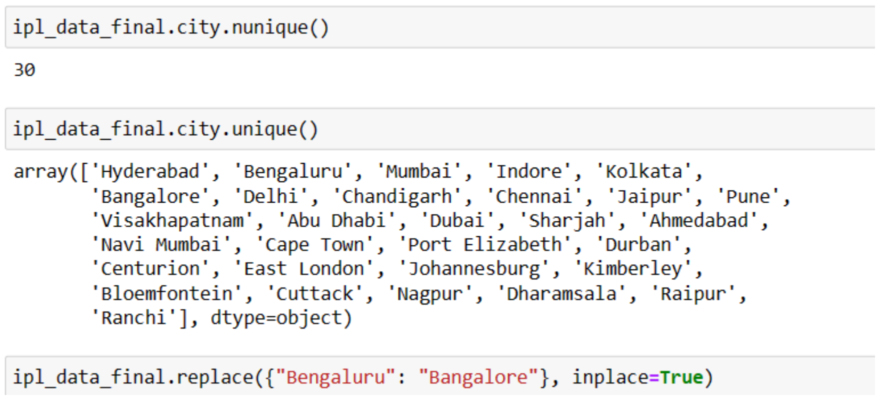
- For the model, we are going to choose only 8 teams.



```
teams_2021_22 = ['Sunrisers Hyderabad', 'Mumbai Indians',
    'Royal Challengers Bangalore','Kolkata Knight Riders', 'Delhi Capitals', 'Punjab Kings',
       'Rajasthan Royals', 'Chennai Super Kings']
```

(1.35)

- We do have total 30 unique cities of the India. From that we replace the name Bengaluru with Bangalore.



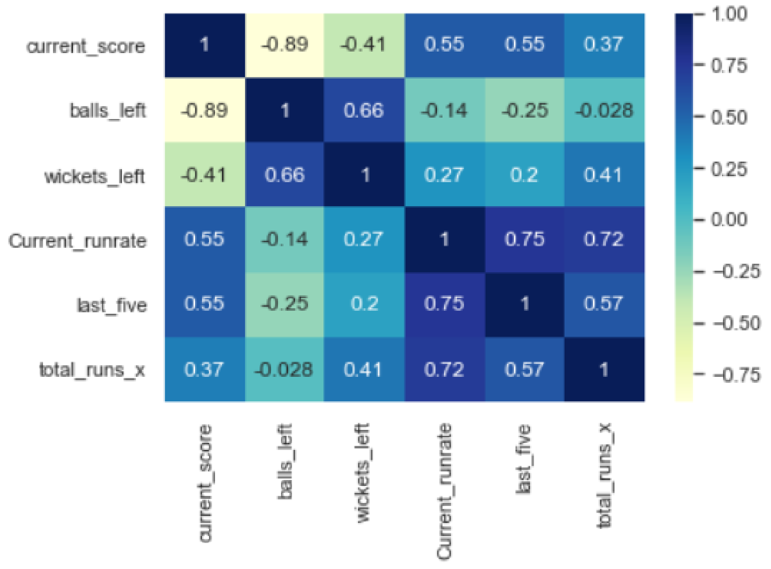
```
ipl_data_final.city.nunique()

30

ipl_data_final.city.unique()

array(['Hyderabad', 'Bengaluru', 'Mumbai', 'Indore', 'Kolkata',
       'Bangalore', 'Delhi', 'Chandigarh', 'Chennai', 'Jaipur', 'Pune',
       'Visakhapatnam', 'Abu Dhabi', 'Dubai', 'Sharjah', 'Ahmedabad',
       'Navi Mumbai', 'Cape Town', 'Port Elizabeth', 'Durban',
       'Centurion', 'East London', 'Johannesburg', 'Kimberley',
       'Bloemfontein', 'Cuttack', 'Nagpur', 'Dharamsala', 'Raipur',
       'Ranchi'], dtype=object)

ipl_data_final.replace({"Bengaluru": "Bangalore"}, inplace=True)
```

(1.36)

Heat map:

- We created a heat map of to show two-dimensional representation of data in which values are represented by colors which provides an immediate visual summary of information.

