Exploratory Data Analysis on Sports

Sporting Future: A Comprehensive Analysis of Athletic Facilities

by

Group 9

Keerrtivardhan Goyal
ID: 202103007
Course:
BTech(MnC)

Yash Mashru
ID: 202103045
Course:
BTech(MnC)

Sanchit Satija
ID: 202103054
Course:
BTech(MnC)

Course Code: IT 462
Semester: Winter 2023

Under the guidance of

Dr. Gopinath Panda

Dhirubhai Ambani Institute of Information and Communication Technology

April 29, 2024

Acknowledgment

I am writing this letter to express my heartfelt gratitude for your guidance and support throughout my project titled “Sporting Future: A Comprehensive Analysis of Athletic Facilities” Your invaluable assistance has played a pivotal role in shaping the successful completion of this endeavor.

I am incredibly fortunate to have had the opportunity to work under your mentorship. Your expertise, encouragement, and willingness to share your knowledge have been instrumental in elevating the quality and scope of my project. Your constructive feedback and insightful suggestions have helped me overcome challenges and develop a deeper understanding of the subject matter.

Furthermore, I would like to thank the entire team at Dhirubhai Ambani Institute of Information and Communication Technology for fostering an environment of collaboration and innovation. The resources and facilities provided have been crucial in conducting comprehensive research and analysis.

I would also like to express my gratitude to my peers and colleagues who have been supportive throughout this journey. Their valuable input and camaraderie have been a constant source of motivation.

Completing this project has been a tremendous learning experience. I am confident that the knowledge and skills acquired during this endeavor will be a solid foundation for my future endeavors.

Sincerely,

Keerrtivardhan Goyal, 202103007
Yash Mashru, 202103045
Sanchit Satija, 202103054

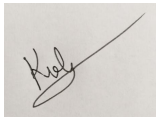
DECLARATION

We, [20203007, 202103045, 202103054] now declare that the EDA project work presented in this report is our original work and has not been submitted for any other academic degree. All the sources cited in this report have been appropriately referenced.

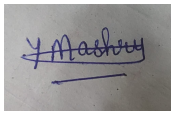
We acknowledge that the data utilized in this project has been sourced from Catalog.Data.gov. We affirm that we have complied with the terms and conditions specified on the website for accessing and using the dataset. We hereby confirm that the dataset employed in this project is accurate and authentic to the best of our knowledge.

We acknowledge that we have received no external help or assistance in conducting this project except for the guidance provided by our mentor, Prof. Gopinath Panda. We declare no conflict of interest in conducting this EDA project.

We have now signed the declaration statement and confirmed the submission of this report on 29 April 2024.



Keerrtivardhan Goyal
ID: 202103007
Course: BTech(Mnc)



Yash Mashru
ID: 202103045
Course: BTech(Mnc)

Sanchit Satija
ID: 202103054
Course: BTech(Mnc)

CERTIFICATE

This is to certify that Group 9 comprising Keerrtivrdhan Goyal, Yash Mashru and Sachit Satija has completed an exploratory data analysis (EDA) project on the Sporting Future: A Comprehensive Analysis of Athletic Facilities, which was obtained from Catalog.Data.gov site.

The EDA project presented by Group 9 is their original work. It was completed under the guidance of the course instructor, Prof. Gopinath Panda, who provided support and guidance throughout the project. The project is based on a thorough analysis of the Athletic_Facilities dataset, and the results presented in the report are based on the data obtained from the dataset.

This certificate is issued to recognize the successful completion of the EDA project on the Sporting Future: A Comprehensive Analysis of Athletic Facilities, which demonstrates the analytical skills and knowledge of the students of Group 9 in the field of data analysis.

Signed,
Dr. Gopinath Panda,
IT 462 Course Instructor
Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar, Gujarat, INDIA.

April 29, 2024

Contents

1 Introduction 1
- 1.1 Project idea and Defining the Problem . . . 1
- 1.2 Data Collection . . . 1
- 1.3 Dataset Description . . . 1
- 1.4 Packages required . . . 1

2 Data Cleaning 3
- 2.1 Missing data analysis . . . 7
  - 2.1.1 Missing Data Charts before Processing . . . 7
  - 2.1.2 Outliers Visualisation and Treatment . . . 11
  - 2.1.3 Missing Data Charts after Removal of Outliers . . . 17
  - 2.1.4 Type of Missingness found in our dataset . . . 25
- 2.2 Imputation . . . 26

3 Visualization 28
- 3.1 Univariate analysis . . . 28
  - 3.1.1 Pie Charts . . . 28
  - 3.1.2 Bar Charts . . . 32
  - 3.1.3 Histograms . . . 35
  - 3.1.4 Line Charts . . . 37
  - 3.1.5 WaterFall Charts . . . 39
  - 3.1.6 Violin Plots . . . 41
  - 3.1.7 KDE Plots . . . 47
- 3.2 Multivariate analysis . . . 53
  - 3.2.1 Scatter Plots . . . 53
  - 3.2.2 Stacked Bar Charts . . . 54
  - 3.2.3 Grouped Bar Charts . . . 56
  - 3.2.4 Tree Maps . . . 57
  - 3.2.5 Point Plots . . . 59
  - 3.2.6 Correlation Map . . . 62
  - 3.2.7 Pair Plot . . . 64
  - 3.2.8 Joint Plot . . . 66

4 Performing Some Basic Problems In Our Cleaned Dataset 68
- 4.1 1^{st} Problem : Which is the most popular sport ? . . . 68

4.2 2^{nd} Problem : What is the count of basketball facilities: Get the total count of entries where BASKETBALL is present ? . . . 68
4.3 3^{rd} Problem: What is the percentage of accessible facilities with basketball? . . . 69
4.4 4^{th} Problem: Calculate the density of athletic facilities per zipcode? . . . 69
4.5 5^{th} Problem: Find the sport with the maximum total area and the sport with the maximum total length among active sports, and display their respective total areas and lengths . . . 71

5 Feature Engineering **72**
5.1 Handling Categorical Variables . . . 72
5.2 Feature Scaling . . . 73
5.3 Feature selection . . . 77
5.3.1 Mutual Information Feature Selection Method: Target Variable: STArea . . . 78
5.3.2 Variance Threshold Feature Selection Method : Target Variable: STLength . . . 81

6 Model fitting and Evaluation **82**
6.1 Regression Problem . . . 84
6.1.1 1st Problem : Predicting the STArea of unseen data(test data) by the help of features selected from Mutual Information Feature Selection Technique. . . . 84
6.1.2 2nd Problem: Predicting the STLength of unseen data(test data) with the help of features selected from Variance Threshold Feature Selection Technique . . . 88
6.2 Classification Problem . . . 91
6.2.1 Predicting the column called BASKETBALL of unseen data(test data) which is of boolean datatype. . . . 91

7 Conclusion & future scope **94**
7.1 Findings/observations . . . 94
7.2 Challenges . . . 95
7.3 Future plan . . . 95

List of Figures

S.No.	Title of Figures	Pg.No.
1	Raw Data Matrix	8
2	Raw Data Dendrogram	8
3	Raw Data Bar Map	9
4	Raw Data Heat Map	10
5	Raw Data Box Plot for COMMUNITYBOARD	11
6	Raw Data Box Plot for COUNCILDISTRICT	12
7	Raw Data Box Plot for PRECINCT	13
8	Raw Data Box Plot for STArea	14
9	Raw Data Box Plot for STLength	15
10	Raw Data Box Plot for ZIPCODE	16
11	Treatment of Outliers	17
12	Cleaned Data Matrix	17
13	Cleaned Data Dendrogram	18
14	Cleaned Data Bar Map	18
15	Cleaned Data Heat Map	19
16	Cleaned Data Box Plot for COMMUNITYBOARD	20
17	Cleaned Data Box Plot for COUNCILDISTRICT	21
18	Cleaned Data Box Plot for PRECINCT	22
19	Cleaned Data Box Plot for STArea	23
20	Cleaned Data Box Plot for STLength	24
21	Cleaned Data Box Plot for ZIPCODE	25
22	ADULT_SOFTBALL Pie Chart	29
23	BOROUGH Pie Chart	29
24	FEATURESTATUS Pie Chart	30
25	LL_SOFTBALL Pie Chart	30
26	SURFACE_TYPE Pie Chart	31
27	VOLLEYBALL Pie Chart	31
28	BOCCE Bar Chart	32
29	DEPARTMENT Bar Chart	33
30	DIMENSIONS Bar Chart	33
31	GISPROPNUM Bar Chart	34
32	Picleball Bar Chart	34
33	PRIMARY_SPORT Bar Chart	35
34	FIELD_NUMBER Histogram Chart	36
35	PERCINCT Histogram	36
36	ZIPCODE Histogram	37
37	COMMUNITYBOARD Line Chart	38
38	STArea Line Chart	38
39	STLength Line Chart	39
40	DIMENSIONS Waterfall Chart	40
41	PRIMARY_SPORT Waterfall Chart	40
42	COMMUNITYBOARD Violin Plot	41
43	COUNCILDISTRICT Violin Plot	42
44	PRECINCT Violin Plot	43
45	STArea Violin Plot	44
46	STLength Violin Plot	45
47	ZIPCODE Violin Plot	46
48	COMMUNITYBOARD KDE Plot	47
49	COUNCILDISTRICT KDE Plot	48
50	PRECINCT KDE Plot	49

51	STArea KDE Plot	50
52	STLength KDE Plot	51
53	ZIPCODE KDE Plot	52
54	STLength and STArea Scatter Plot	54
55	DEPARTMENT by PRECINCT Stacked Bar Chart	55
56	DEPARTMENT by SURFACE_TYPE Stacked Bar Chart	55
57	BOROUGH BY PRIMARY_SPORT Grouped Bar Chart	56
58	SURFACE_TYPE BY PRIMARY_SPORT Grouped Bar Chart	57
59	BASKETBALL Tree Chart	58
60	HANDBALL Tree Chart	58
61	TENNIS Tree Chart	59
62	PRECINCT by PRIMARY_SPORT Point Plot	60
63	ZIPCODE by PRIMARY_SPORT Point Plot	61
64	Numerical Column Correlation Map	63
65	Numerical Column Pair Plot	65
66	Numerical Column Joint Plot	67

Abstract

Urban planning is essential for the development of cities and communities to ensure that the residents have access to all the required facilities. Some of the facilities that contribute to the well-being of the community are the athletic facilities, which include the courts, basketball courts, tennis courts, and soccer fields, among others.

A better understanding of the distribution and the nature of the various fields is essential for proper development and planning. The project is going to analyze the dataset of the athletic facilities . This is to explore a better knowledge of how the athletic facilities in the dataset are distributed. The project is structured to perform an Exploratory Data Analysis of the dataset obtained from data.gov.in.

The primary objective is to analyse distribution and characteristics of athletic facilities and make predictions using Machine learning models for better planning and development of these athletic facilities.

The report begins with defining the problem and Data collection , followed by data preprocessing steps, Visualizations, Data Transformation, Feature Engineering, Statistical analysis, Model Development and its Evaluation.

The main goal behind conducting Exploratory Data Analysis is to enable the drawing of patterns, trends, relationships, and other cognitive evidence that could be useful for local authorities, policymakers, and urban planners in their efforts to make more informed decisions about athletic facility development and maintenance, which in turn will help improve communities and the quality of life for residents.

Chapter 1. Introduction

1.1 Project idea and Defining the Problem

Accurate predictions of the Surface Area and Surface Length of sports facilities are paramount in making resource allocation and cost-effective selections during urban planning and infrastructure Establishment. These key indicators are critical predictors of a facility's magnitude and an essential aspect of space planning. They affect the budget and land use strategy. This gives a compelling explanation of the optimal predictive models of STArea and STLength based on historical data on sports facilities. The inquiry may be beneficial in rational decision-making and responsible resource management, thereby improving the recreational infrastructure quality and, as a result, the overall well-being and physical activity of the community.

1.2 Data Collection

We have collected our dataset 'Athletic_Facilities' from the website Catalog.Data.gov which is an official data collection website of the government. The dataset was downloaded in CSV format.

1.3 Dataset Description

The dataset 'Athletic_Facilities' describes the distribution of various sports facilities around different districts. It distinguishes the facilities based on factors such as the sport the facility is made for (BASKETBALL, VOLLEYBALL, LL_SOFTBALL, TENNIS, etc), its material (SURFACE_TYPE), the department and jurisdiction it belongs to (BOROUGH, COMMUNITYBOARD, COUNCILDISTRICT, DEPARTMENT, GISPROPNUM, ZIPCODE), its various dimensions and measurements (PRECINCT, STArea, STLength), identification details (PRIMARY_SPORT, SYSTEM, FIELD_NUMBER) and accessibility (ACCESSIBILITY, FEATURESTATUS, FIELD_LIGHTED, MAINTENANCEAGREEMENT).

Most of the columns are there to imply the sport the facility was made for, and a portion is specifically dedicated to detailing its jurisdictions.

1.4 Packages required

We require many different ***python*** libraries for our dataset cleaning and visualization. They are :

- NumPy Library

- Pandas Library
- PyPlot from MatPlotLib Library
- MissingNo Library
- SeaBorn Library
- KNNImputer from SKLearn Library

Visualization Tool -

We have also made use of ***Microsoft PowerBI*** software as a medium for data visualization, utilizing its functions to demonstrate some visualization techniques.

Chapter 2. Data Cleaning

After data collection, we begin our EDA by importing the required libraries and loading the csv file into dataframe using:

df=pd.read_csv('Athletic_Facilities_Test.csv')

Our Dataset *'Athletic_Facilities_Test.csv'* has 6,897 rows and 44 columns. We have used the df.shape attribute of pandas to see the shape of our dataset.

To have a quick view into our Athletic Facilities dataset, we can use the sample() function to sample 10 random rows. The following represents a random sample of 10 records in our dataset. These 10 random rows give a view of the type of data contained in this dataset.

The df.info() method is very crucial as it provides an overview of the dataset. It includes critical information regarding the columns of the dataset such as the column name, number of non-null values stored in the column as well as data type of each column.
The data types of the columns are:

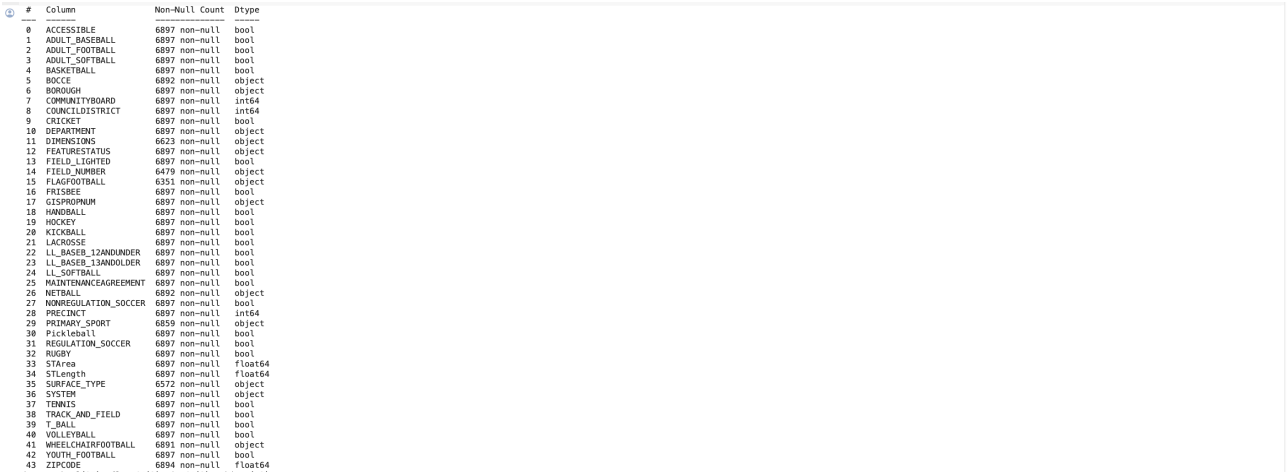
- Bool (Boolean) : Indicates True or False values.
- int64 : Represents integer values.
- object : Represents string values.
- float64 : Represents floating point values.

Pandas has an attribute df.dtypes that provides information about the data types of the columns. It is a very important method since the data types are required for data preprocessing and analysis of our dataset.

ACCESSIBLE	ADULT_BASEBALL	ADULT_FOOTBALL	ADULT_SOFTBALL	BASKETBALL	BOCCE	BOROUGH	COMMUNITYBOARD	COUNCILDISTRICT	CRICKET	...	STLength	SURFACE_TYPE	SYSTEM	TENNIS	TRACK_AND_FIELD	T_BALL	VOLLEYBALL	WHEELCHAIRFOOTBALL	YOUTH_FOOTBALL	ZIPCODE
False	False	False	False	False	False	M	102	3	False	...	124.577623	Concrete	M038-PICKLEBALL-1	False	False	False	False	False	False	10014.0
False	False	False	False	True	False	Q	409	28	False	...	247.347881	Asphalt	Q337-BASKETBALL-3	False	False	False	False	False	False	11416.0
False	False	False	True	False	False	B	308	35	False	...	373.020314	Synthetic	B232-BASEBALL-1	False	False	True	False	False	False	11217.0
False	False	False	True	False	False	X	203	18	False	...	626.958756	Synthetic	X147A-SOFTBALL-2	False	False	True	False	False	False	10460.0
False	False	False	False	False	False	B	313	47	False	...	233.990841	NaN	B337-TENNIS-1	False	False	False	False	False	False	11224.0
False	False	False	False	False	False	Q	408	23	False	...	238.634474	Asphalt	Q021-ZN02-TENNIS-15	True	False	False	False	False	False	11423.0
False	False	False	False	False	False	R	501	49	False	...	248.356241	Asphalt	R061-BASKETBALL-4	False	False	False	False	False	False	10304.0
False	False	False	False	False	False	B	317	45	False	...	301.946227	Asphalt	B285-SOCCER-1	False	False	False	False	False	False	11203.0
False	False	False	True	False	False	R	502	50	False	...	652.102469	Natural	R065-ZN08-BASEBALL-1	False	False	False	False	False	False	10314.0
False	False	False	False	False	False	X	226	11	False	...	600.613290	Natural	X092-ZN11-FOOTBALL-1	False	False	False	False	False	True	10470.0

× 44 columns

Figure 2.1: df.sample



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6897 entries, 0 to 6896
Data columns (total 44 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ACCESSIBLE            6897 non-null   bool
 1   ADULT_BASEBALL        6897 non-null   bool
 2   ADULT_FOOTBALL        6897 non-null   bool
 3   ADULT_SOFTBALL        6897 non-null   bool
 4   BASKETBALL            6897 non-null   bool
 5   BOCCE                 6892 non-null   object
 6   BOROUGH               6897 non-null   object
 7   COMMUNITYBOARD        6897 non-null   int64
 8   COUNCILDISTRICT       6897 non-null   int64
 9   CRICKET               6897 non-null   bool
 10  DEPARTMENT            6897 non-null   object
 11  DIMENSIONS            6623 non-null   object
 12  FEATURESTATUS         6897 non-null   object
 13  FIELD_LIGHTED         6897 non-null   bool
 14  FIELD_NUMBER          6479 non-null   object
 15  FLAGFOOTBALL          6351 non-null   object
 16  FRISBEE               6897 non-null   bool
 17  GISPROPNUM            6897 non-null   object
 18  HANDBALL              6897 non-null   bool
 19  HOCKEY                6897 non-null   bool
 20  KICKBALL              6897 non-null   bool
 21  LACROSSE              6897 non-null   bool
 22  LL_BASEB_12ANDUNDER   6897 non-null   bool
 23  LL_BASEB_13ANDOLDER   6897 non-null   bool
 24  LL_SOFTBALL           6897 non-null   bool
 25  MAINTENANCEAGREEMENT  6897 non-null   bool
 26  NETBALL               6892 non-null   object
 27  NONREGULATION_SOCCER  6897 non-null   bool
 28  PRECINCT              6897 non-null   int64
 29  PRIMARY_SPORT         6859 non-null   object
 30  Pickleball            6897 non-null   bool
 31  REGULATION_SOCCER     6897 non-null   bool
 32  RUGBY                 6897 non-null   bool
 33  STArea                6897 non-null   float64
 34  STLength              6897 non-null   float64
 35  SURFACE_TYPE          6572 non-null   object
 36  SYSTEM                6897 non-null   object
 37  TENNIS                6897 non-null   bool
 38  TRACK_AND_FIELD       6897 non-null   bool
 39  T_BALL                6897 non-null   bool
 40  VOLLEYBALL            6897 non-null   bool
 41  WHEELCHAIRFOOTBALL    6891 non-null   object
 42  YOUTH_FOOTBALL        6897 non-null   bool
 43  ZIPCODE               6894 non-null   float64
```

Figure 2.2: df.info

Here's the data type of each column:

- Boolean (True/False) Data Types:

  1. ACCESSIBLE
  2. ADULT_BASEBALL
  3. ADULT_FOOTBALL
  4. ADULT_SOFTBALL
  5. BASKETBALL
  6. CRICKET
  7. FRISBEE
  8. FIELD_LIGHTED
  9. HANDBALL
  10. HOCKEY
  11. KICKBALL
  12. LACROSSE
  13. LL_BASEB_12ANDUNDER
  14. LL_BASEB_13ANDOLDER
  15. LL_SOFTBALL
  16. MAINTENANCEAGREEMENT
  17. FLAGFOOTBALL
  18. Pickleball
  19. NONREGULATION_SOCCER
  20. REGULATION_SOCCER
  21. RUGBY

22. TENNIS
23. TRACK_AND_FIELD
24. T_BALL
25. VOLLEYBALL
26. YOUTH_FOOTBALL

- Integer Data Types:

1. COMMUNITYBOARD
2. COUNCILDISTRICT
3. PRECINCT

- Float Data Types:

1. STArea
2. STLength
3. ZIPCODE

- Object (String) Data Types:

1. BOCCE
2. BOROUGH
3. DEPARTMENT
4. DIMENSIONS
5. FEATURESTATUS
6. FIELD_NUMBER
7. GISPROPNUM
8. NETBALL
9. PRIMARY_SPORT
10. SURFACE_TYPE
11. SYSTEM
12. WHEELCHAIRFOOTBALL

The df.nunique gets the count of unique values in each column of the dataset. It is important to understand the count of unique values in each column as this helps to have an overview of how variables are varied and data is distributed.

The df.describe() gets descriptive statistics for the numeric columns in the dataset. The descriptive statistics are used to show the central tendency (mean, median or mode), dispersion (variance, standard deviation, minimum, and maximum), and the shape of the distribution of numeric columns in the dataset.

```
df.nunique()

ACCESSIBLE                2
ADULT_BASEBALL            2
ADULT_FOOTBALL            2
ADULT_SOFTBALL            2
BASKETBALL                2
BOCCE                     2
BOROUGH                   5
COMMUNITYBOARD           66
COUNCILDISTRICT          55
CRICKET                   2
DEPARTMENT               72
DIMENSIONS               19
FEATURESTATUS             5
FIELD_LIGHTED             2
FIELD_NUMBER            122
FLAGFOOTBALL              2
FRISBEE                   2
GISPROPNUM              801
HANDBALL                  2
HOCKEY                    2
KICKBALL                  2
LACROSSE                  2
LL_BASEB_12ANDUNDER       2
LL_BASEB_13ANDOLDER       2
LL_SOFTBALL               2
MAINTENANCEAGREEMENT      2
NETBALL                   2
NONREGULATION_SOCCER      2
PRECINCT                 78
PRIMARY_SPORT            16
Pickleball                2
REGULATION_SOCCER         2
RUGBY                     2
STArea                 6783
STLength               6642
SURFACE_TYPE             10
SYSTEM                 6897
TENNIS                    2
TRACK_AND_FIELD           2
T_BALL                    2
VOLLEYBALL                2
WHEELCHAIRFOOTBALL        2
YOUTH_FOOTBALL            2
ZIPCODE                 166
dtype: int64
```

Figure 2.3: df.nunique

```
df.describe()
```

	COMMUNITYBOARD	COUNCILDISTRICT	PRECINCT	STArea	STLength	ZIPCODE
count	6897.000000	6897.000000	6897.000000	6897.000000	6897.000000	6894.000000
mean	301.545455	29.544730	72.651443	9691.495987	315.868768	10915.043226
std	114.419828	91.871972	32.855971	17950.584068	303.677573	532.520876
min	101.000000	1.000000	1.000000	203.048508	58.054472	10001.000000
25%	208.000000	15.000000	45.000000	1289.053223	147.109697	10457.000000
50%	310.000000	27.000000	73.000000	2650.568771	216.744262	11211.000000
75%	407.000000	38.000000	105.000000	4259.432831	271.658003	11360.000000
max	595.000000	3730.000000	123.000000	200995.310600	3250.215801	11694.000000

Figure 2.4: df.describe

2.1 Missing data analysis

Missing data is a frequent problem with datasets, and it has a substantial impact on the quality of our analysis. Prior to any data processing, it is critical to identify the amount and proportion of empty data present in the dataset. The following missing data analysis complements our Athletic Facilities dataset.

So, we will move to the missing value analysis or we can say the data cleaning part or data preprocessing part.

Firstly, we need to know how many missing values are in every column. It can be done using *df.isnull().sum()* In fact, the *df.isnull().sum()* function calculates the number of missing values in each column of our dataset. It returns a series where each column name is a key and the value is correlated to the number of missing values in this column.

For example, we have a column FLAGFOOTBALL which has 546 missing value and it is maximum in missing values . If we have missing values then the following can see in the output of *df.isnull().sum()* that is “FLAGFOOTBALL: 546”.

The number of missing values in our dataset is. FLAG FOOTBALL: 546, FIELD NUMBER: 418, DIMENSIONS: 274, SURFACE TYPE: 325, PRIMARY SPORT: 38, WHEELCHAIRFOOTBALL: 6, NETBALL: 5, BOCCE: 5, ZIPCODE: 3 and all other columns have no missing values.