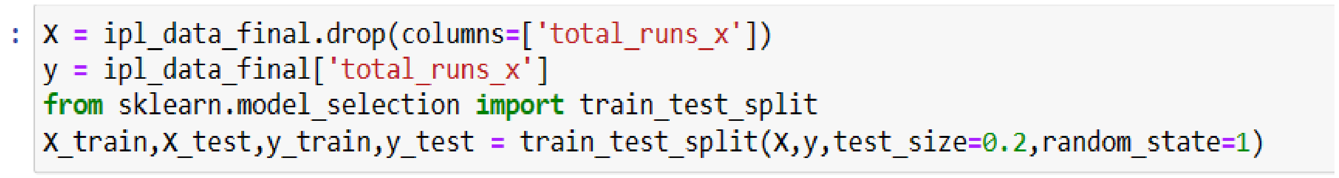
```
In [88]: from seaborn import heatmap
         sns.heatmap(ipl_data_final.corr(), annot=True, cmap="YlGnBu")

Out[88]: <AxesSubplot:>
```



(1.37)

Train and Test data-set:

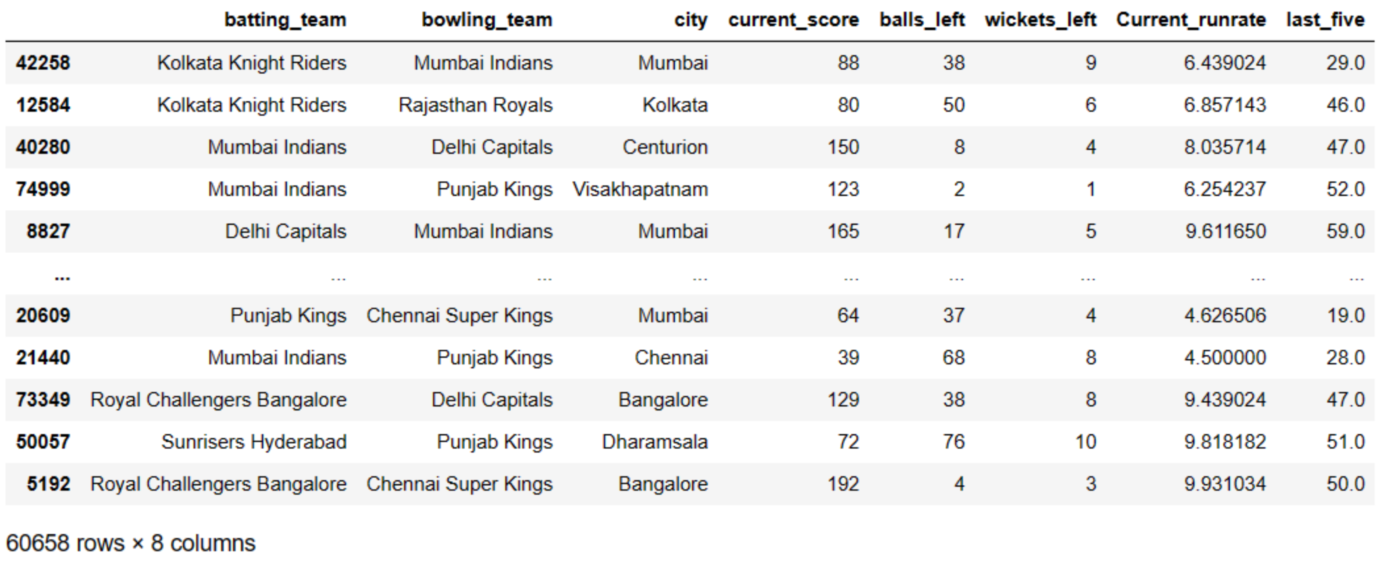
- For the model, we split the dataset in 80:20 ratio. In that, 80% is training dataset and 20% is test data set. Our target variable is total run x.



```
: X = ipl_data_final.drop(columns=['total_runs_x'])
  y = ipl_data_final['total_runs_x']
  from sklearn.model_selection import train_test_split
  X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=1)
```

(1.38)