```
ACCESSIBLE                0
ADULT_BASEBALL            0
ADULT_FOOTBALL            0
ADULT_SOFTBALL            0
BASKETBALL                0
BOCCE                     5
BOROUGH                   0
COMMUNITYBOARD            0
COUNCILDISTRICT           0
CRICKET                   0
DEPARTMENT                0
DIMENSIONS              274
FEATURESTATUS             0
FIELD_LIGHTED             0
FIELD_NUMBER            418
FLAGFOOTBALL            546
FRISBEE                   0
GISPROPNUM                0
HANDBALL                  0
HOCKEY                    0
KICKBALL                  0
LACROSSE                  0
LL_BASEB_12ANDUNDER       0
LL_BASEB_13AND0LDER       0
LL_SOFTBALL               0
MAINTENANCEAGREEMENT      0
NETBALL                   5
NONREGULATION_SOCCER      0
PRECINCT                  0
PRIMARY_SPORT            38
Pickleball                0
REGULATION_SOCCER         0
RUGBY                     0
STArea                    0
STLength                  0
SURFACE_TYPE            325
SYSTEM                    0
TENNIS                    0
TRACK_AND_FIELD           0
T_BALL                    0
VOLLEYBALL                0
WHEELCHAIRFOOTBALL        6
YOUTH_FOOTBALL            0
ZIPCODE                   3
dtype: int64
```

Figure 2.5: df.isnull().sum()

2.1.1 Missing Data Charts before Processing

Dendogram, Bar chart, Matrix plot have been used to analyse the missing data. These visualization will assist us to identify the patterns and the missing data in the dataset so we can make right decisions about the relative imputation approaches.

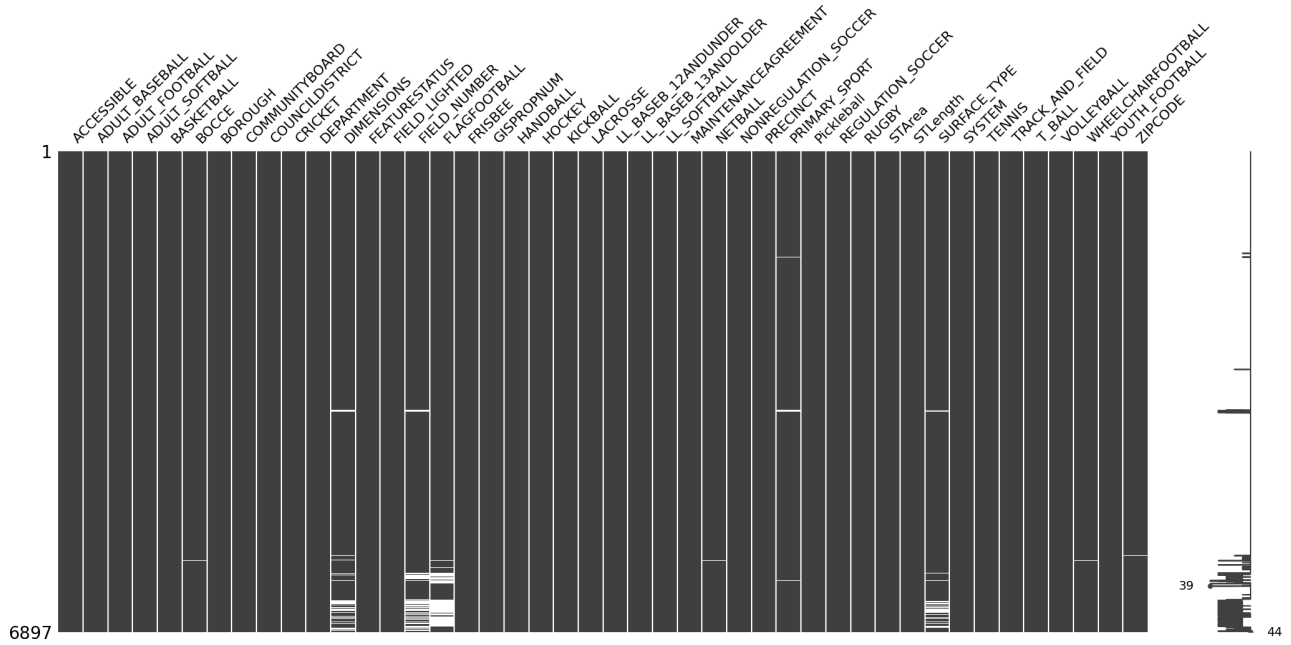


Figure 2.6: The above figure depicts the Matrix plot of the Raw Data. The blank nodes in the Matrix represent the missing data.

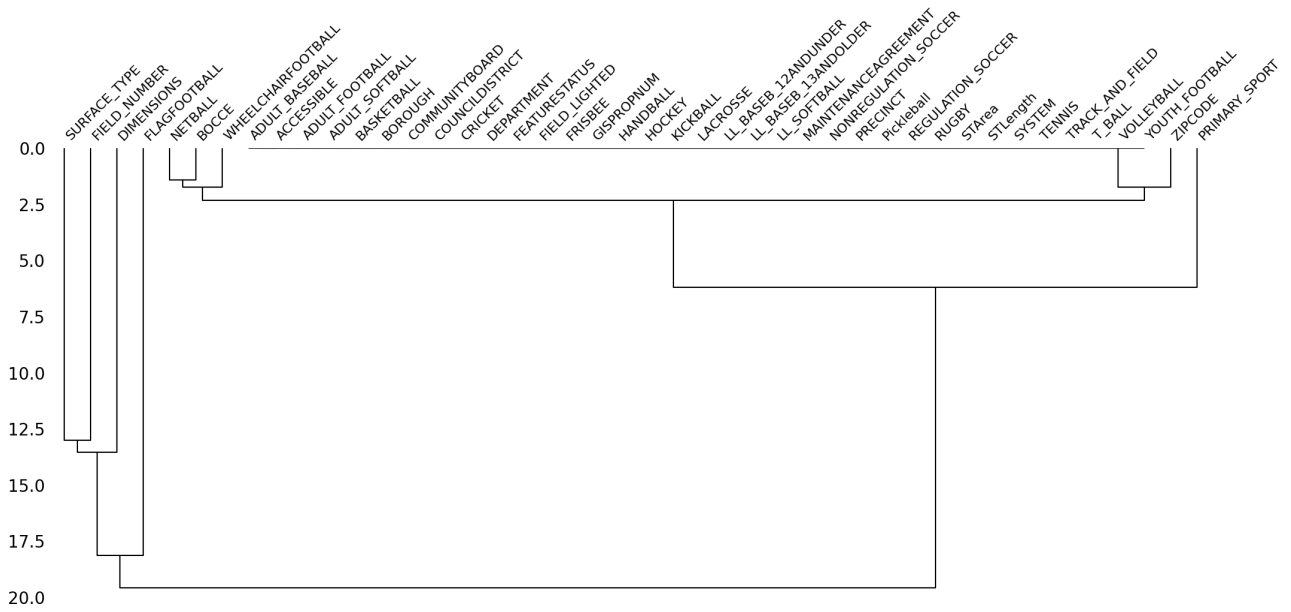


Figure 2.7: The above figure shows the Dendrogram plot of Raw Data. Dendrogram is a form of tree that shows the closeness/similarity of the columns. The closer two columns are placed in the Dendrogram, the more they are similar to each other.

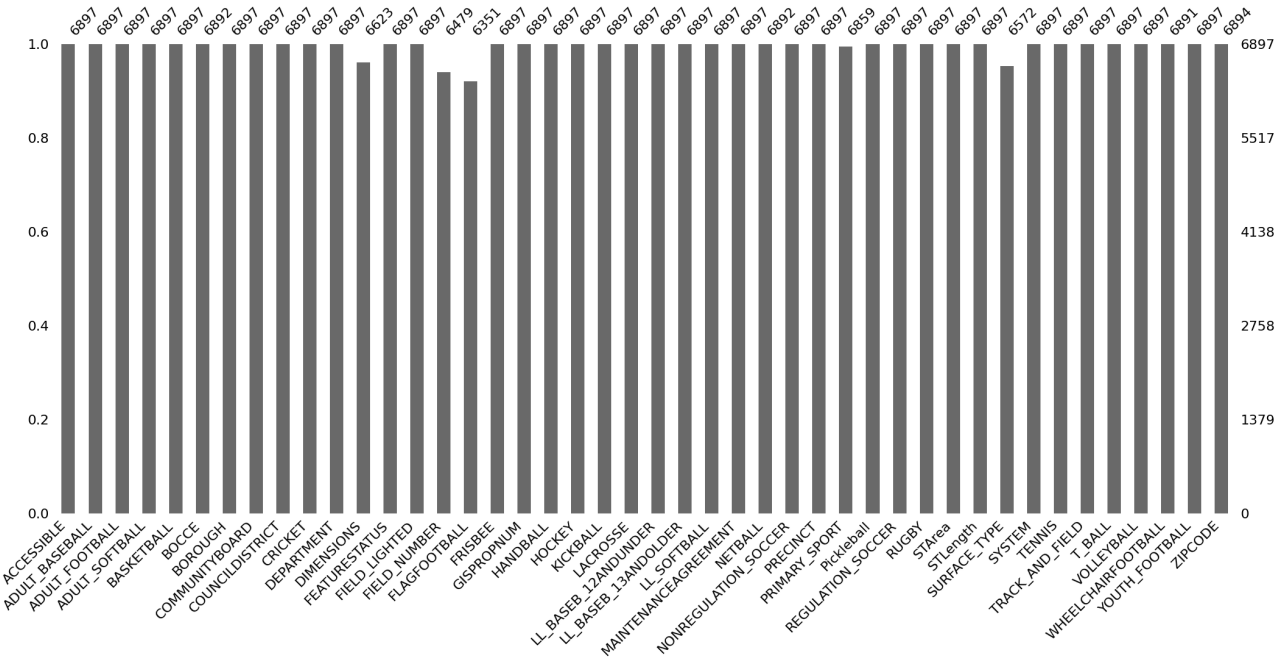


Figure 2.8: The above figure depicts the Bar chart of the Raw Data. The short bars in the Bar chart represent the missing data.

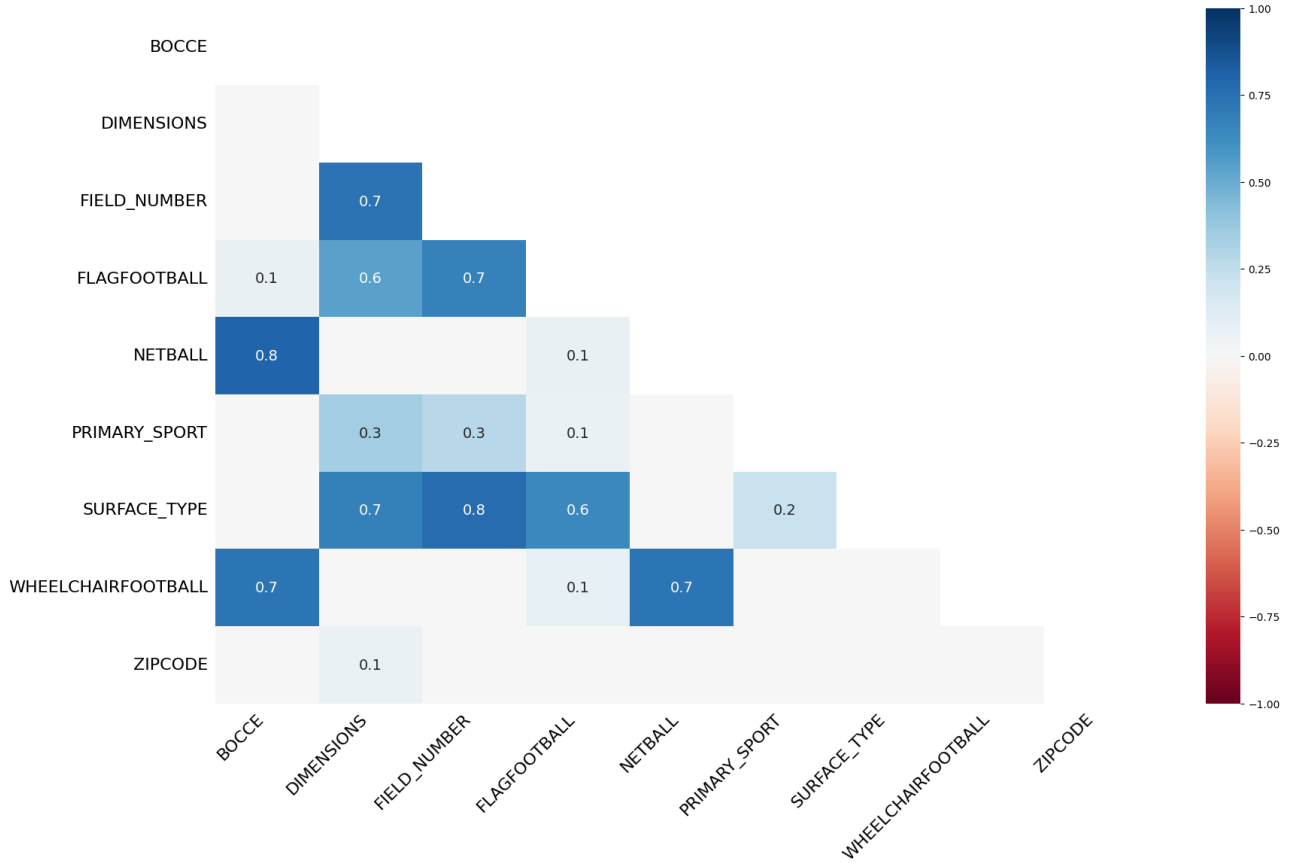


Figure 2.9: The above figure depicts the Heat Map of the Raw Data. This heatmap gives a clear representation of how much the missing value in one feature predicts the absence of missing values in another feature. Darker squares show a higher correlation in missingness between the features. Lighter squares show a lower correlation in missingness.

2.1.2 Outliers Visualisation and Treatment

To find outliers in the dataset, we have plotted box plots. Here are the vitals of the box plots we obtained:
Box plots represent the minimum, maximum, 25th percentile, median and the 75the percentile. The whiskers of the plot represent minimum and maximum values. Box plots give us a better understanding of how each numerical feature is distributed and identify potential outliers.

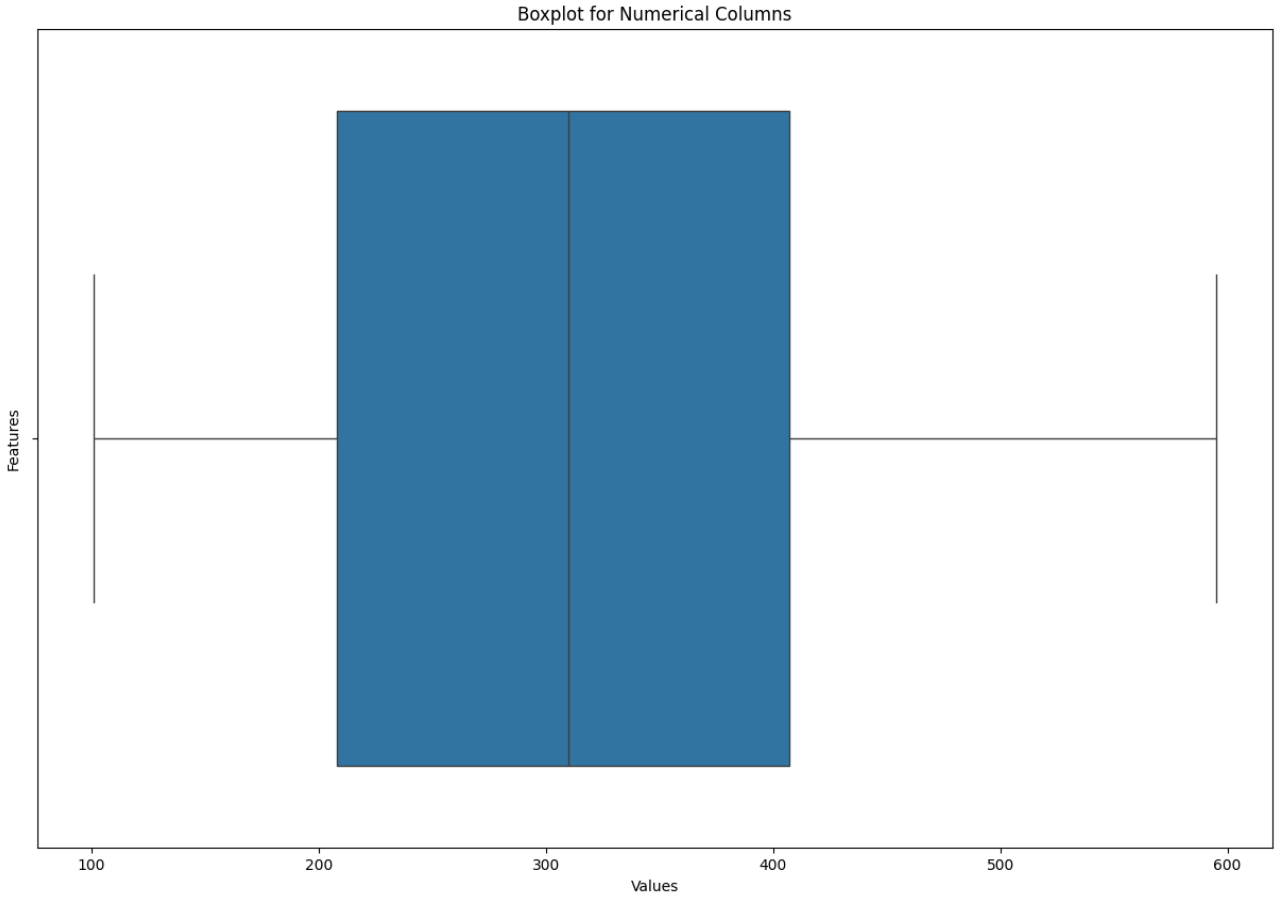


Figure 2.10: The above figure shows the Box Plot of Raw Data for *COMMUNITYBOARD* Column.

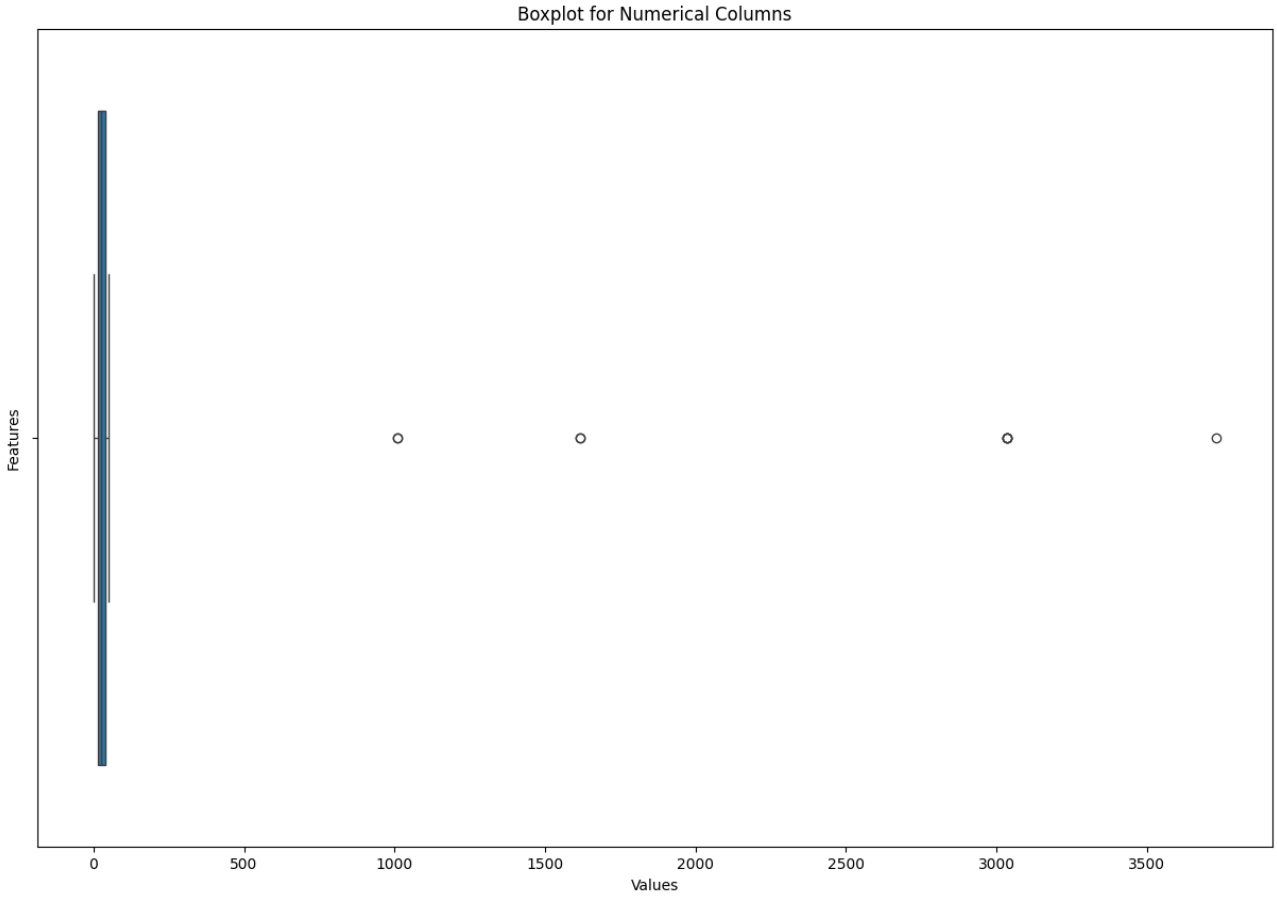


Figure 2.11: The above figure shows Box Plot of the Raw Data for *COUNCILDISTRICT* Column.It depicts a few outliers in this column.

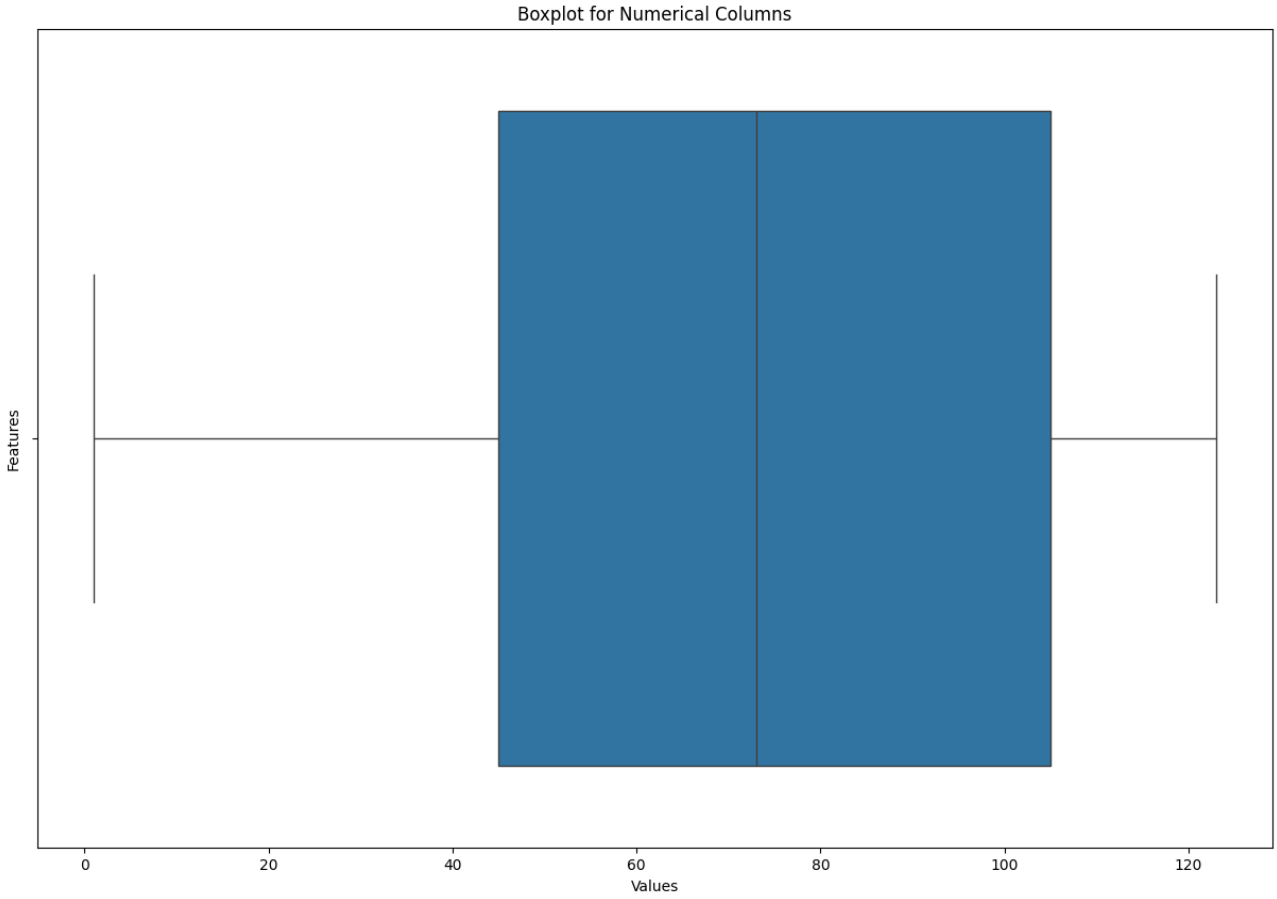


Figure 2.12: The above figure shows Box Plot of Raw Data for the *PRECINCT* Column.

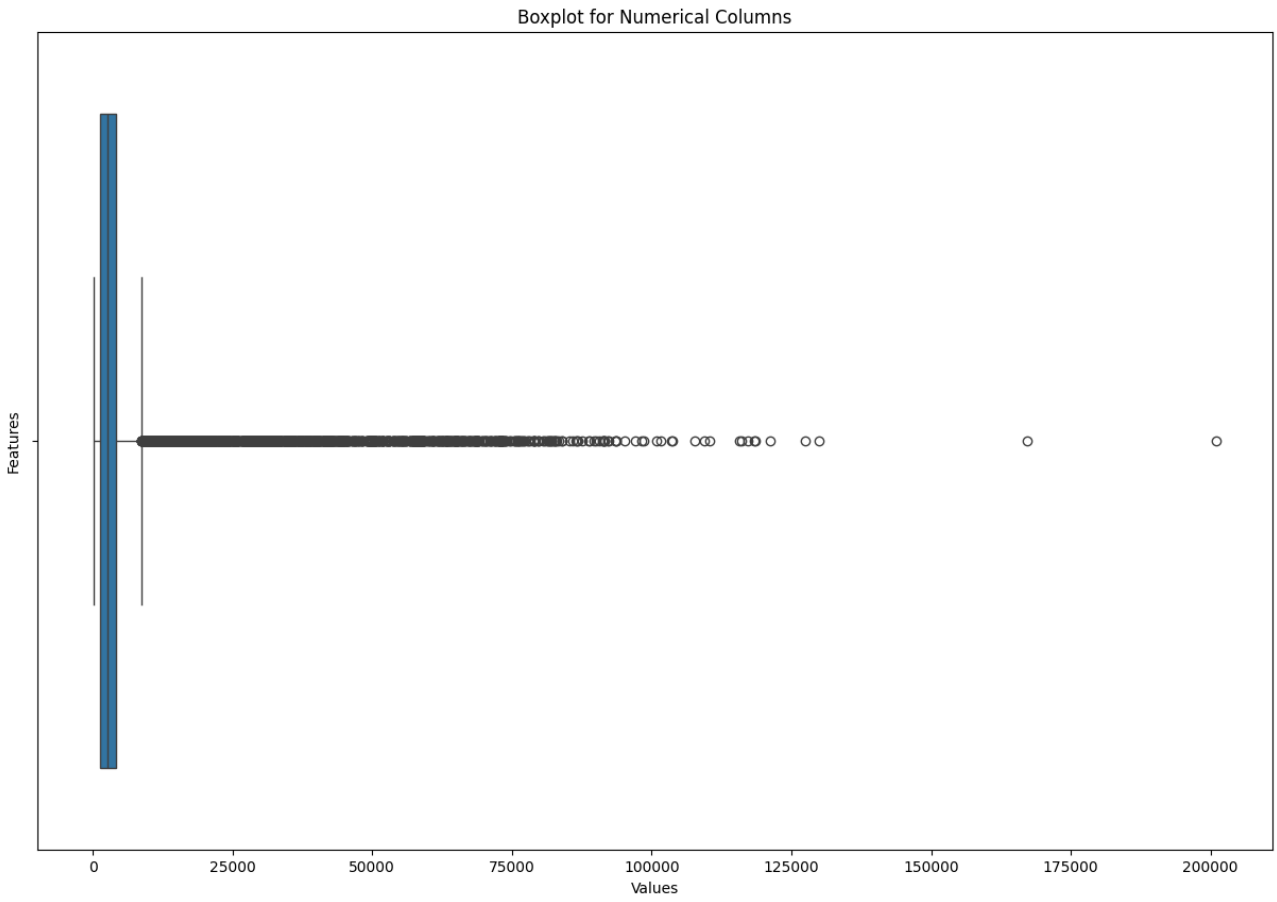


Figure 2.13: The above figure shows Box Plot of Raw Data for the *STArea* Column. This columns depicts a lot of outliers which need treatment.

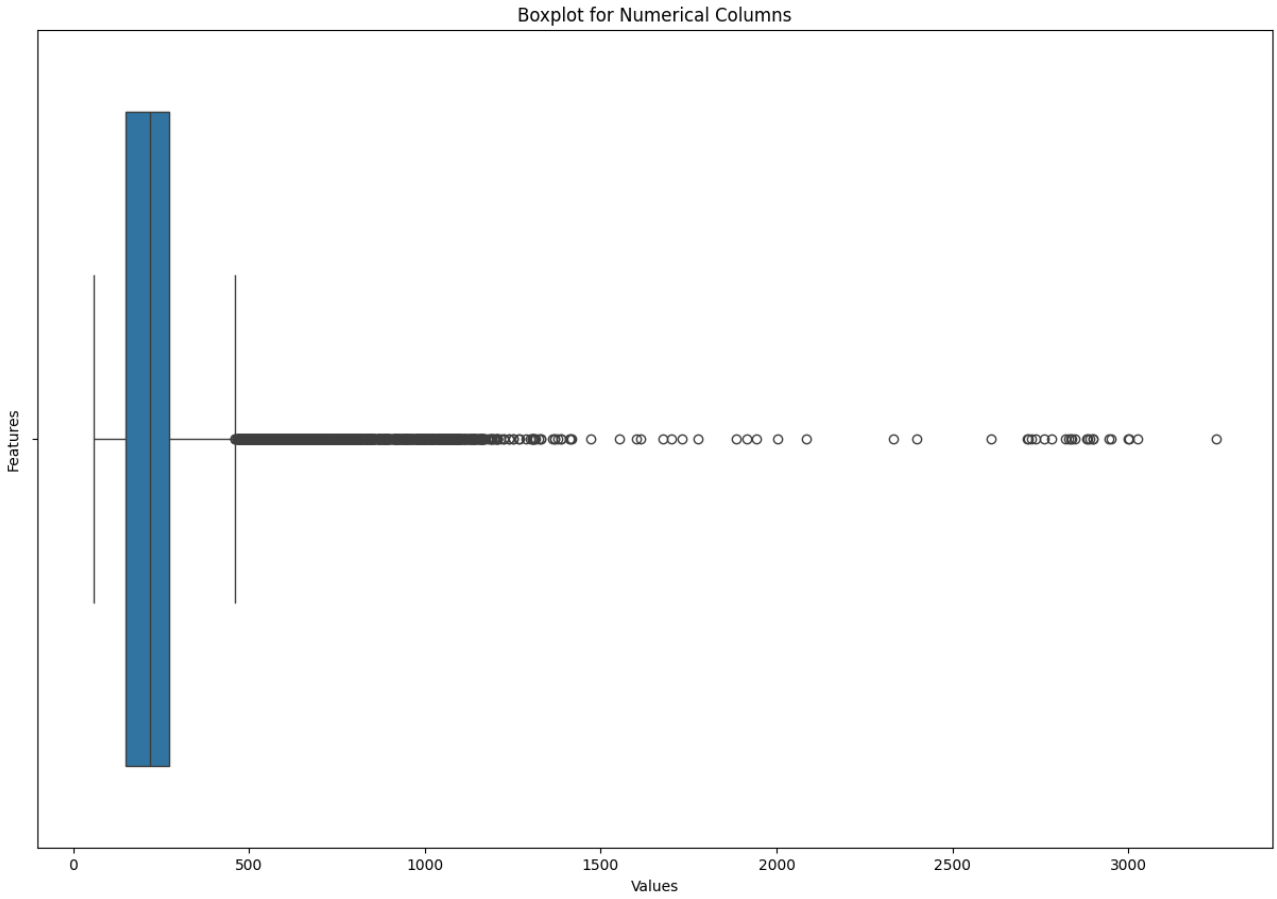


Figure 2.14: The above figure shows Box Plot of the Raw Data for the *STLength* Column.This columns depicts a lot of outliers which need treatment.

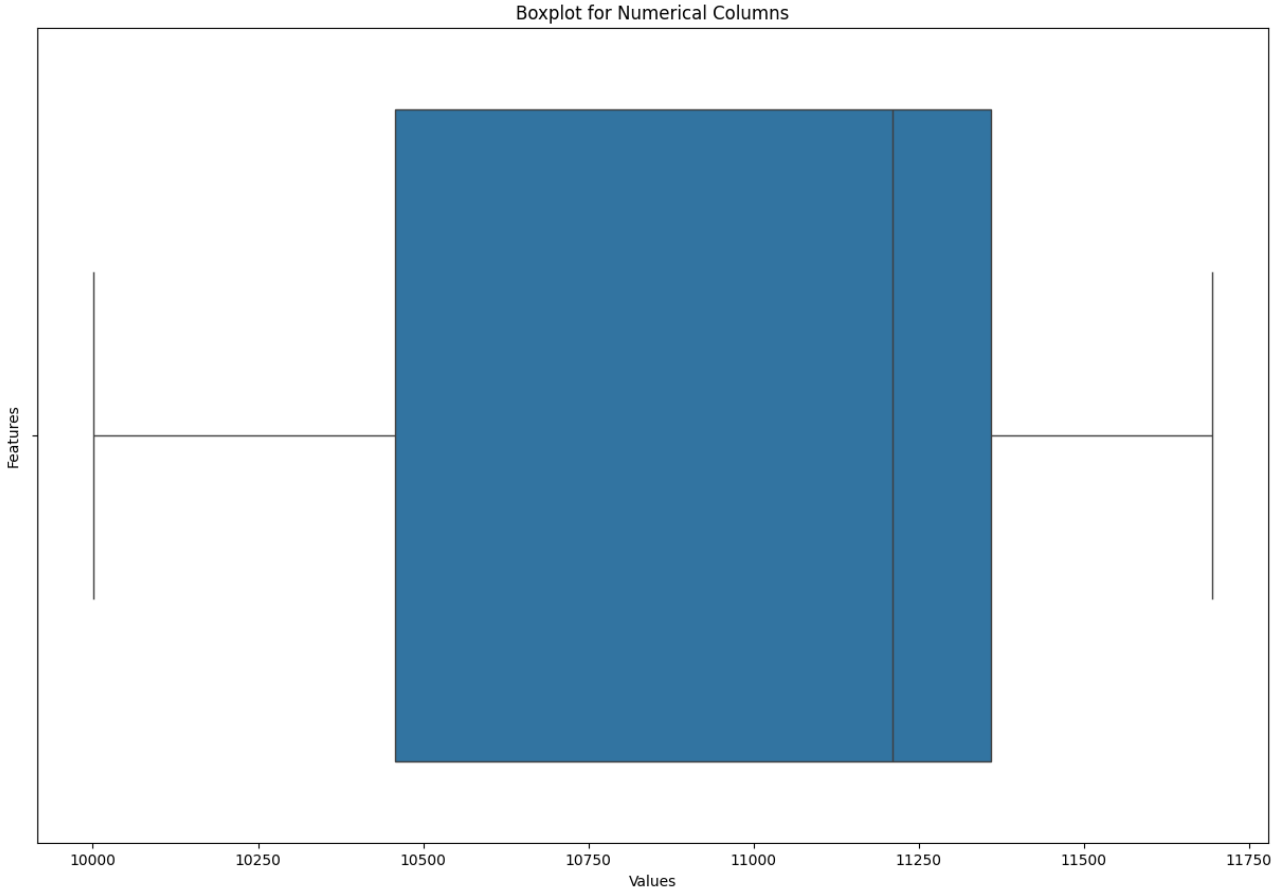


Figure 2.15: The above figure shows Box Plot of Raw Data for the *ZIPCODE* Column.

2.1.3 Missing Data Charts after Removal of Outliers

Outliers of the features were removed, and missing data was plotted again with the Dendrogram, bar chart, and matrix plot to see and understand how they were distributed and correlated after the elimination of outliers. Box plots visuals show that the outliers are removed.