- **X-train dataset:**

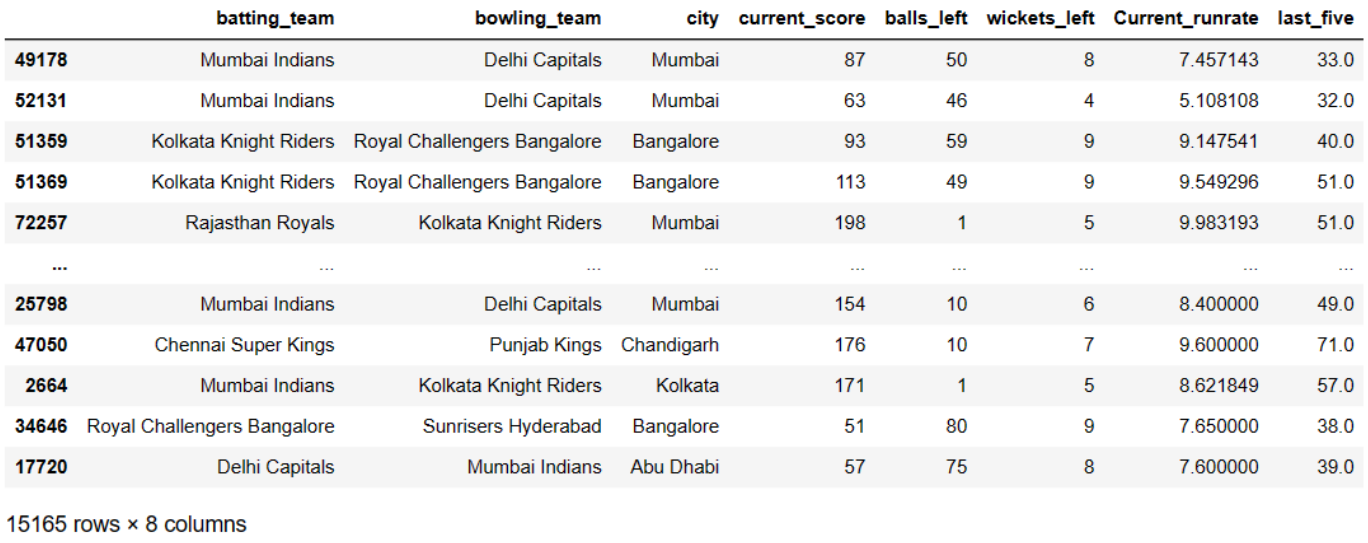


	batting_team	bowling_team	city	current_score	balls_left	wickets_left	Current_runrate	last_five
42258	Kolkata Knight Riders	Mumbai Indians	Mumbai	88	38	9	6.439024	29.0
12584	Kolkata Knight Riders	Rajasthan Royals	Kolkata	80	50	6	6.857143	46.0
40280	Mumbai Indians	Delhi Capitals	Centurion	150	8	4	8.035714	47.0
74999	Mumbai Indians	Punjab Kings	Visakhapatnam	123	2	1	6.254237	52.0
8827	Delhi Capitals	Mumbai Indians	Mumbai	165	17	5	9.611650	59.0
...	...	...	...	...	...	...	...	...
20609	Punjab Kings	Chennai Super Kings	Mumbai	64	37	4	4.626506	19.0
21440	Mumbai Indians	Punjab Kings	Chennai	39	68	8	4.500000	28.0
73349	Royal Challengers Bangalore	Delhi Capitals	Bangalore	129	38	8	9.439024	47.0
50057	Sunrisers Hyderabad	Punjab Kings	Dharamsala	72	76	10	9.818182	51.0
5192	Royal Challengers Bangalore	Chennai Super Kings	Bangalore	192	4	3	9.931034	50.0

60658 rows × 8 columns

(1.39)

- **X_Test Dataset:**

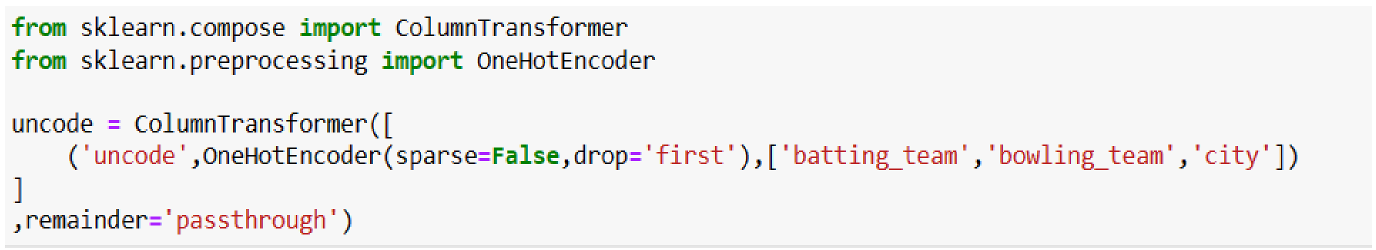


	batting_team	bowling_team	city	current_score	balls_left	wickets_left	Current_runrate	last_five
49178	Mumbai Indians	Delhi Capitals	Mumbai	87	50	8	7.457143	33.0
52131	Mumbai Indians	Delhi Capitals	Mumbai	63	46	4	5.108108	32.0
51359	Kolkata Knight Riders	Royal Challengers Bangalore	Bangalore	93	59	9	9.147541	40.0
51369	Kolkata Knight Riders	Royal Challengers Bangalore	Bangalore	113	49	9	9.549296	51.0
72257	Rajasthan Royals	Kolkata Knight Riders	Mumbai	198	1	5	9.983193	51.0
...	...	...	...	...	...	...	...	...
25798	Mumbai Indians	Delhi Capitals	Mumbai	154	10	6	8.400000	49.0
47050	Chennai Super Kings	Punjab Kings	Chandigarh	176	10	7	9.600000	71.0
2664	Mumbai Indians	Kolkata Knight Riders	Kolkata	171	1	5	8.621849	57.0
34646	Royal Challengers Bangalore	Sunrisers Hyderabad	Bangalore	51	80	9	7.650000	38.0
17720	Delhi Capitals	Mumbai Indians	Abu Dhabi	57	75	8	7.600000	39.0

15165 rows × 8 columns

(1.40)

- Then, we perform one hot-coding encoding on the column 'batting_team', 'bowling_team', and 'city'.



```python
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

uncode = ColumnTransformer([
    ('uncode',OneHotEncoder(sparse=False,drop='first'),['batting_team','bowling_team','city'])
]
,remainder='passthrough')
```

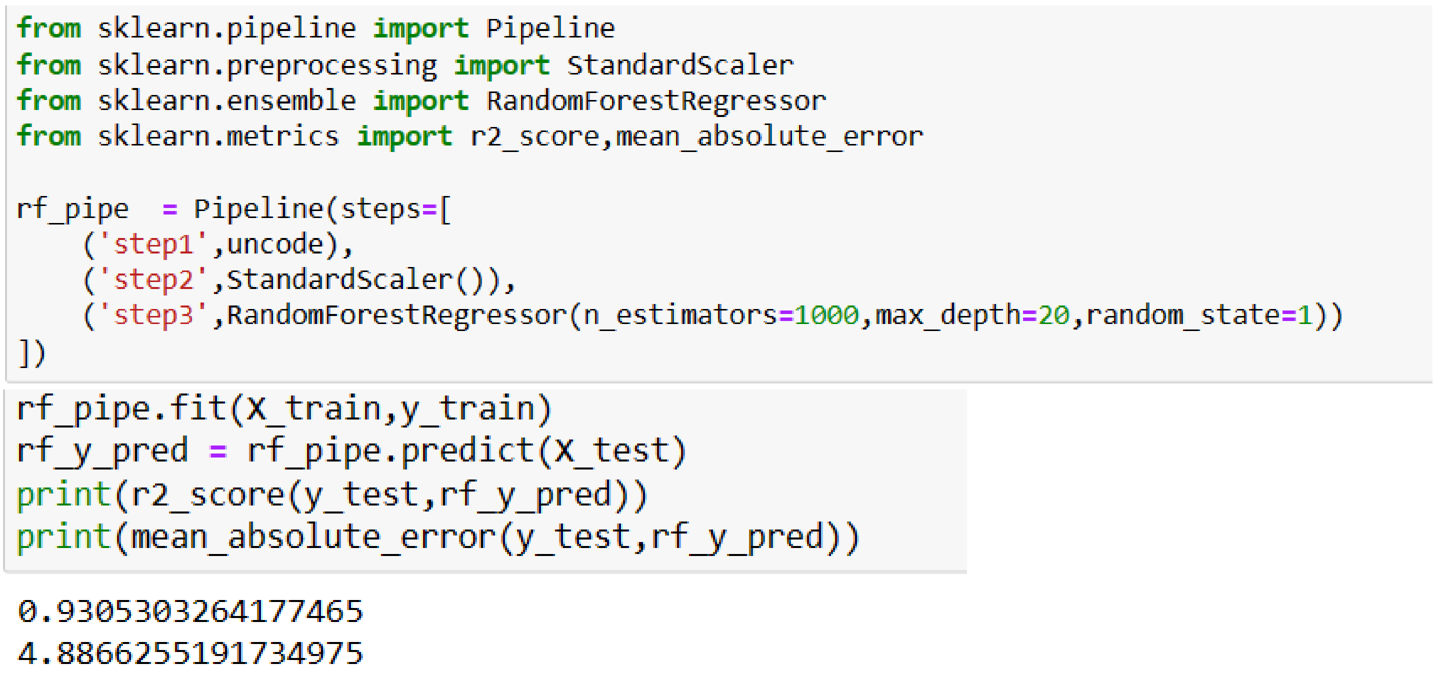
(1.41)