```
df_after = df
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

for column in numerical_cols:
  df = remove_outliers_iqr(df, column)
```

Figure 2.16: **Treatment of outliers**- The above piece of python code depicts the outlier removing function. The given function takes the Quartile values Q1(25% Quantile) and Q3(75% Quantile), and their difference as Inter-Quartile Range(IQR). The threshold values are taken as [Q1 - 1.5*IQR, Q3 + 1.5*IQR], and all values beyond this range are removed from the columns

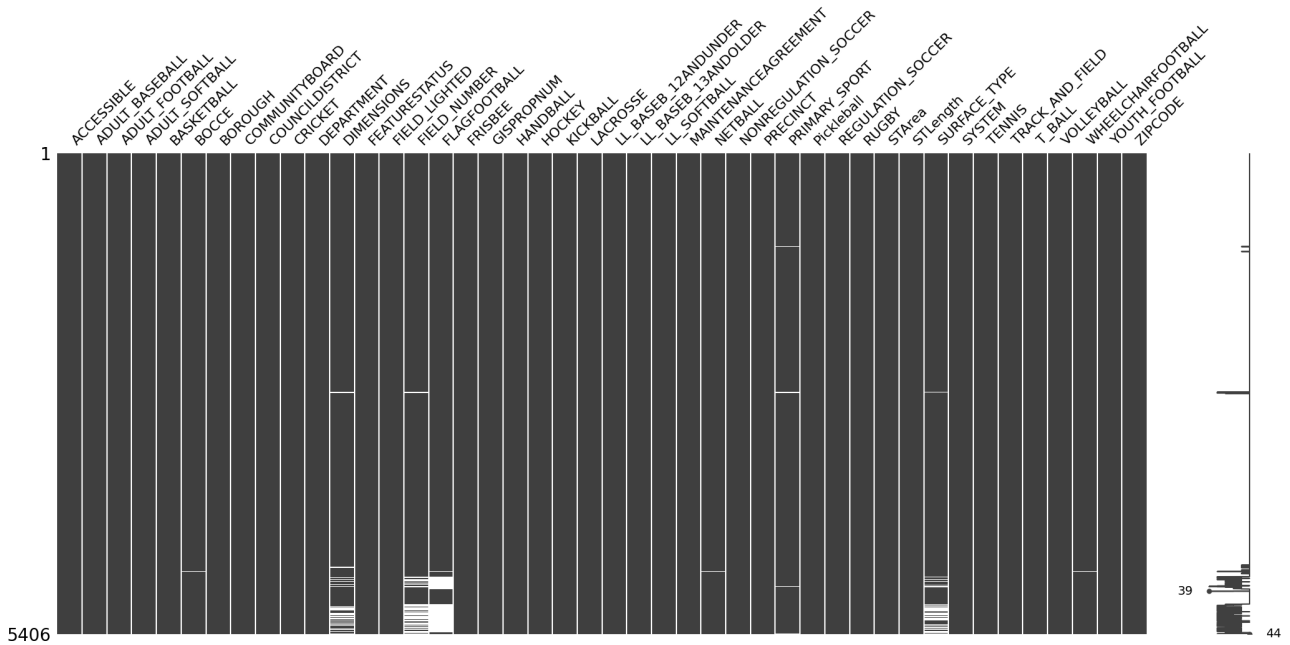


Figure 2.17: The above figure depicts the Matrix plot of the Cleaned Data. The blank nodes in the Matrix represent the missing data.

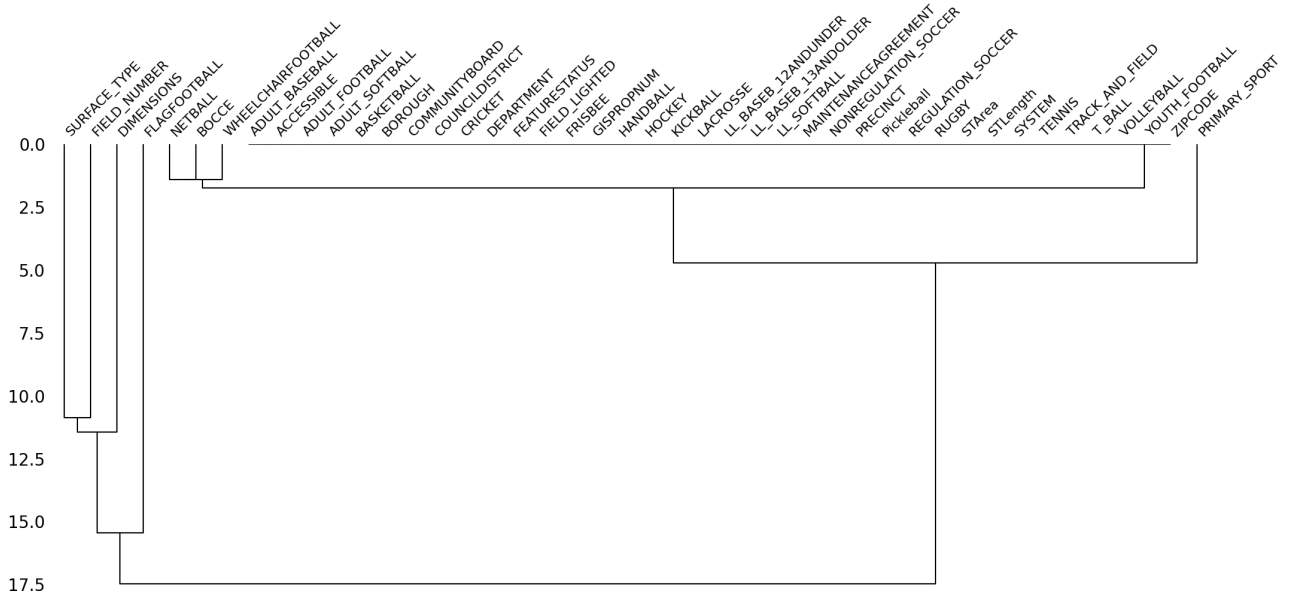


Figure 2.18: The above figure depicts the Dendogram plot of the Cleaned Data. Dendrogram is a form of tree that shows the closeness/similarity of the columns. The closer two columns are placed in the Dendrogram, the more they are similar to each other. We find that there is no change from the raw data except that some nodes are a little closer in distance.

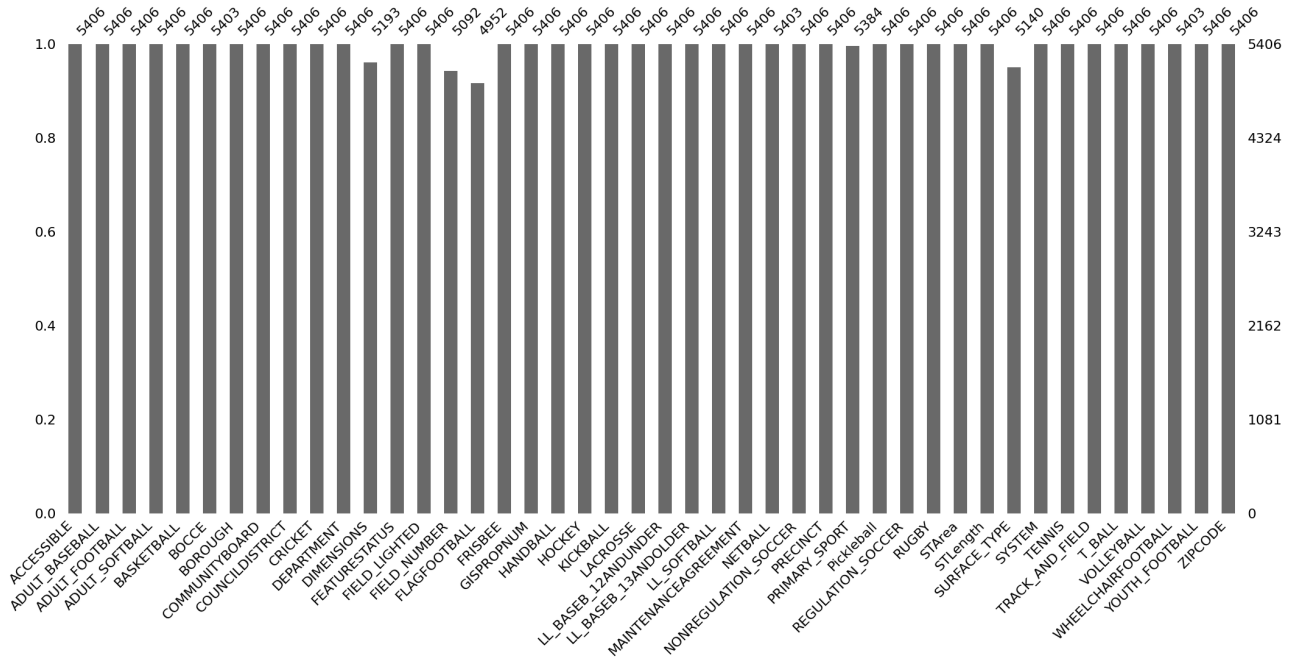


Figure 2.19: The above figure depicts the Bar Map of the Cleaned Data. The short bars in the Bar Map represent the missing data.

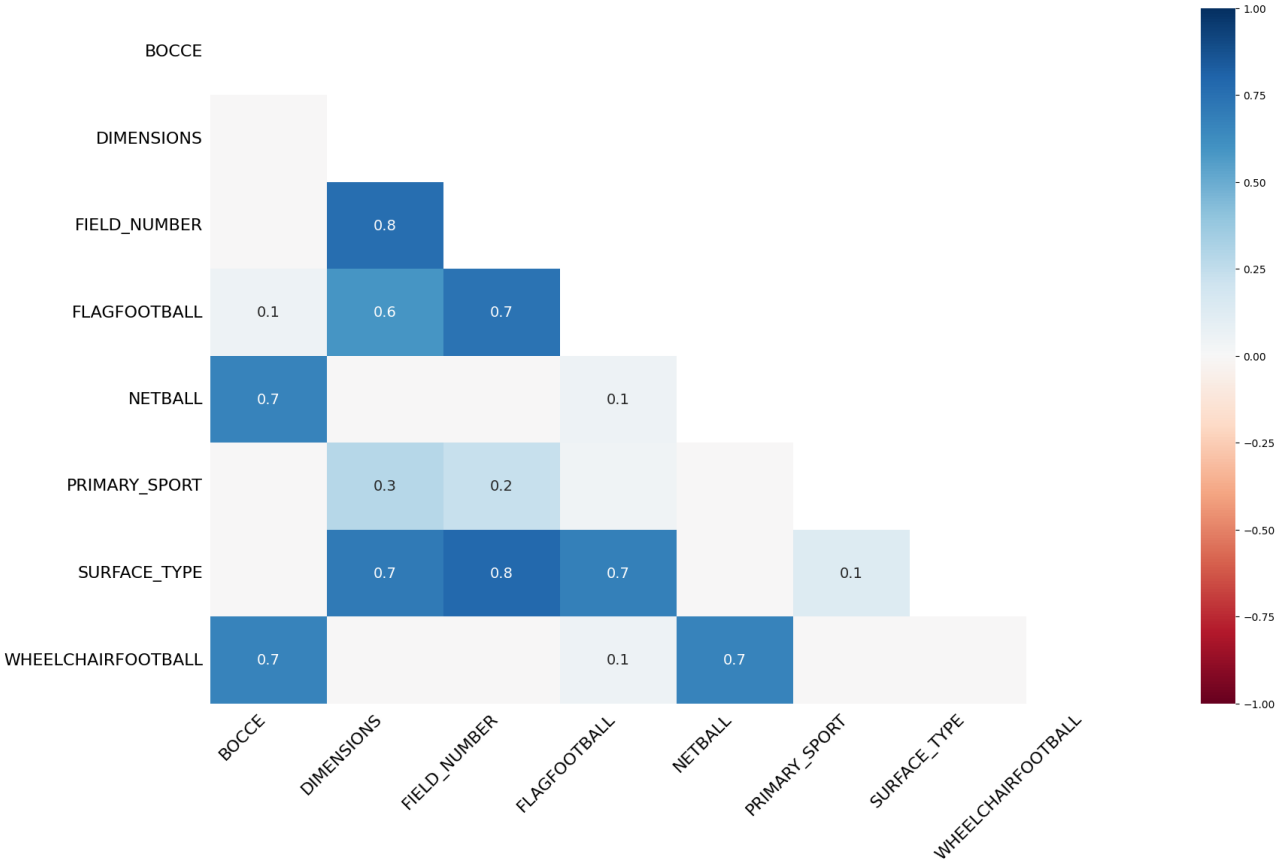


Figure 2.20: The above figure depicts the Heat Map of the Cleaned Data.

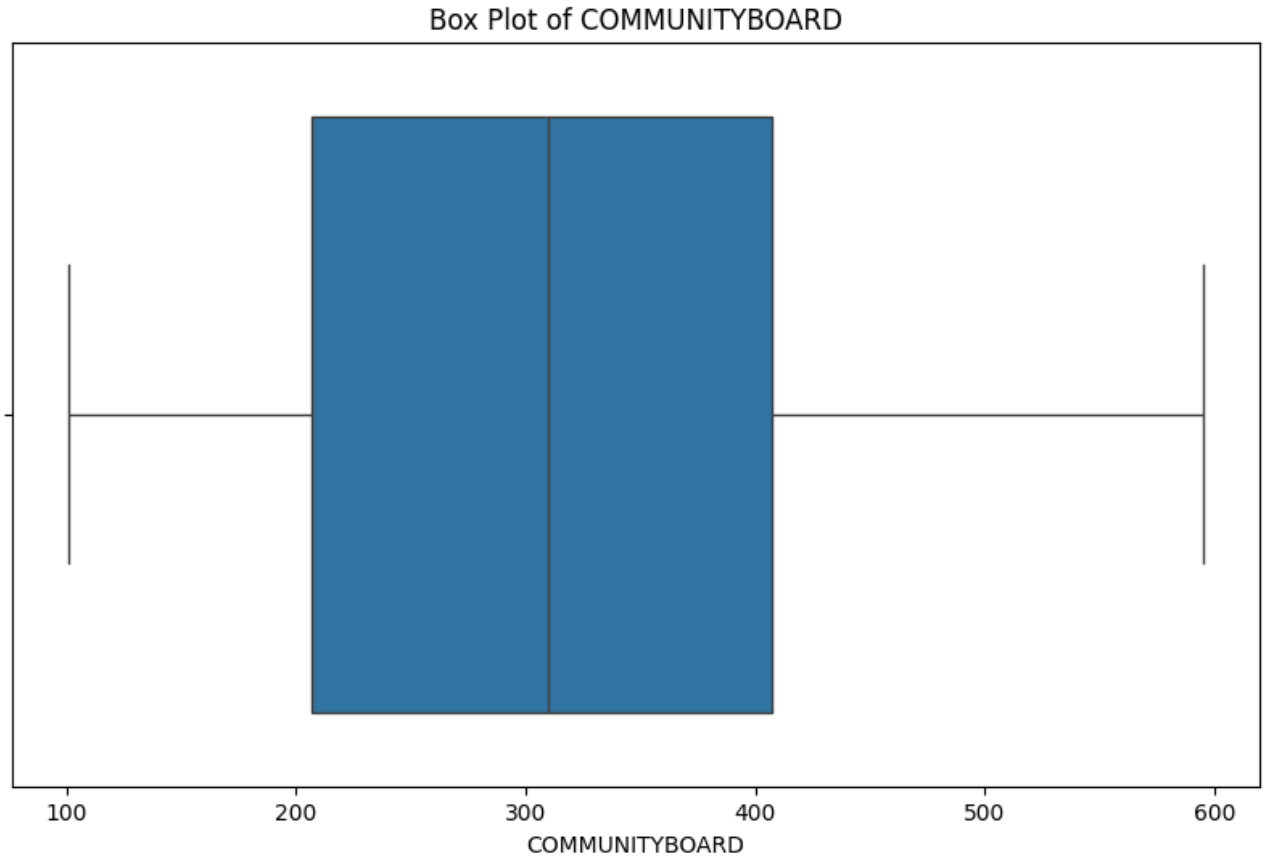


Figure 2.21: The above figure shows Box Plot of the Cleaned Data for *COMMUNITYBOARD* Column.

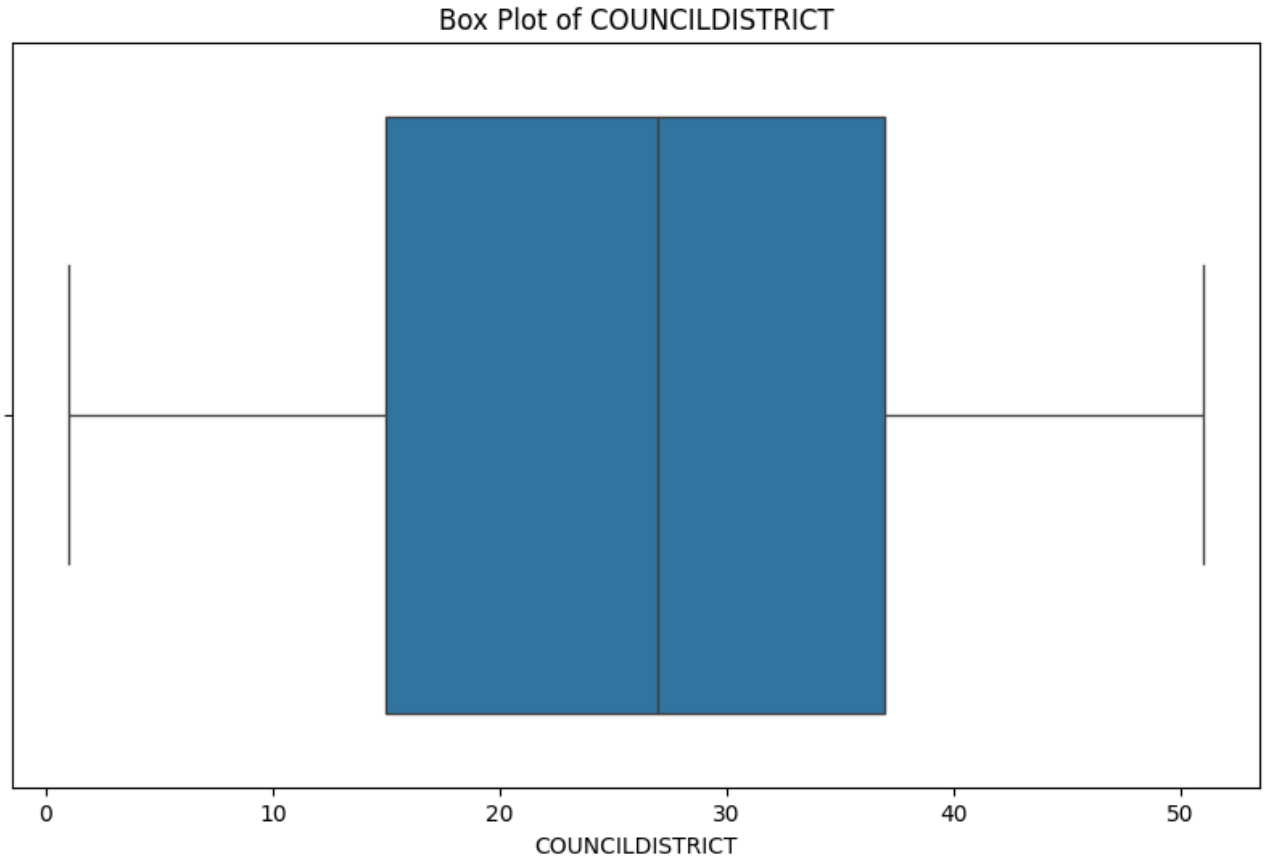


Figure 2.22: The above figure shows Box Plot of the Cleaned Data for *COUNCILDISTRICT* Column.

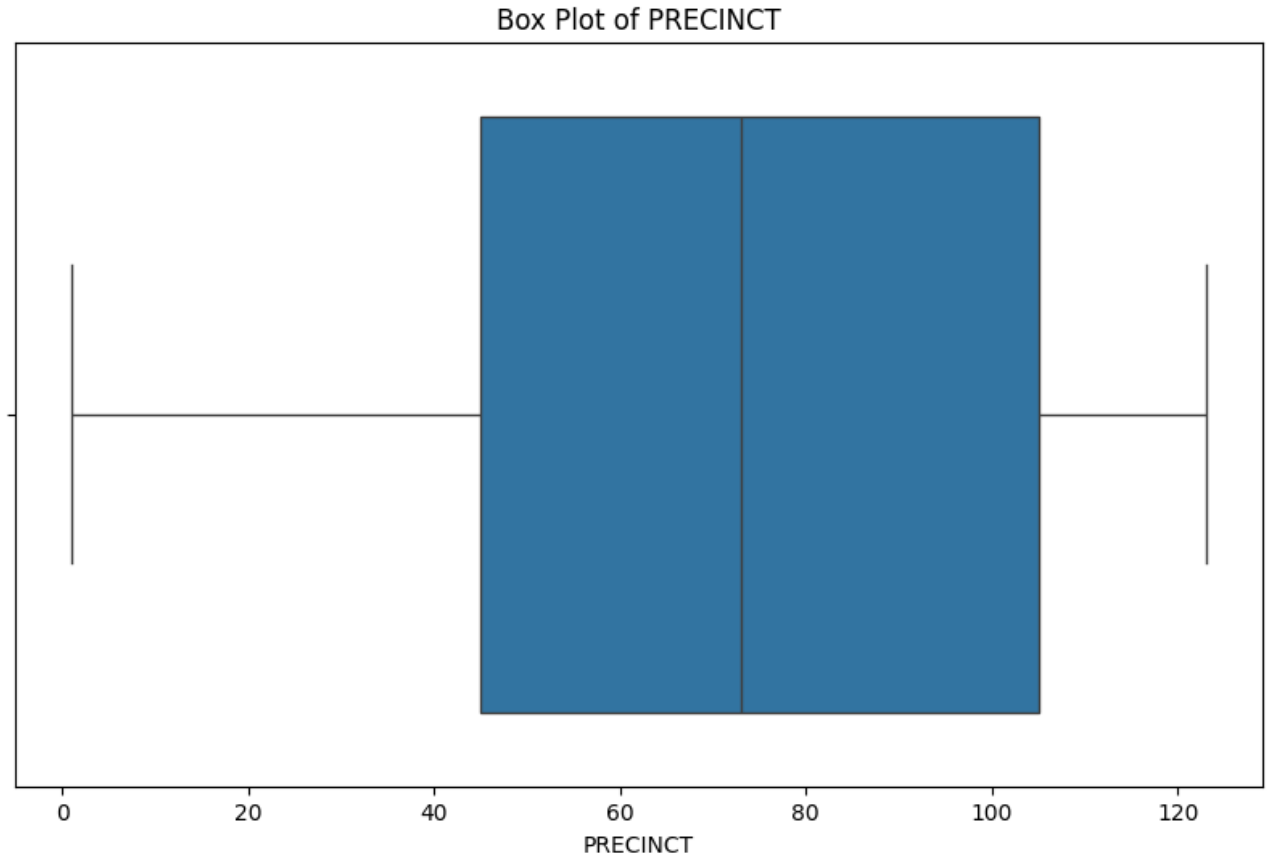


Figure 2.23: The above figure shows Box Plot of the Cleaned Data for *PRECINCT* Column.

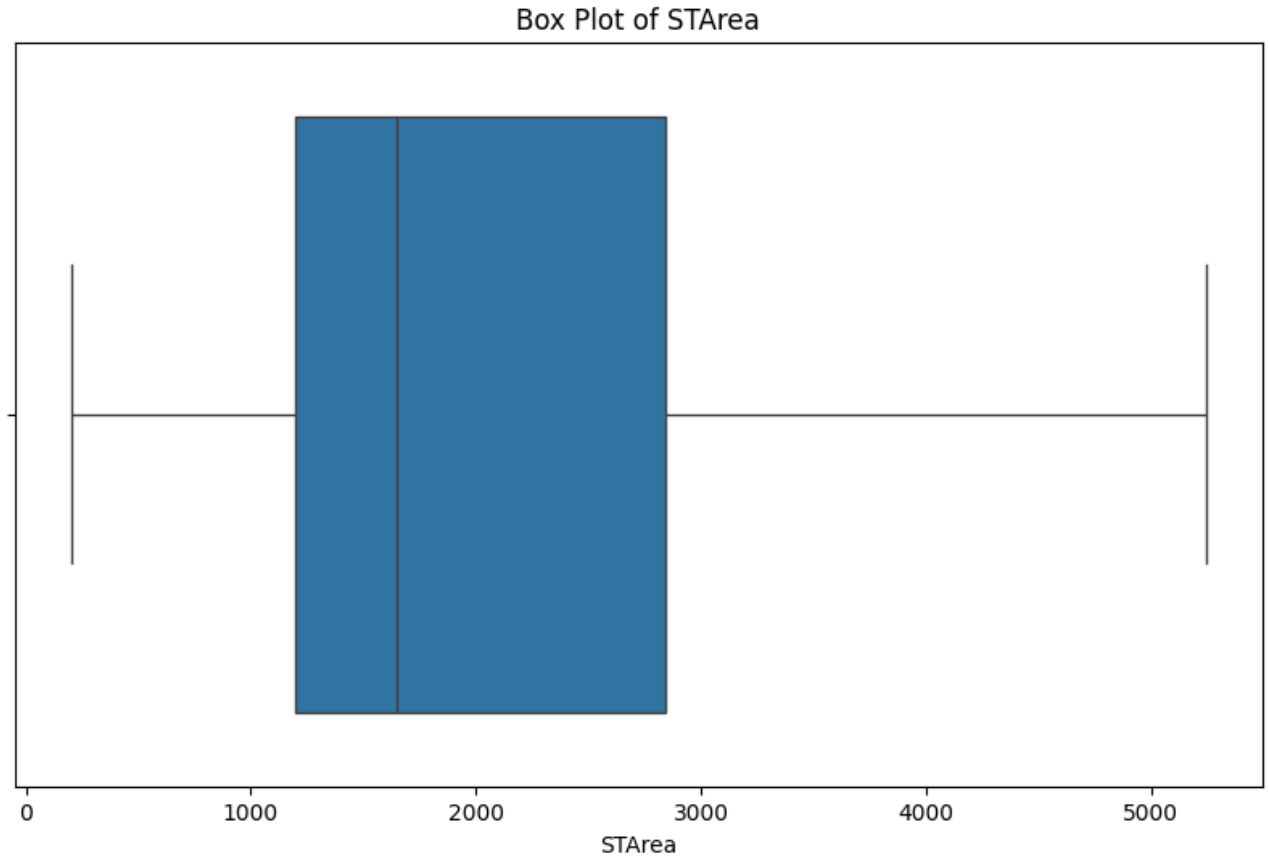


Figure 2.24: The above figure shows Box Plot of the Cleaned Data for *STArea* Column.

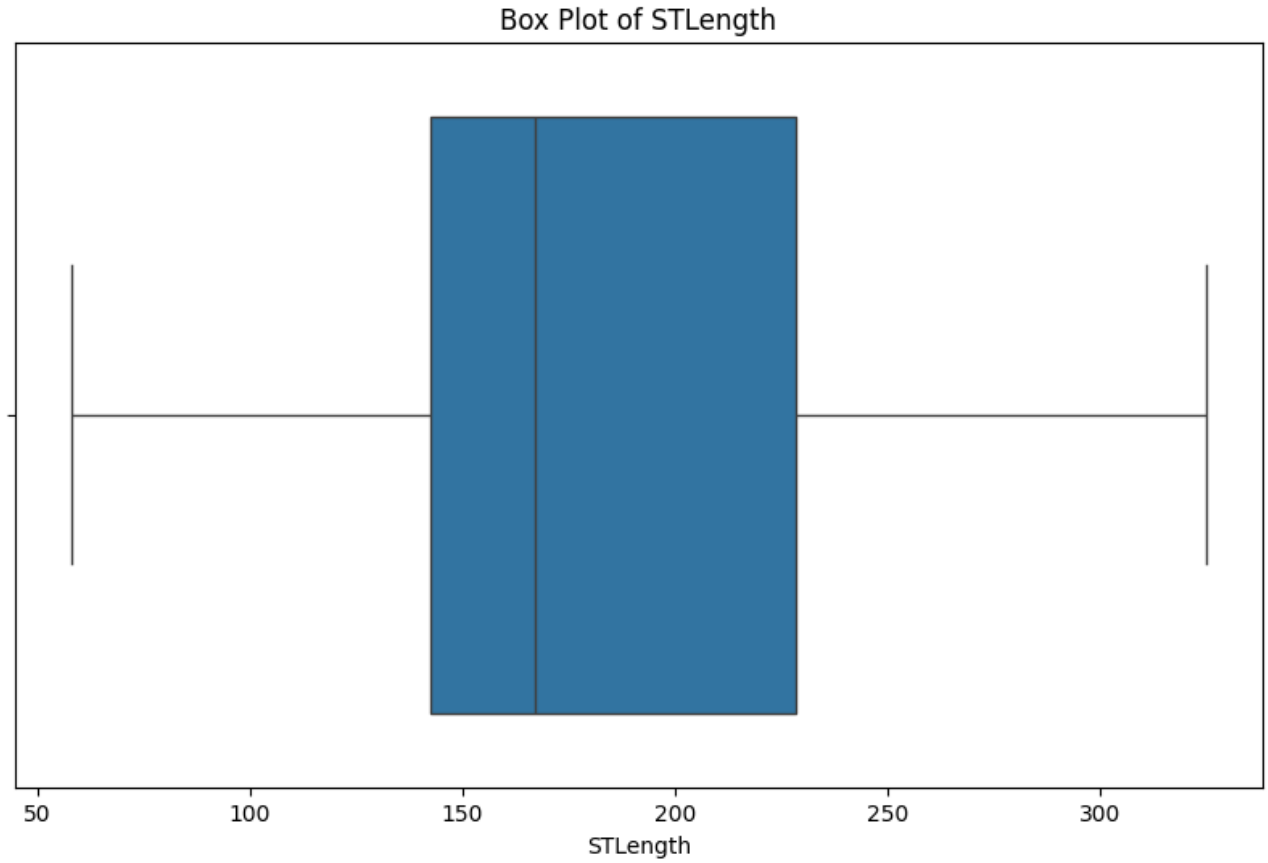


Figure 2.25: The above figure shows Box Plot of the Cleaned Data for *STLength* Column.

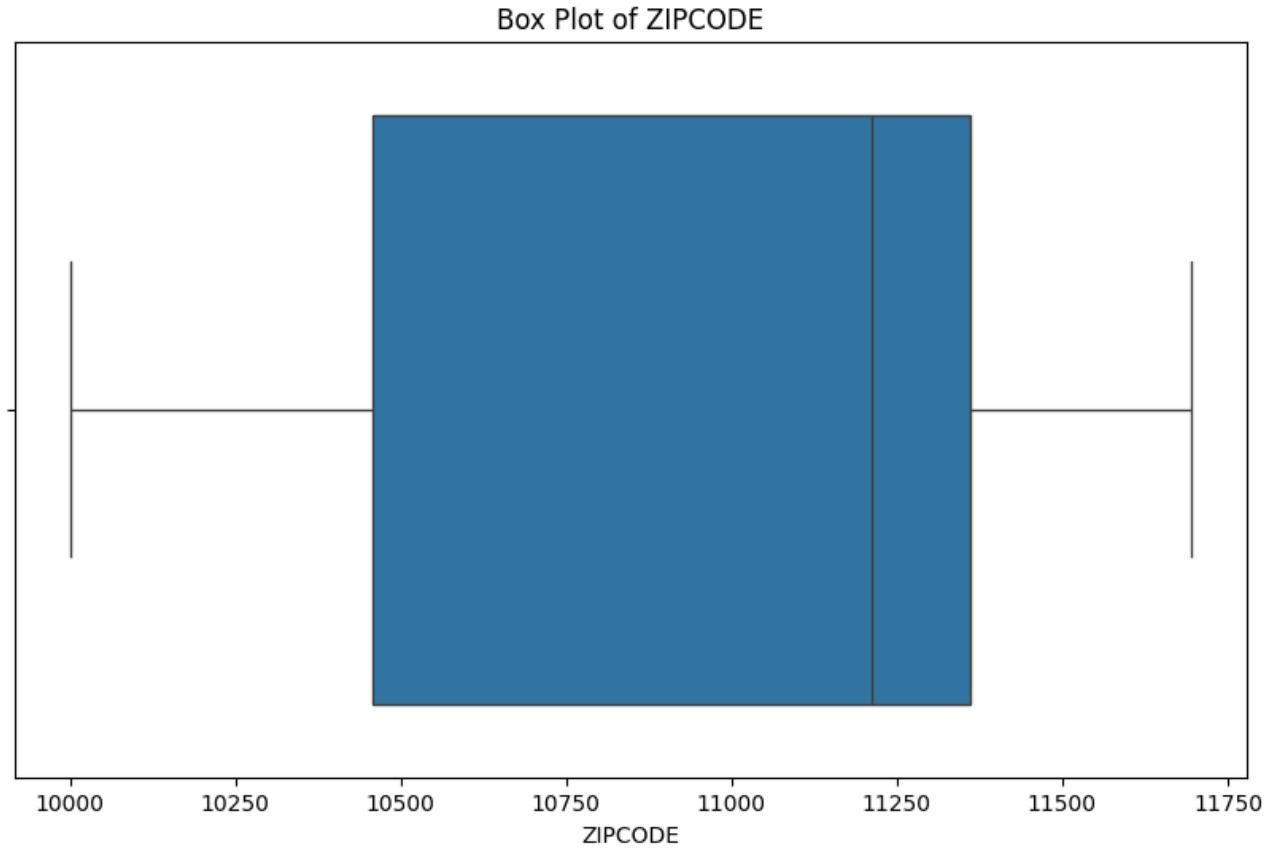


Figure 2.26: The above figure shows the Box Plot of Cleaned Data for *ZIPCODE* Column.

2.1.4 Type of Missingness found in our dataset

After analyzing the outliers and plts generated by using the missingno package, next step is to classify the dataset concerning the missingness. There are three ways possible: Missing Completely at Random , Missing at Random and Missing Not at Random.

The above three missingness will be discussed in short as follows:

- MCAR
  ☞ The probability of value missing is independent of the values of dataset.
  ☞ In MCAR, the probability of a value being missing is independent of other variables in dataset. This basically means that the occurences of missingness is random and no systematic order is followed.

- MAR
  ☞ The probability of value missing may or may not depend on observed values of dataset.
  ☞ In MAR, the probability of a value being missing may likely depend on other observed variables. However, the probability is independent of values which are the missing.

- MNAR
  ☞ The probability of value missing is related to values which are missing and it is significant.

In our analysis, we have conducted Little's MCAR test, so that we came to know that our missingness is of MCAR type or Not.

So the code for the same is:

```
from scipy.stats import chi2

def little_mcar_test(data, alpha=0.05):
    data = pd.DataFrame(data)
    data.columns = ['x' + str(i) for i in range(data.shape[1])]
    data['missing'] = np.sum(data.isnull(), axis=1)
    n = data.shape[0]
    k = data.shape[1] - 1
    df = k * (k - 1) / 2
    chi2_crit = chi2.ppf(1 - alpha, df)
    chi2_val = ((n - 1 - (k - 1) / 2) ** 2) / (k - 1) / ((n - k) * np.mean(data['missing']))
    p_val = 1 - chi2.cdf(chi2_val, df)
    if chi2_val > chi2_crit:
        print(
            'Reject null hypothesis: Data is not MCAR (p-value={:.4f}, chi-square={:.4f})'.format(p_val, chi2_val)
        )
    else:
        print(
            'Do not reject null hypothesis: Data is MCAR (p-value={:.4f}, chi-square={:.4f})'.format(p_val, chi2_val)
        )
little_mcar_test(df,0.05)
```

```
Do not reject null hypothesis: Data is MCAR (p-value=1.0000, chi-square=531.7155)
```

Figure 2.27: Performing Little MCAR test to check whether the missingness is of type MCAR or not.We can see from the code that the null hypothesis is not rejected .So missingness is of MCAR type.

The following Python code defines a function, Little MCAR test, that performs a test to discover if the missing values of a dataset are intended to be Missing Completely at Random . This function takes a dataset as input, calculates a test statistic , and compares the result with a critical value based on the chi-square distribution at a specified significance level . The output provides information about rejecting and not rejecting the null hypothesis: the data is not MCAR if the test statistic is larger than the critical value and the data is MCAR if the test statistic is lesser than or equal to the critical value. Therefore, after this test, we came to know whether our missingness is mcar or not.

So we can see clearly from the output that missingness is of MCAR type.

2.2 Imputation

Here as we have seen missingness is of MCAR type and we have many missing values .So we have to impute or fill those missing values for further analysis. Here we will discuss how we will impute the numerical as well as categorical columns.

1.) **Numerical Columns** :-

If missingness in our dataset is of MCAR type that means there is no relationship between the probability of a value missing and other varibles, so then the correct method for handling the missing

values is mean imputation or median imputation .

In the case where our data seems normal, which means the distribution of our dataset is bell-shaped, expert would use mean imputations, which replaces values that are missing with mean of non- missing values for that variable.

If the distribution is skewed, i.e., not symmetrical, one can use the median for imputation as it is considered more robust than the mean when considering outliers and extreme values.

Therefore, if we print the df.skew() for numerical columns in our dataset. i.e., COMMUNITYBOARD it has skewness equal to -0.241179 (slight left skewed) COUNCILDISTRICT has skewness equal to 31.619119 (high right skewed) PRECINCT has skewness equal to -0.254286(slight left skewed) STArea has skewness equal to 2.842526 (right skewed) STLength has skewness equal to 3.356757 (right skewed) and last ZIPCODE has skewness equal to -0.578887 (slight left skewed) .