- This code is creating a ColumnTransformer object that will perform one-hot encoding on the columns 'batting_team', 'bowling_team', and 'city'.

- Basically, One-hot encoding is a method used to represent a categorical variable as a numerical one, by creating a new binary column for each unique category in the variable. In this representation, each column corresponds to a category and the value in the column is either 0 or 1, indicating the absence or presence of the category in the original data. The columns are often referred to as "dummy variables".

Model Selection:

• After the research, firstly we would like to use Random Forest algorithm because of its accuracy, and versatility. It can be used to solve a wide range of problems and is especially useful for handling large datasets and high-dimensional problems. So, we decided to use this particular model for our dataset to get the better and accurate answer of this problem.



```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score,mean_absolute_error

rf_pipe  = Pipeline(steps=[
    ('step1',uncode),
    ('step2',StandardScaler()),
    ('step3',RandomForestRegressor(n_estimators=1000,max_depth=20,random_state=1))
])
```

```
rf_pipe.fit(X_train,y_train)
rf_y_pred = rf_pipe.predict(X_test)
print(r2_score(y_test,rf_y_pred))
print(mean_absolute_error(y_test,rf_y_pred))
```

```
0.9305303264177465
4.886625519173497
```

(1.42)

- After applying this particular algorithm, we got 93% accuracy for the model for our problem statment.

What's Next….????

- Our next step is to check the other model with our dataset for the accuracy. For that , we are planning to take xgboost for our dataset to check where the accuracy is high or low for the given problem.
- Also, we are planning to create web interface to explain and demonstrate our project in better way

Group work contribution

Name	Activity
Mayur Donda	- worked on feature engineering, on hot encoding, test and train data with different splitting method and first model building.
Jeel Jiyani	- helped Mayur with the coding part. Also, going to present the half presentation on Monday.
Krupa Koladiya	- Gathered all the information to create the ppt and also did the whole ppt work.
Isha Jivani	- Collect information for the documentation Did the project documentation. Also, going to present the presentation on Monday.

References

- https://www.iplt20.com/
- https://cricsheet.org/
- https://pandas.pydata.org/
- https://numpy.org/
- https://www.tutorialspoint.com/python/python_nested_loops.htm
- https://datatofish.com/json-string-to-csv-python/
- https://zetcode.com/python/yaml/#:~:text=YAML%20
- https://seaborn.pydata.org/tutorial/introduction
- https://drmattcrooks.+medium.com/how-to-set-up-rcparams-in-matplotlib-355a0b9494ec
- https://drmattcrooks.medium.com/how-to-set-up-rcparams-in-matplotlib-355a0b9494ec
- https://www.programiz.com/python-programming/datetime