We have displayed df.skew() for numerical columns for your reference :

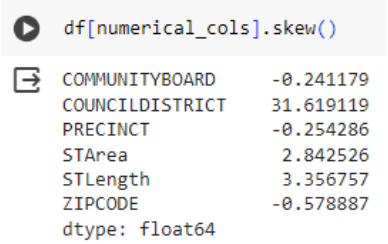


Figure 2.28: df.skew . It displays the skewness of the numerical columns.Negative skewness value means it is left skewed and positive skewness value means it is right skewed

But among this, only ZIPCODE has a missing value and also this has a skewness of very low i.e., 0.578887. Therefore, we can impute a mean for all numerical columns.

So we can do mean imputation of all numerical columns.

2.) **Categorical Columns** :-

→ For Categorical Columns , we have imputed them with the mode, as we got missingness of MCAR type .

Chapter 3. Visualization

Visualizations are a crucial component of EDA as it allows to form an idea about the distribution, relationship, and patterns of data. Here, we will explore various visualisations to get insights into the dataset.

3.1 Univariate analysis

In univariate analysis, we will analyze each variable in the dataset. The goal of the univariate analysis is to understand the distribution of each variable and examine their statistical properties. The main techniques involved in this are:

- Pie charts
- Bar charts
- Histograms
- Line charts
- Waterfall charts
- Violin plots
- Kernel Density Estimation plots

3.1.1 Pie Charts

Pie charts were used to present the distribution of different features. They provide a representation of the proportion of each category within a feature. The pie charts provided valuable visualization of how each feature is distributed in the dataset. The following are some examples of pie charts during this study:

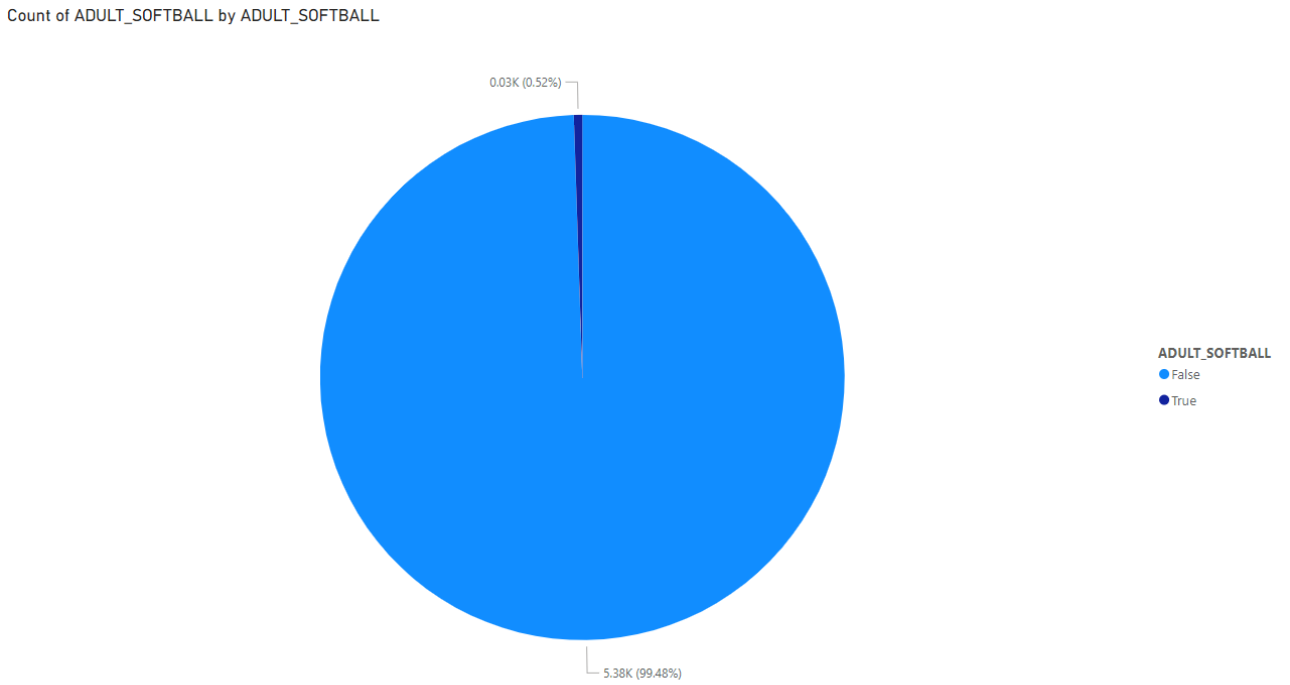


Figure 3.1: The above figure depicts a Pie Chart representing distribution of availability of *ADULT_SOFTBALL* fields. Since the records have not been grouped according to the *DIMENSIONS* and Districts, the chart shows a significant portion as FALSE.

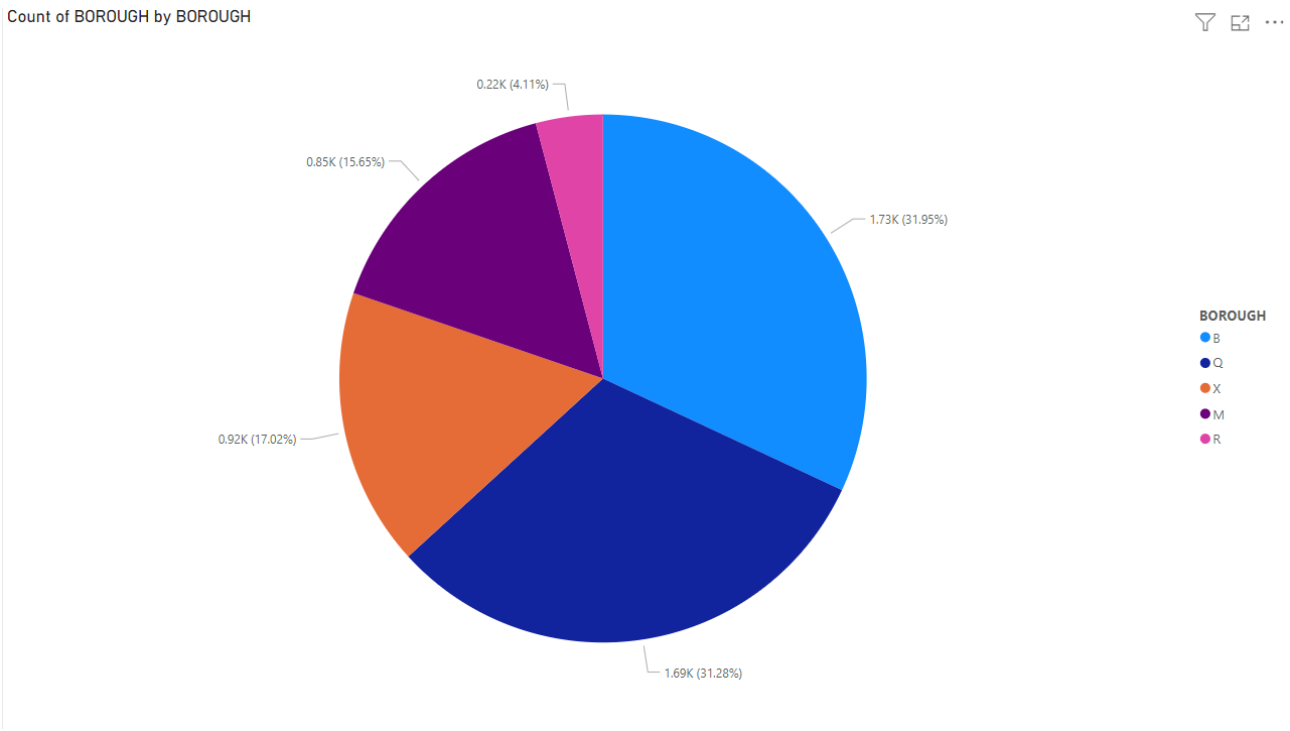


Figure 3.2: The above figure depicts a Pie Chart representing distribution of fields in the *BOROUGH*. We can see that two *BOROUGH* 'B' and 'Q' contain nearly a third of the total fields each and the rest of the *BOROUGH* account for the rest.

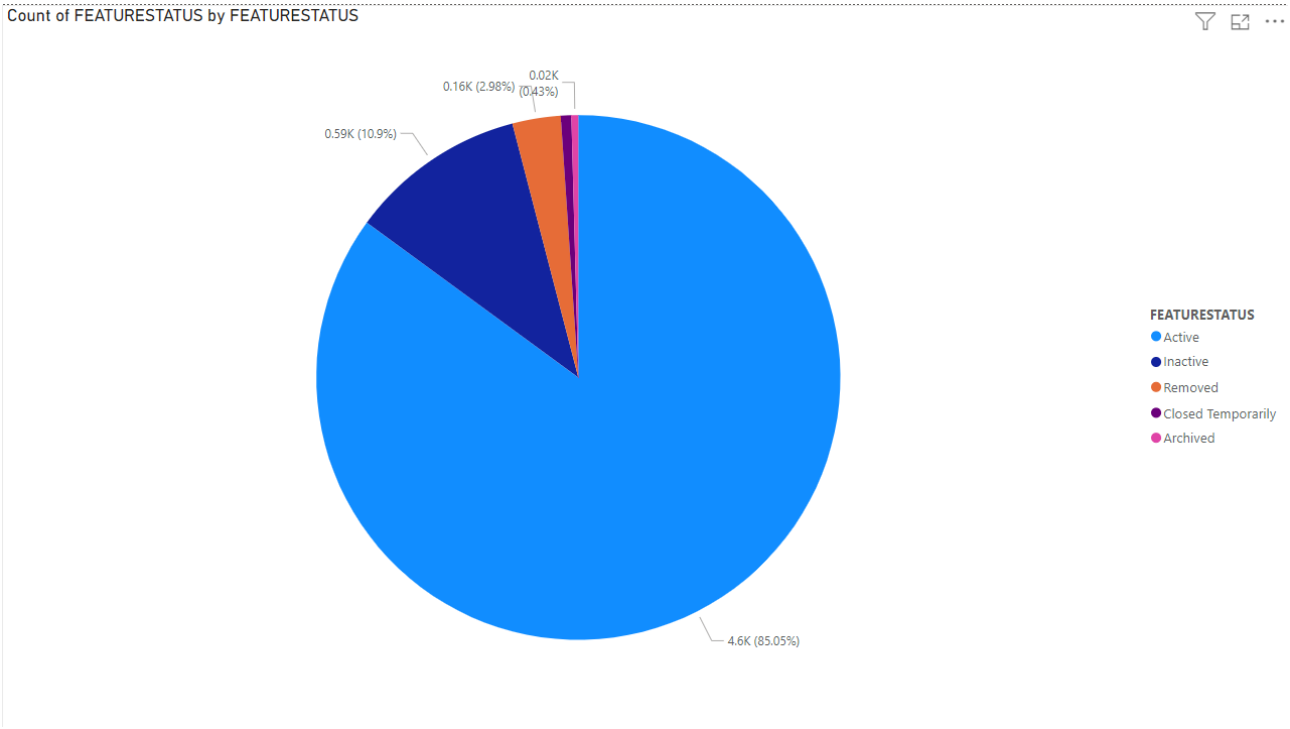


Figure 3.3: The above figure depicts a Pie Chart representing distribution of *FEATURESTATUS* of the fields. While most of the fields are active and running, nearly $\frac{1}{7}$th of them are not operational due to various reasons.

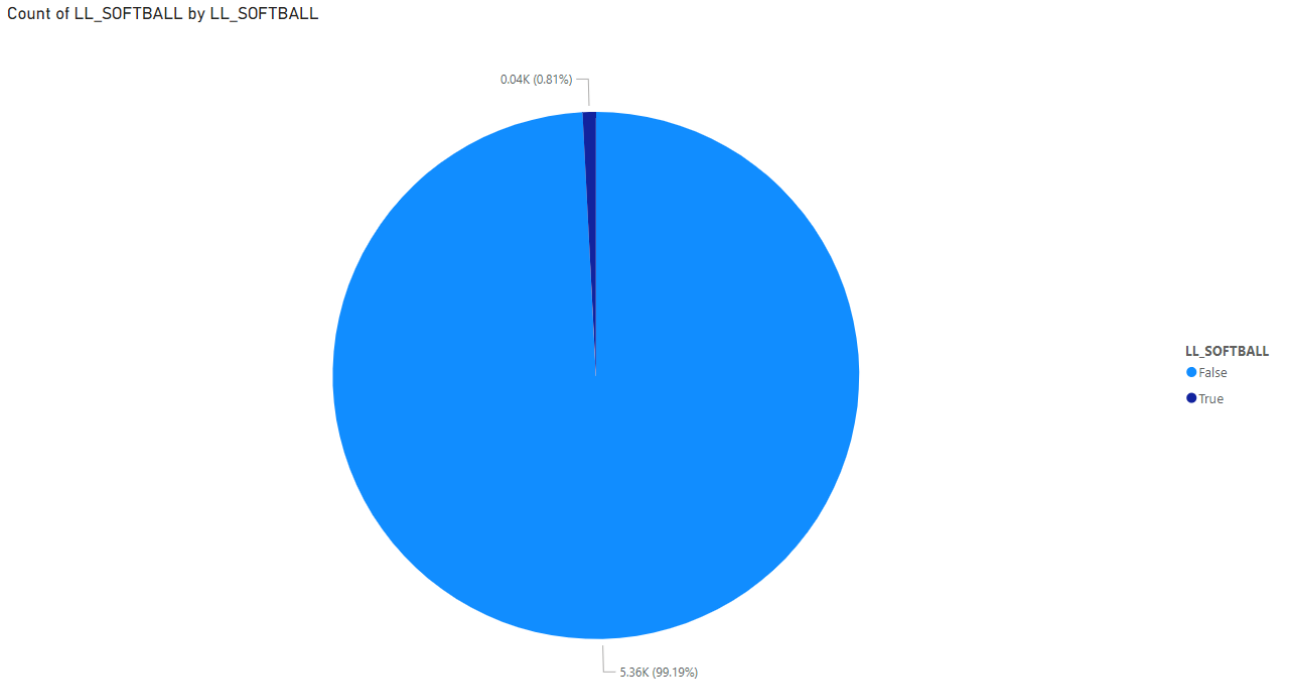


Figure 3.4: The above figure depicts a Pie Chart representing distribution of availability of *LL_SOFTBALL* fields. Since the records have not been grouped according to the *DIMENSIONS* and Districts, the chart shows a significant portion as FALSE.

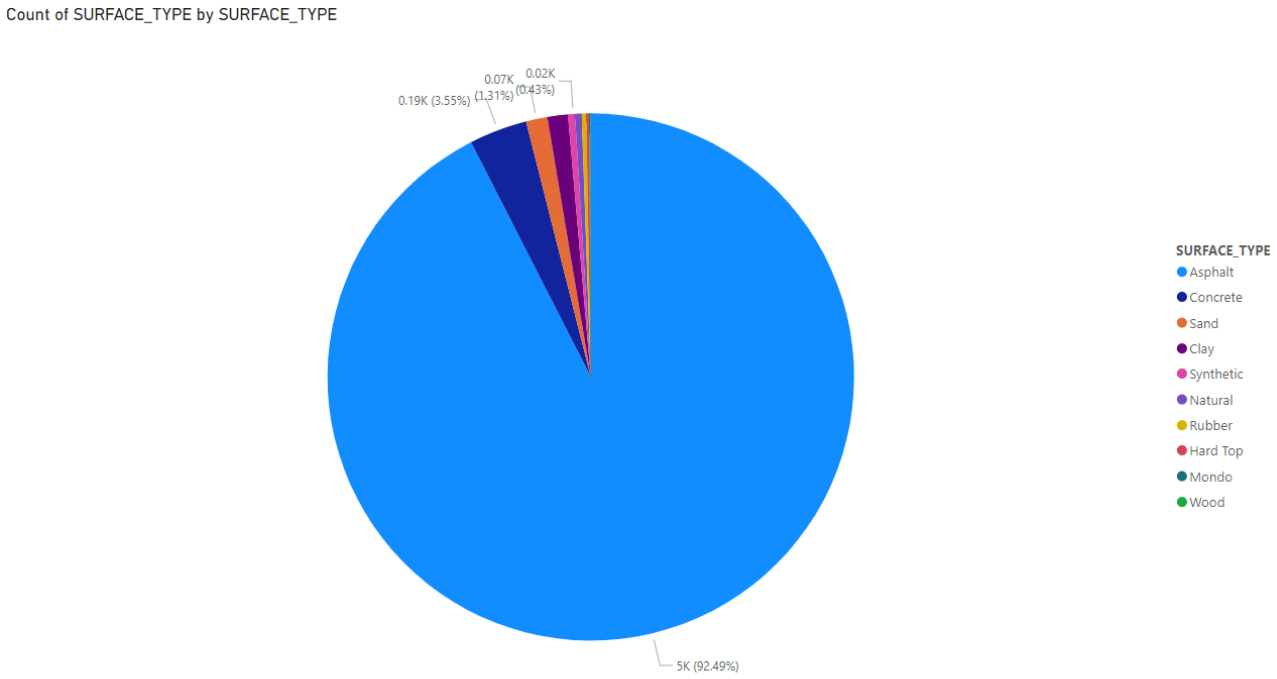


Figure 3.5: The above figure depicts a Pie Chart representing distribution of *SURFACE_TYPE* of the fields. We see that although there are 10 different types of materials used to make the field grounds, only two materials, namely Asphalt and Concrete account for more than 96% of them.

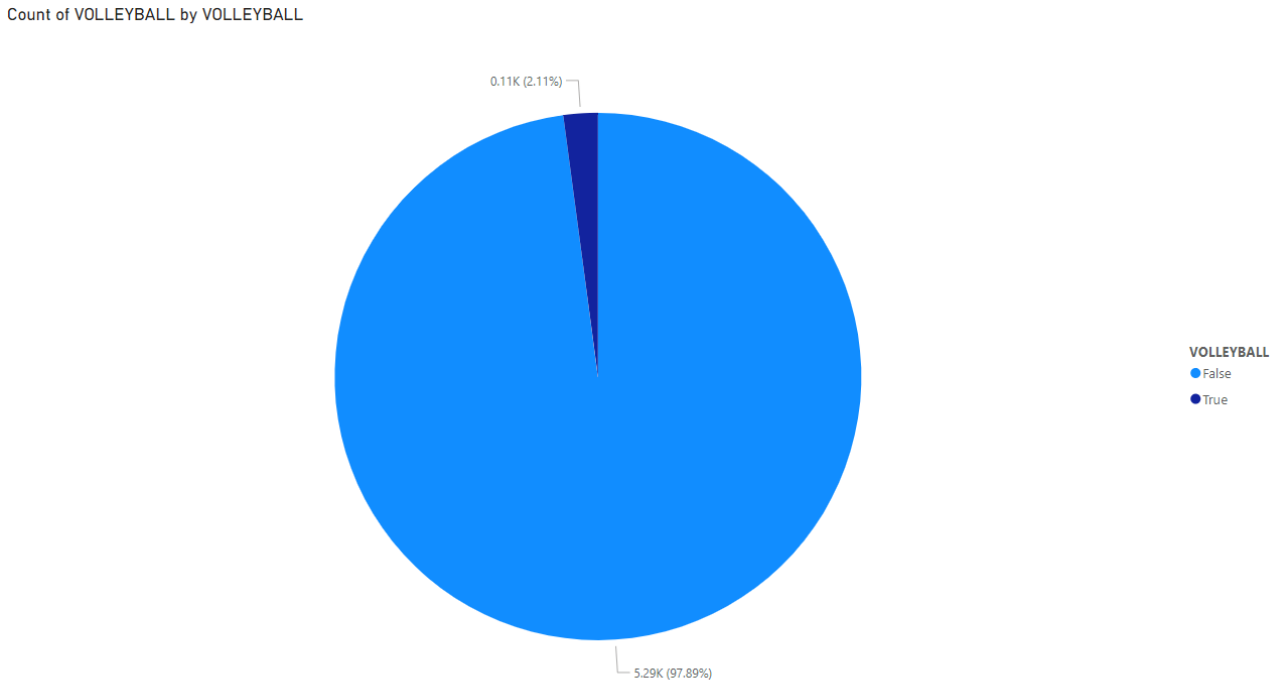


Figure 3.6: The above figure depicts a Pie Chart representing distribution of availability of *VOLLEYBALL* fields. Since the records have not been grouped according to the *DIMENSIONS* and Districts, the chart shows a significant portion as FALSE.

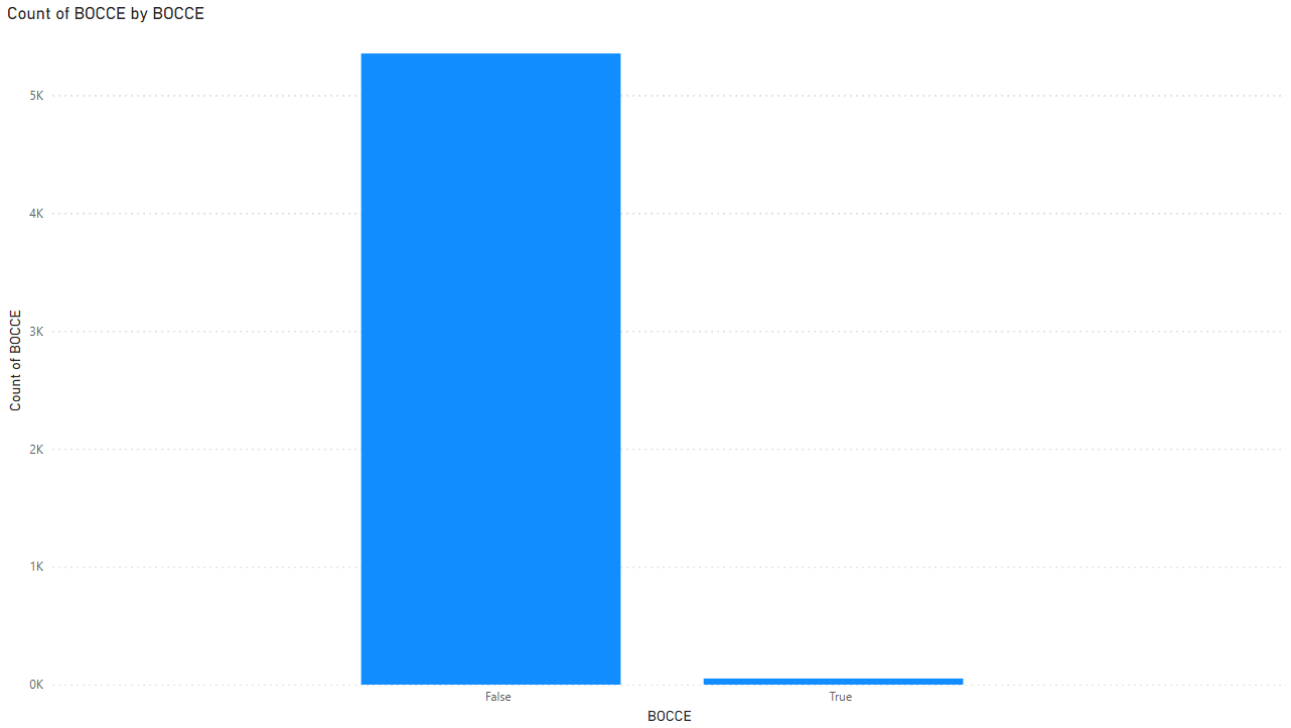


Figure 3.7: The above figure depicts a Bar Chart exhibiting distribution of availability of *BOCCE* fields. Since the records have not been grouped according to the *DIMENSIONS* and Districts, the chart shows a significant portion as FALSE.

3.1.2 Bar Charts

Bar chart is a graph of categorical data represented by rectangular bars. The height of the bar is the frequency of category. Bar chart is used to show the distribution of categorical data such as the count of athletic facilities across different boroughs in our dataset. The following are some examples of bar charts during this study:

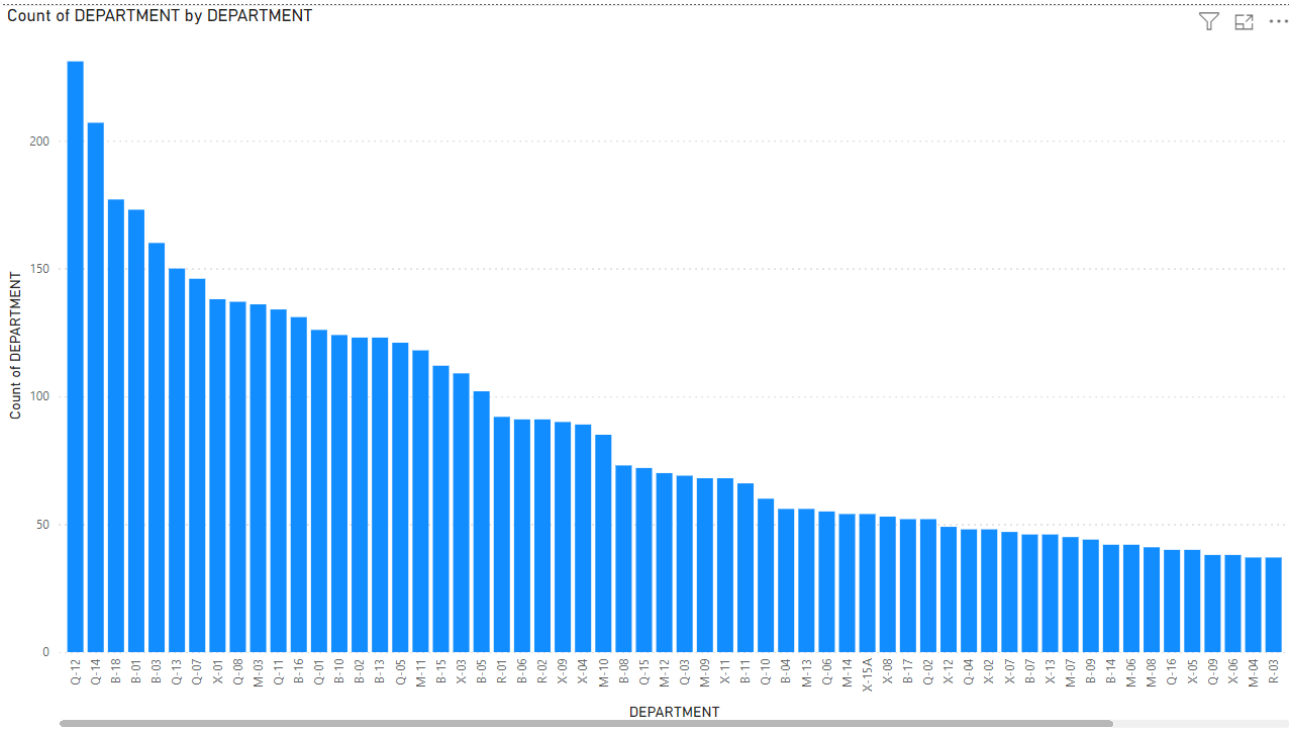


Figure 3.8: The above figure depicts a Bar Chart exhibiting amount of fields under each *DEPARTMENT*. On average the average number of fields under a *DEPARTMENT* under the *BOROUGH* 'B' and 'Q' are $2.5 - 3$ times that of others.

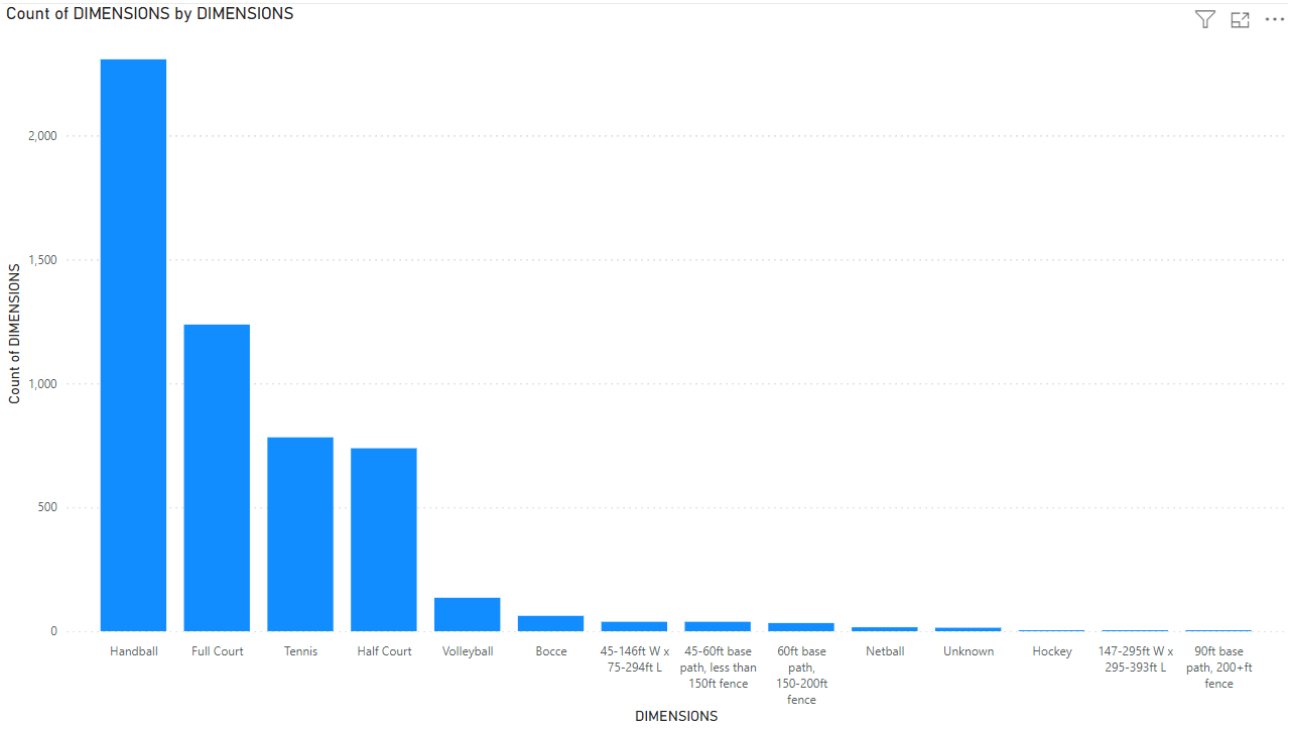


Figure 3.9: The above figure depicts a Bar Chart exhibiting *DIMENSIONS* of various fields. Obviously, many ball game courts have the numerical majority.

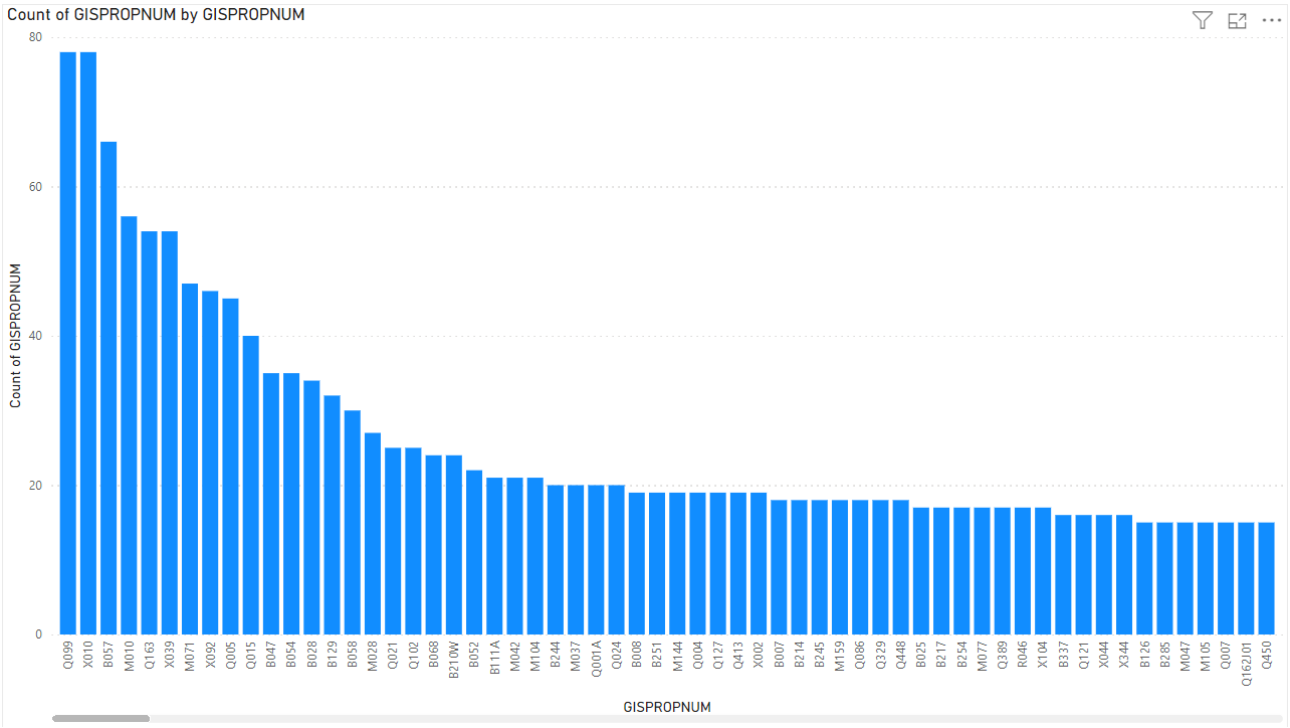


Figure 3.10: The above figure depicts a Bar Chart exhibiting distribution of various fields according to their *GISPROPNUM*. It depicts the number fields under a property.

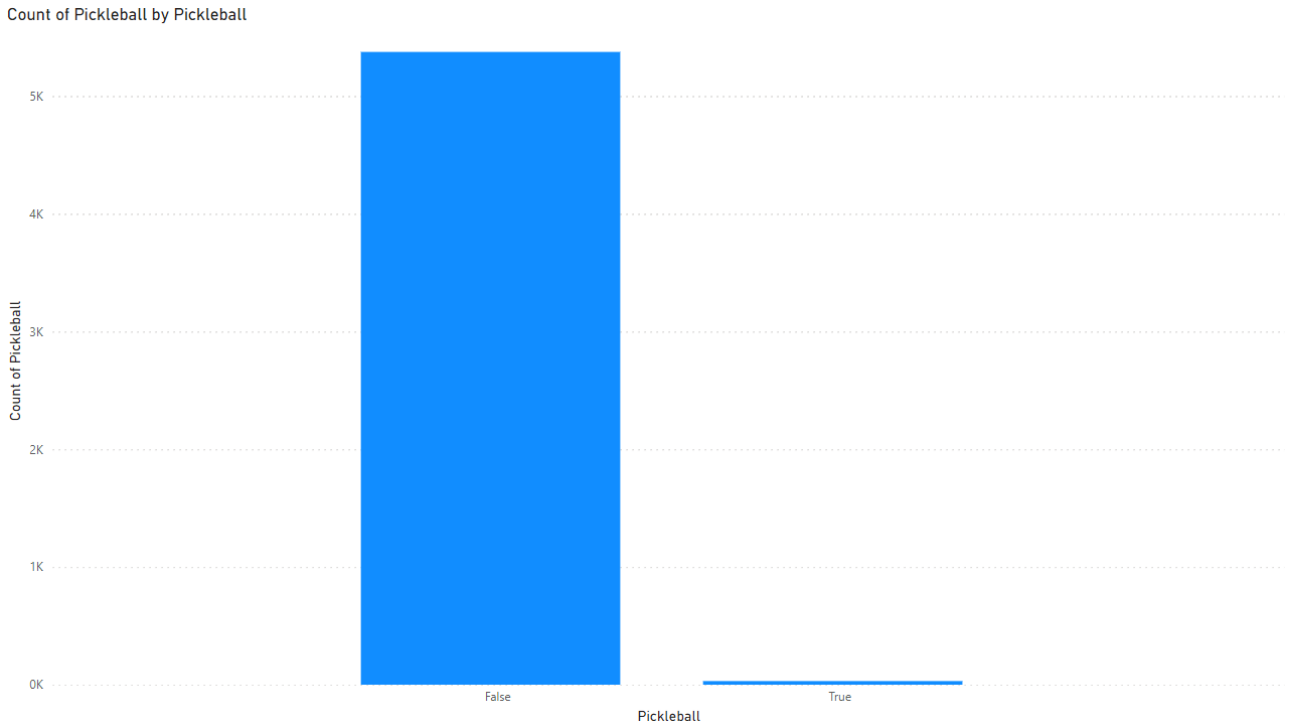


Figure 3.11: The above figure depicts a Bar Chart exhibiting distribution of availability of *Picleball* fields. Since the records have not been grouped according to the *DIMENSIONS* and Districts, the chart shows a significant portion as FALSE.

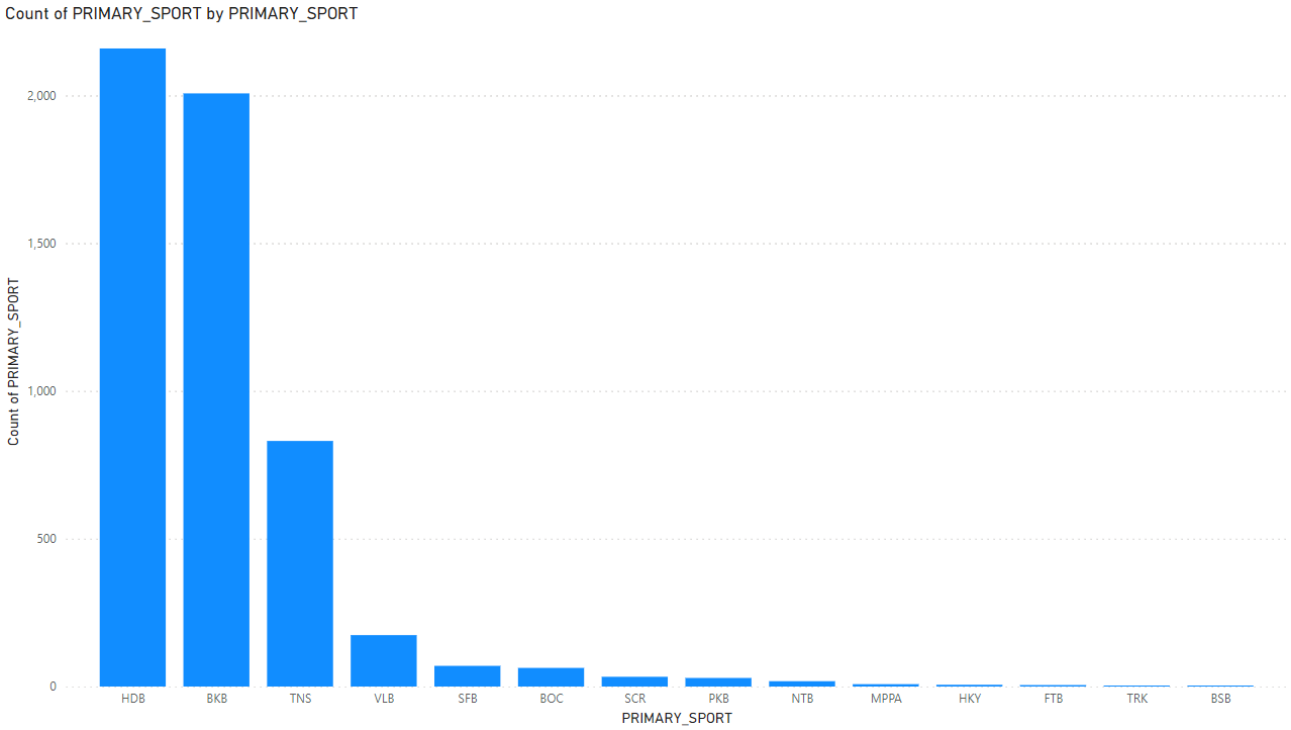


Figure 3.12: The above figure depicts a Bar Chart exhibiting distribution of fields for various *PRIMARY_SPORT*. The three sports Handball, Basketball and Tennis account for more than 92% of them.

3.1.3 Histograms

Histogram is a graph of categorical data similar to bar chart. The height of the bar is the frequency of the category. Unlike bar chart, Histogram does not have any spacing between the bars and is used to show distribution of numerical data such as the count of athletic facilities with different precincts in our dataset. The following are some examples of Histograms during this study:

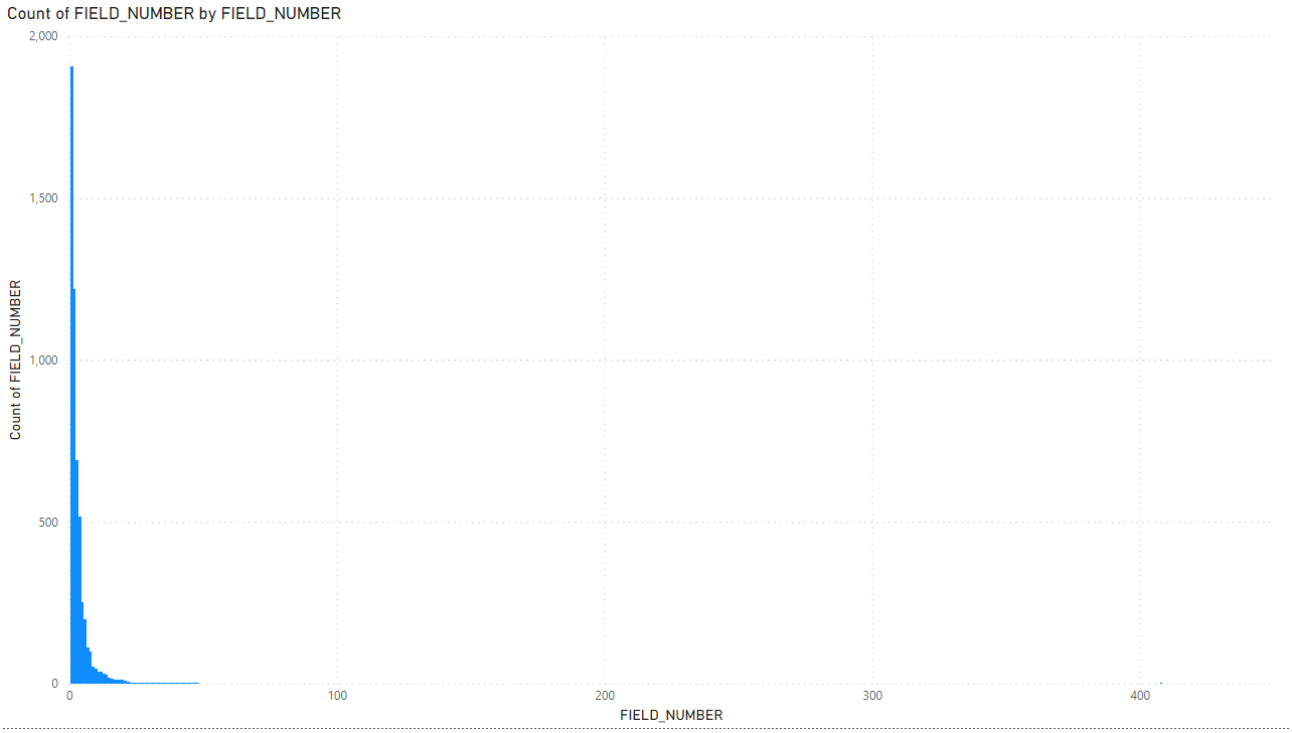


Figure 3.13: The above figure depicts a Histogram Chart showing the distribution of the fields according to *FIELD_NUMBER* without any form of grouping.

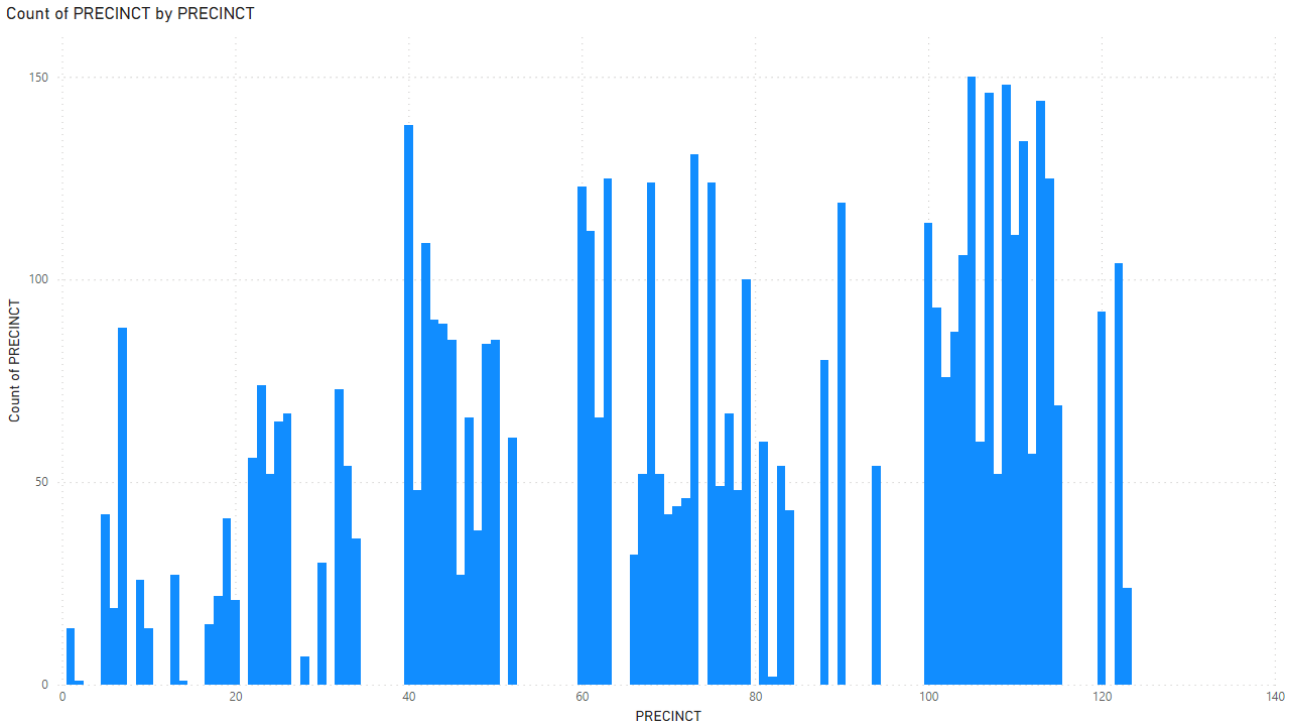


Figure 3.14: The above figure depicts a Histogram Chart showing the distribution of fields according to *PRECINCT*. Precinct refers to the are surrounding a facility or grounds. Most facilities have a Precinct in the range of $40 - 110$.

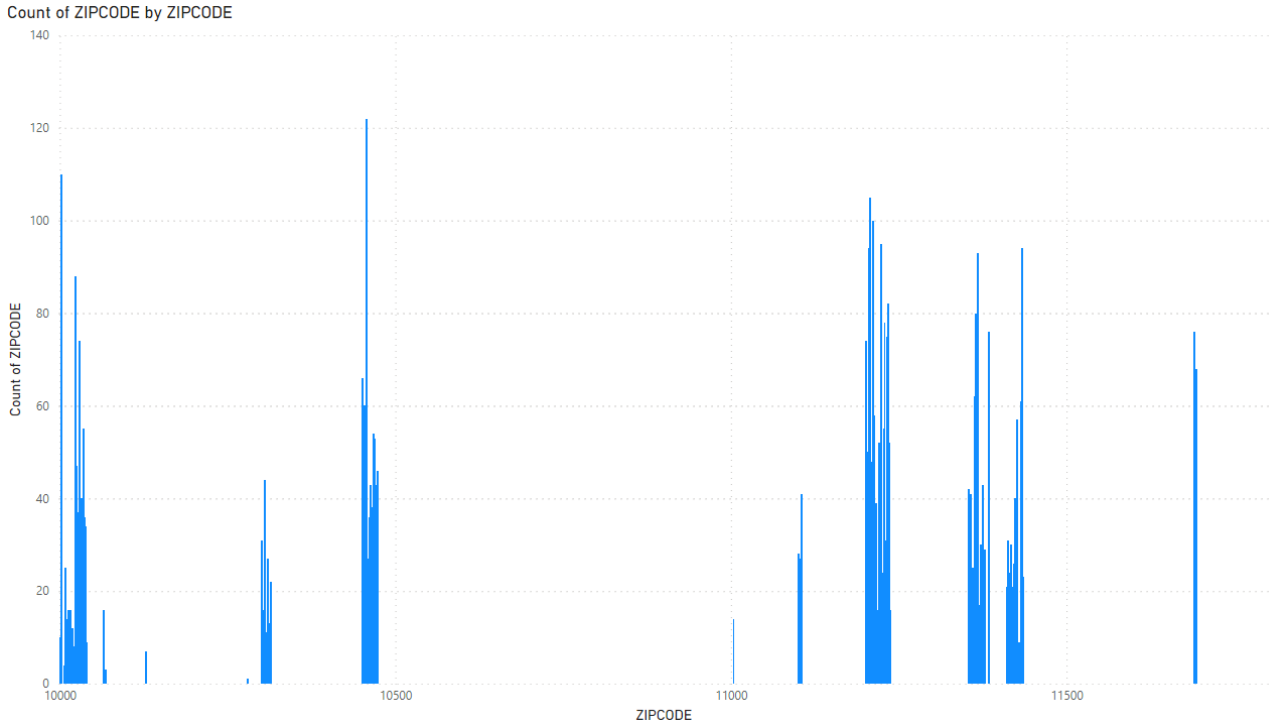


Figure 3.15: The above figure depicts a Histogram Chart showing the distribution of the fields in an are by virtue of their according to *ZIPCODE*.

3.1.4 Line Charts

Line chart is a form of graph where data is visualized in the form of points connected with straight lines. It can be used for both categorical and numerical data. The following are some examples of Line charts during this study:

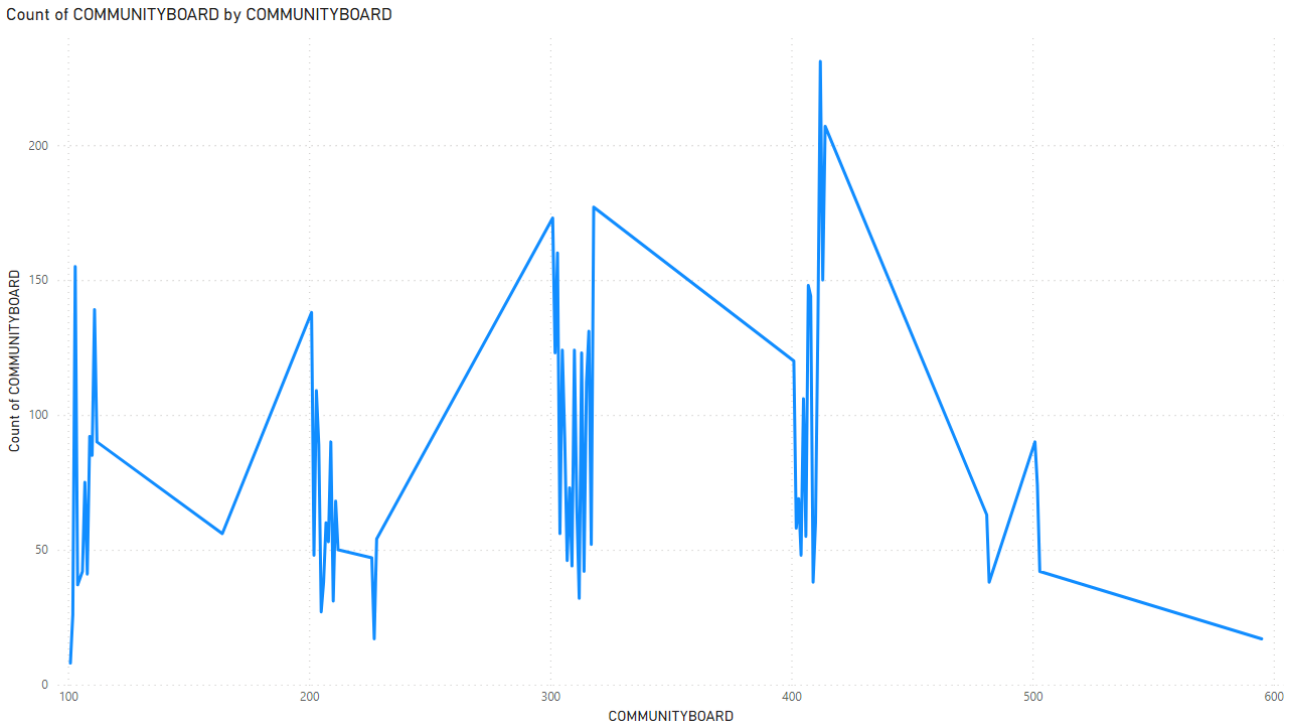


Figure 3.16: The above figure exhibits a Line Chart showing distribution of fields under each *COMMUNITYBOARD*.

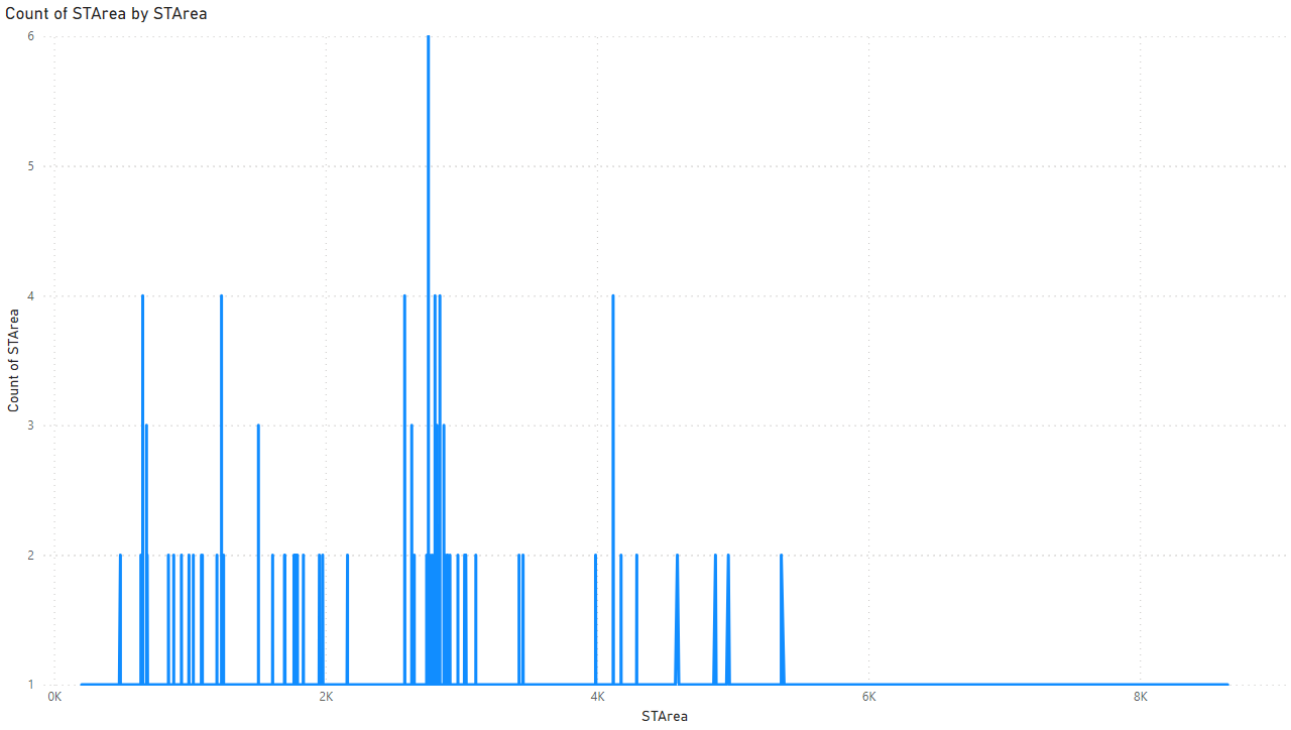


Figure 3.17: The above figure exhibits a Line Chart showing distribution of *STArea*. We can observe a spike in the region of 2700-3000, which is also about the same as the mean *STArea*.

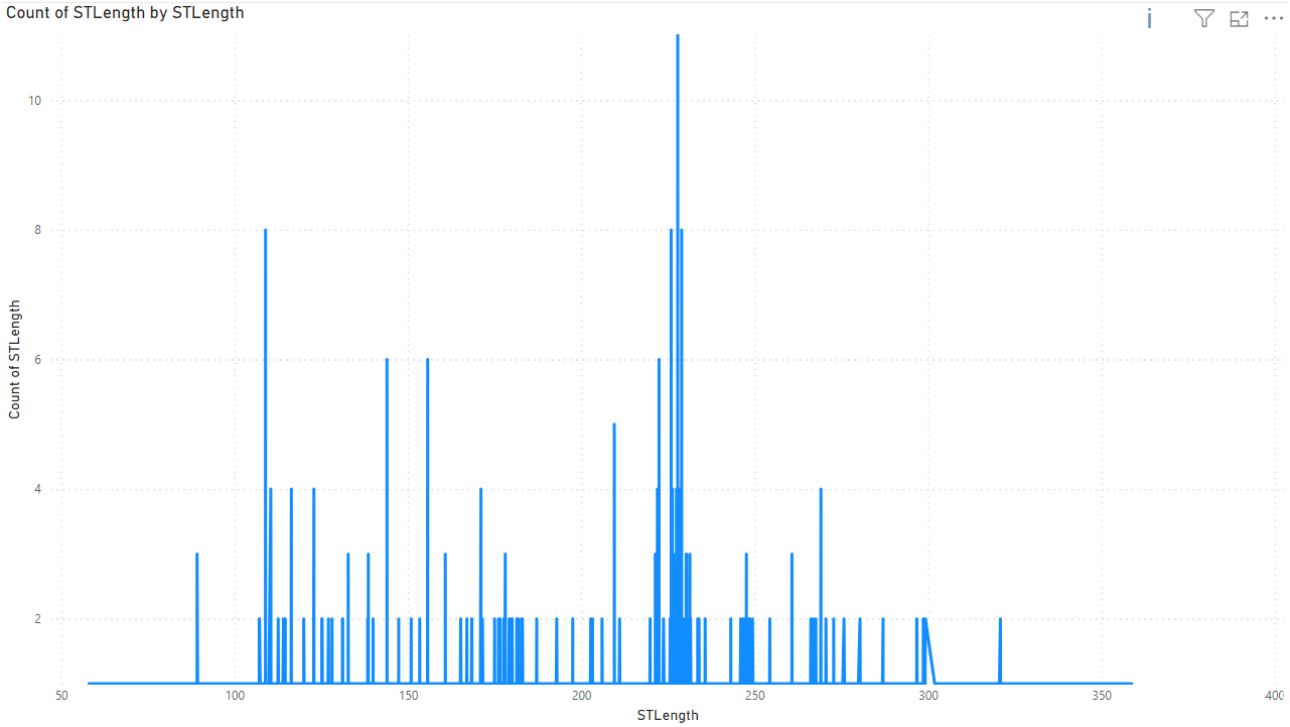


Figure 3.18: The above figure depicts a Line Chart showing the distribution of *STLength*. We can observe a spike in the region of 220-230, which is also about the same as the mean *STLength*.

3.1.5 WaterFall Charts

A waterfall chart is a graphical tool primarily used to show the collective influence of successive positive and negative variables on an initial starting point. The following are some examples of Waterfall charts during this study:

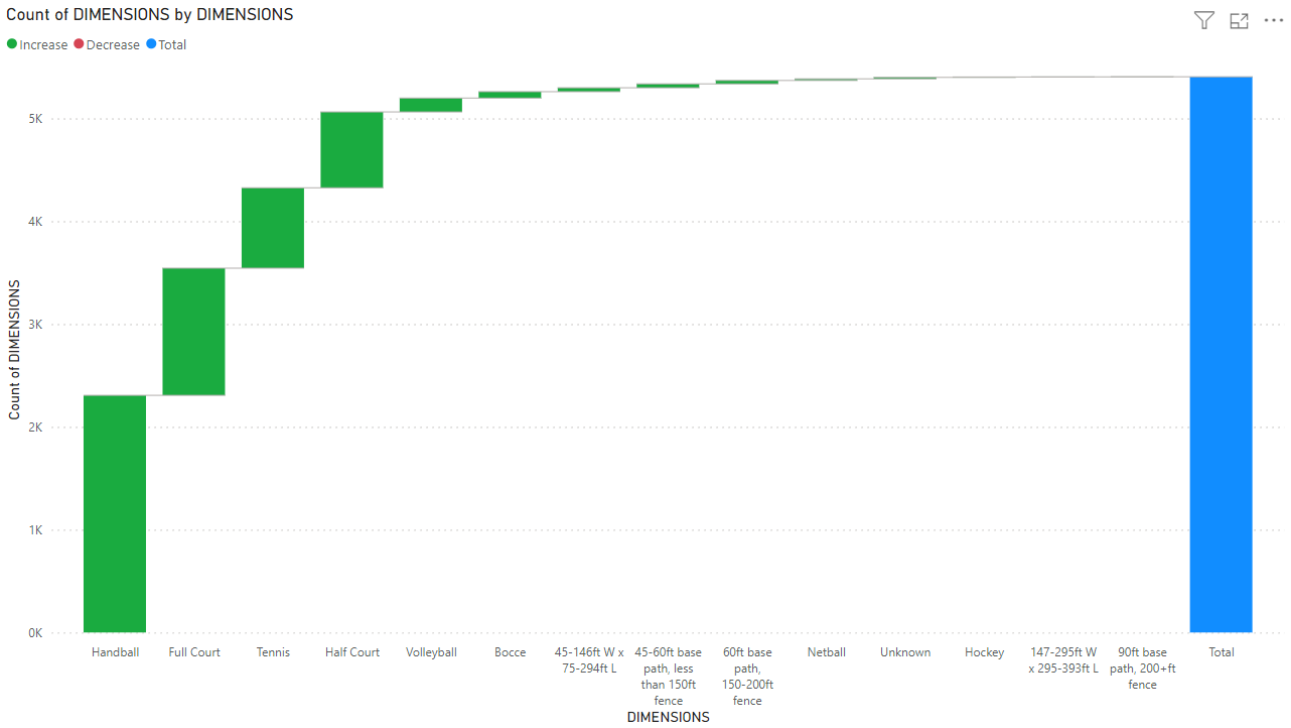


Figure 3.19: The above figure depicts a Waterfall Chart showing the distribution of various *DIMENSIONS*.

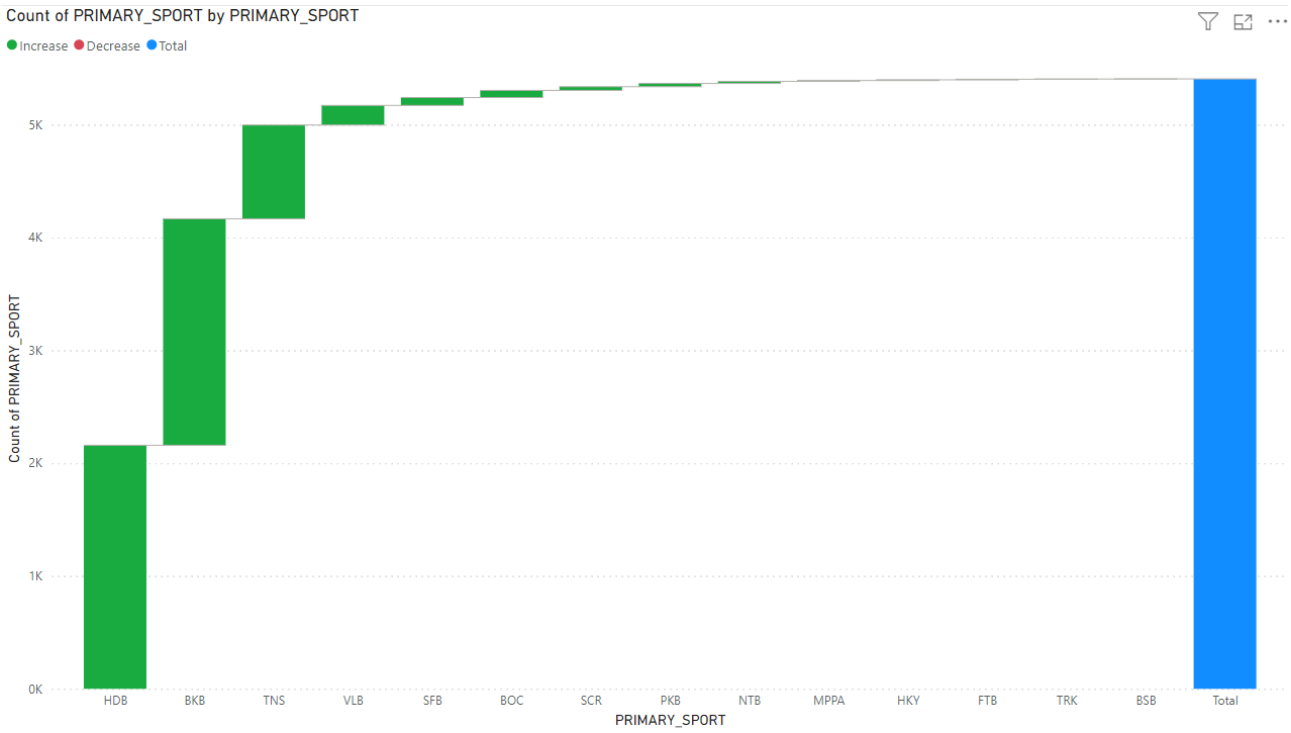


Figure 3.20: The above figure depicts a Waterfall Chart showing the distribution of the various *PRIMARY_SPORT*.

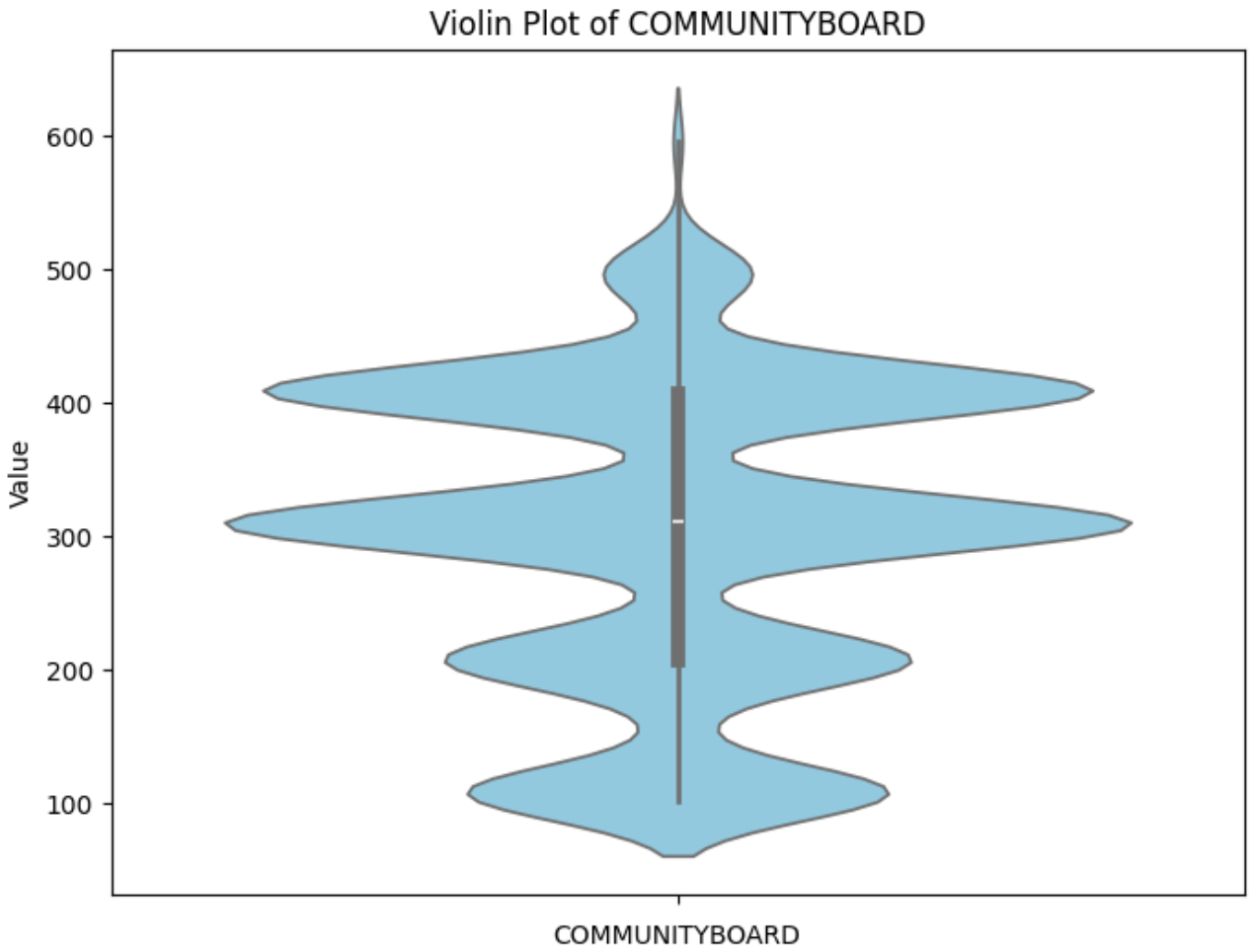


Figure 3.21: The above figure exhibits Violin Plot of Cleaned Data for *COMMUNITYBOARD* Column.

3.1.6 Violin Plots

A violin plot is another method of plotting numeric data, roughly a combination of a box plot and a kernel density plot. They are used to describe the distribution of the data on different categories. The following are some examples of Violin charts during this study:

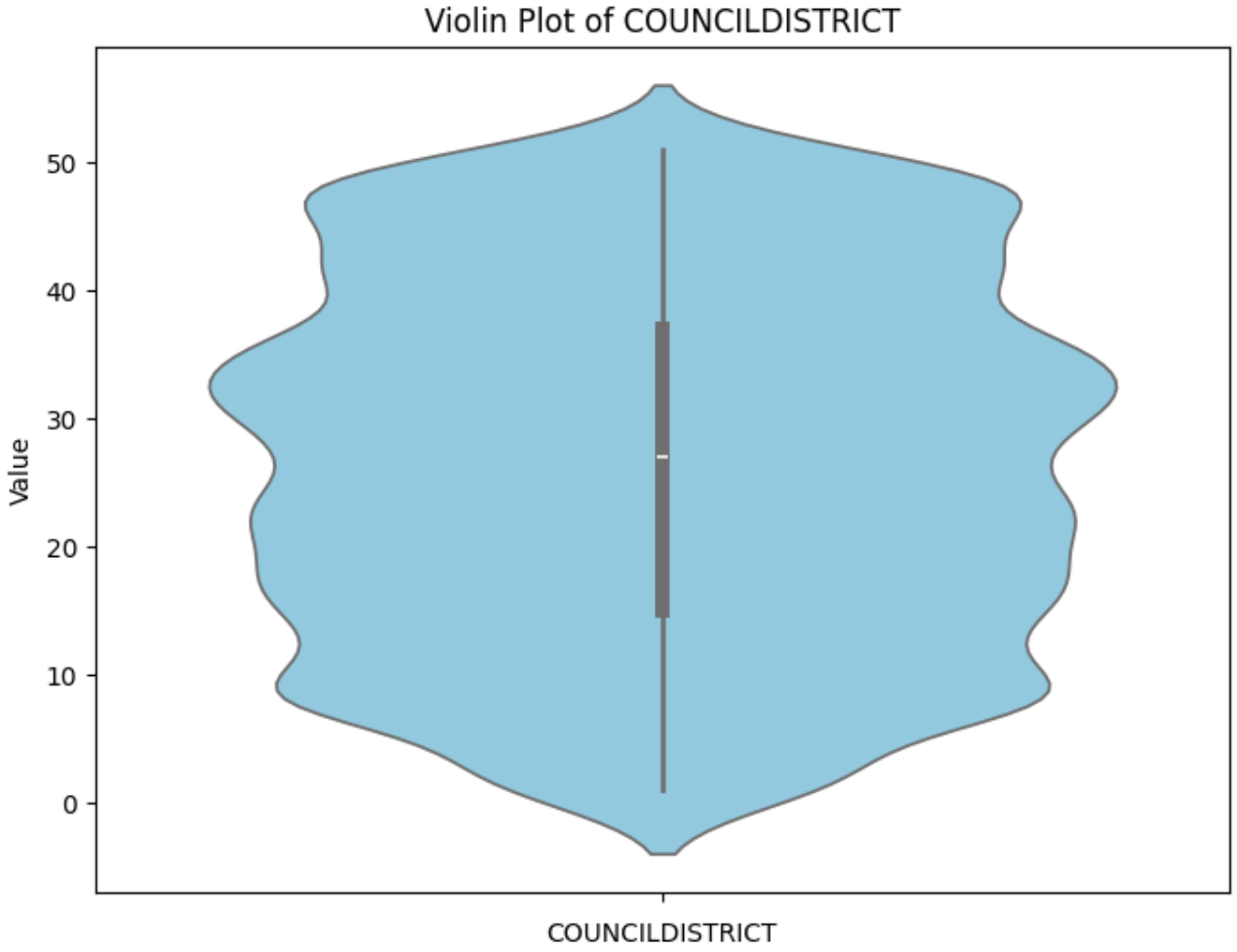


Figure 3.22: The above figure depicts the Violin Plot of the Cleaned Data for the *COUNCILDISTRICT* Column.

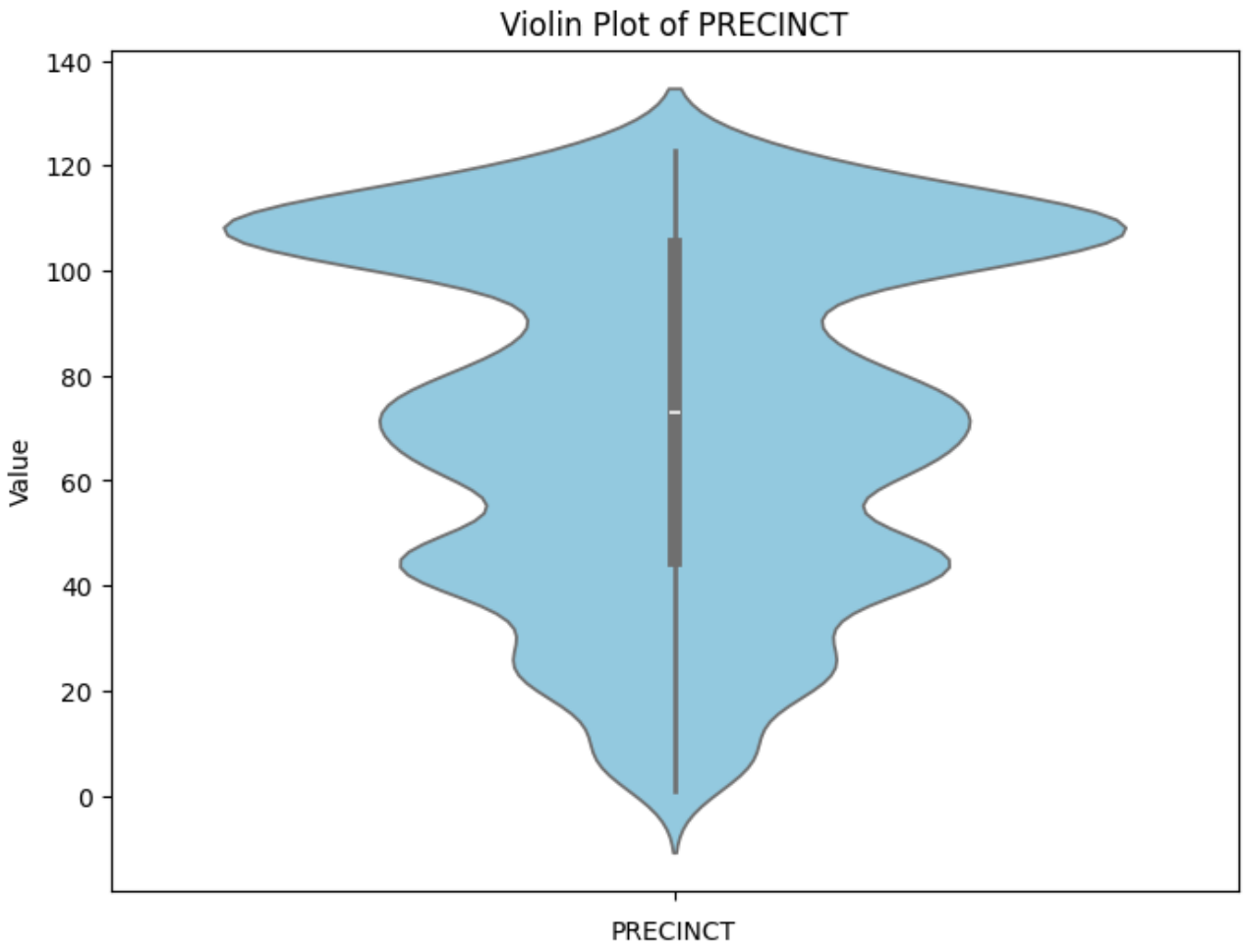


Figure 3.23: The above figure depicts the Violin Plot of the Cleaned Data for the *PRECINCT* Column.

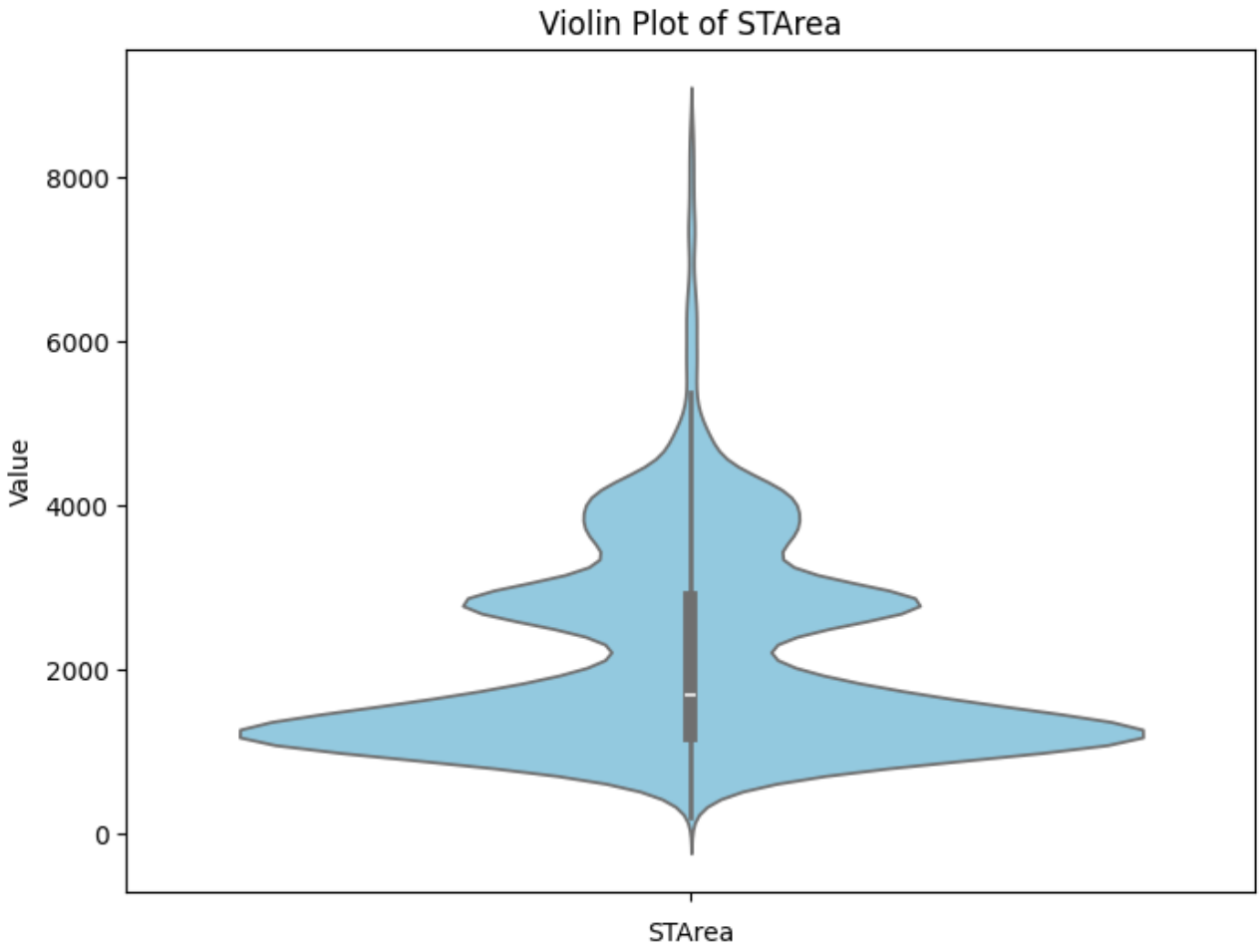


Figure 3.24: The above figure depicts the Violin Plot of the Cleaned Data for the *STArea* Column.

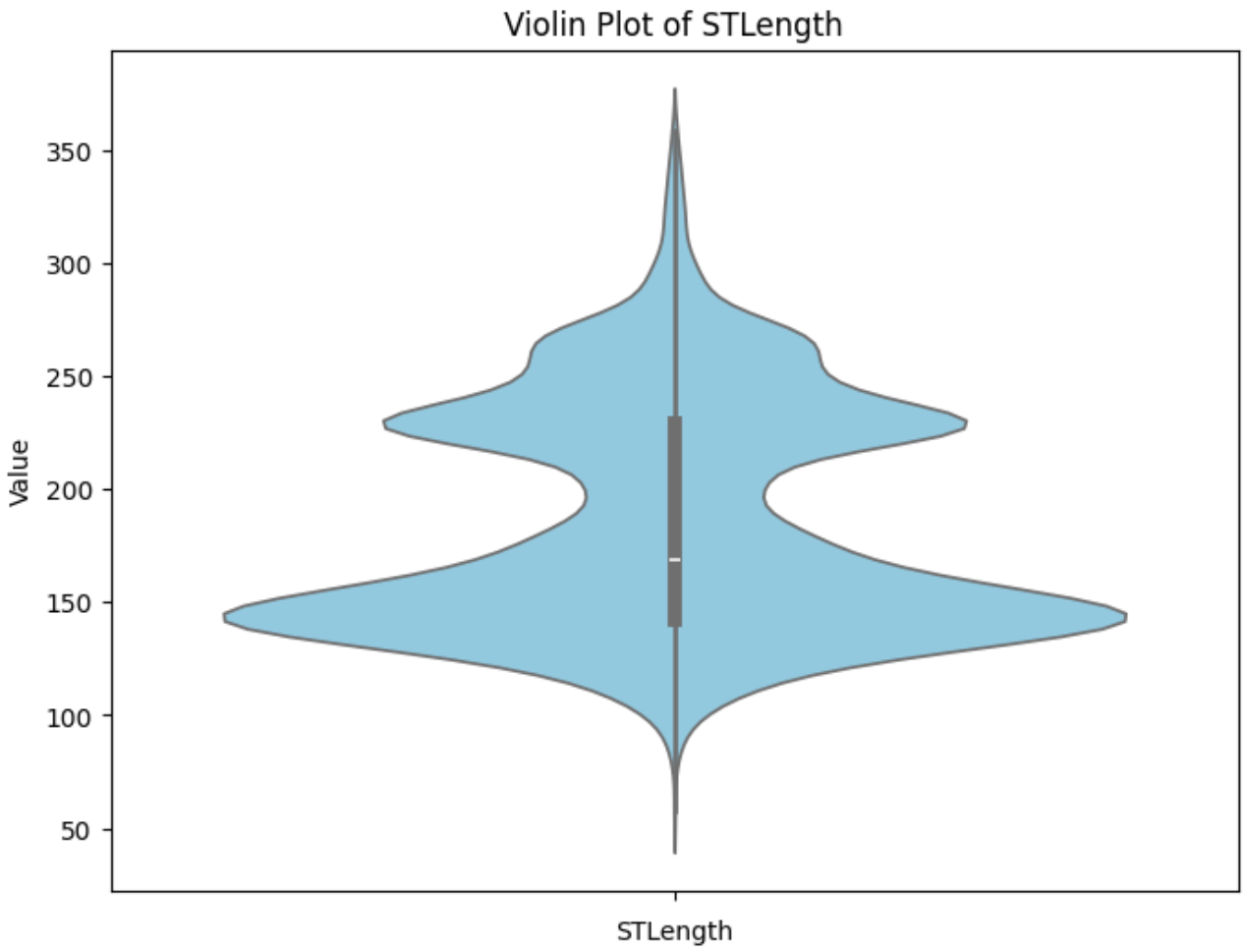


Figure 3.25: The above figure depicts the Violin Plot of the Cleaned Data for the *STLength* Column.

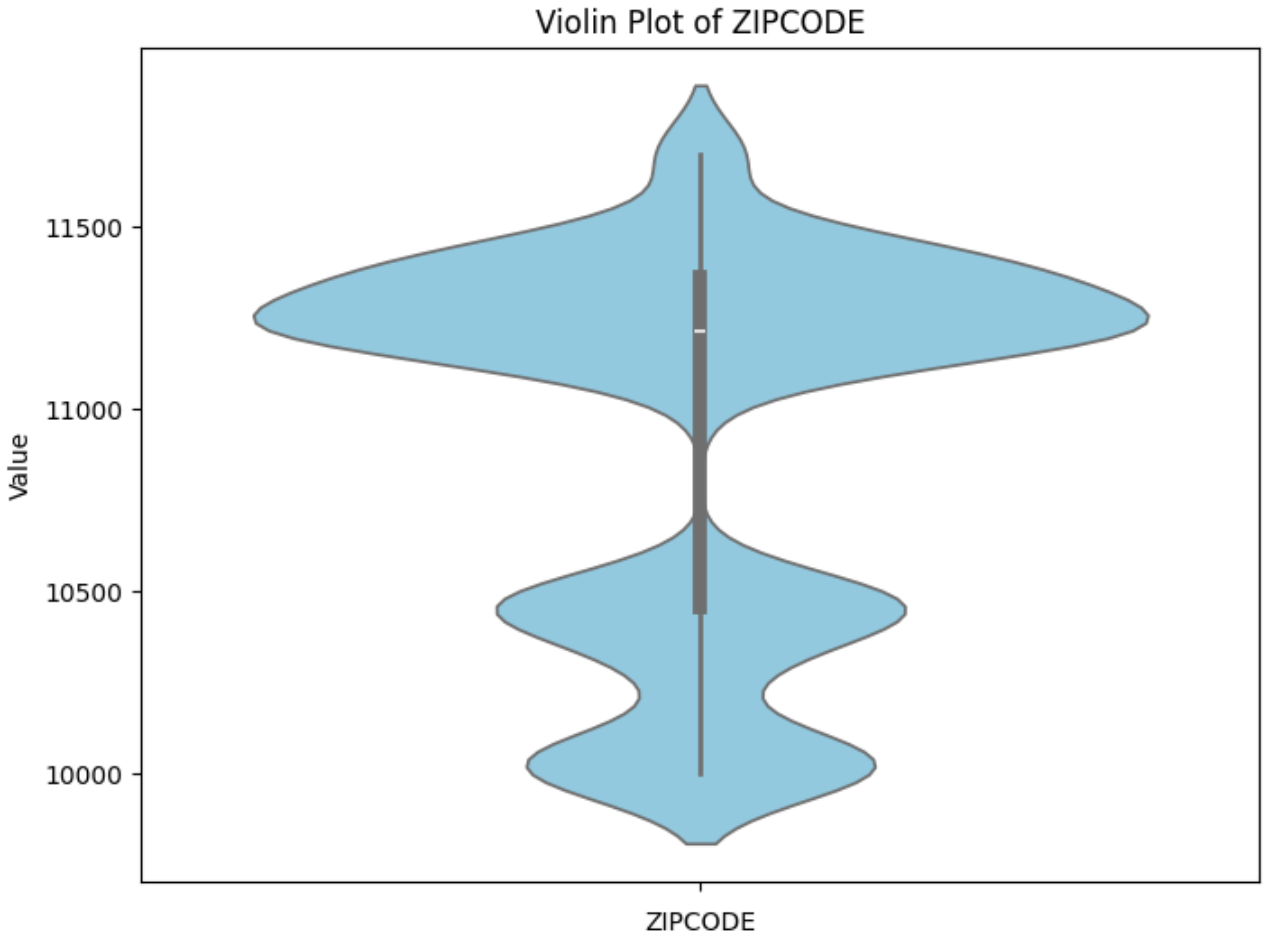


Figure 3.26: The above figure depicts the Violin Plot of the Cleaned Data for the *ZIPCODE* Column.

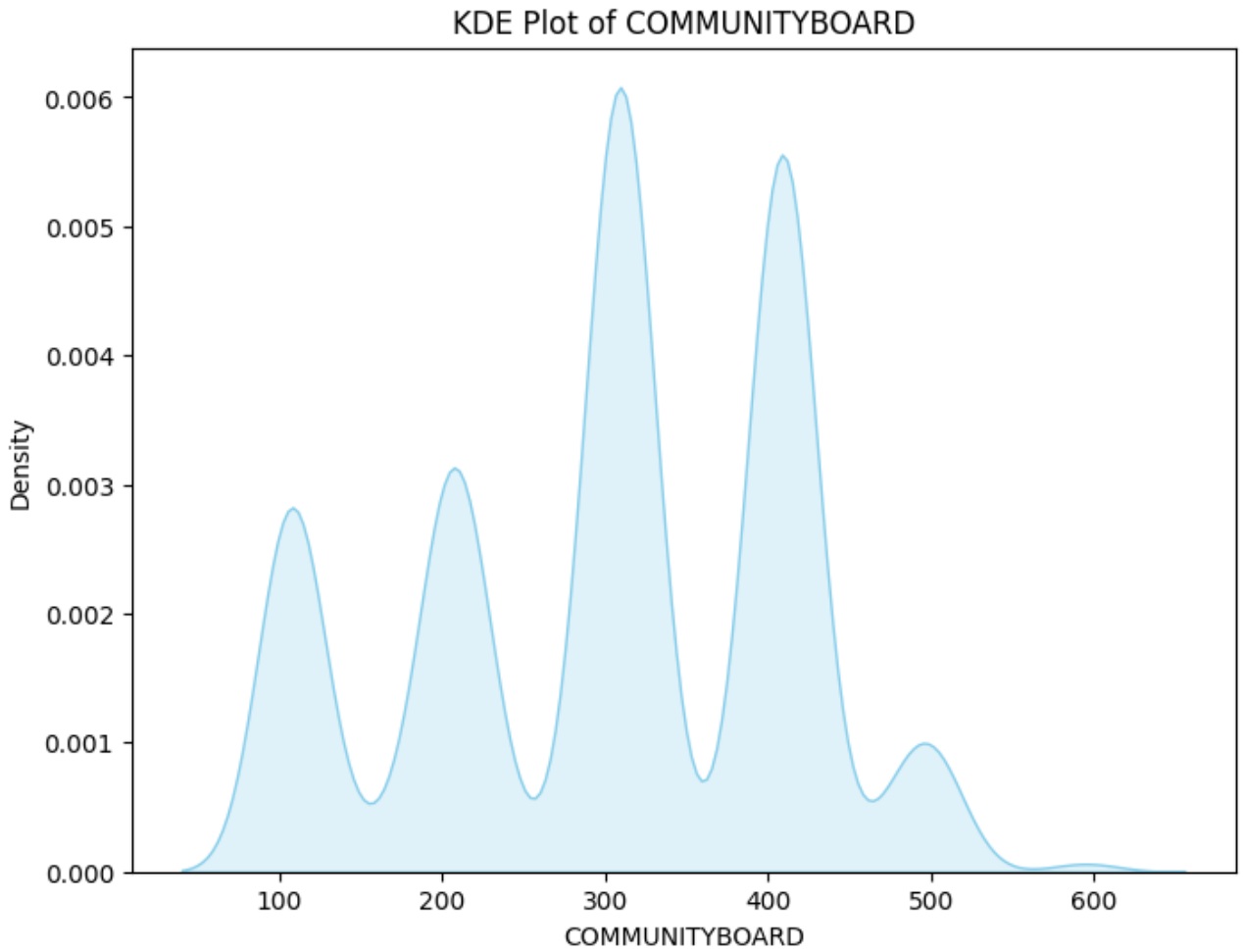


Figure 3.27: The above figure depicts the KDE Plot of the Cleaned Data for the *COMMUNITYBOARD* Column.

3.1.7 KDE Plots

KDE plots or Kernel Density Estimate Charts is used to find density function of a given distribution. It represents data using a continuous probability density curve. The following are some examples of KDE charts during this study:

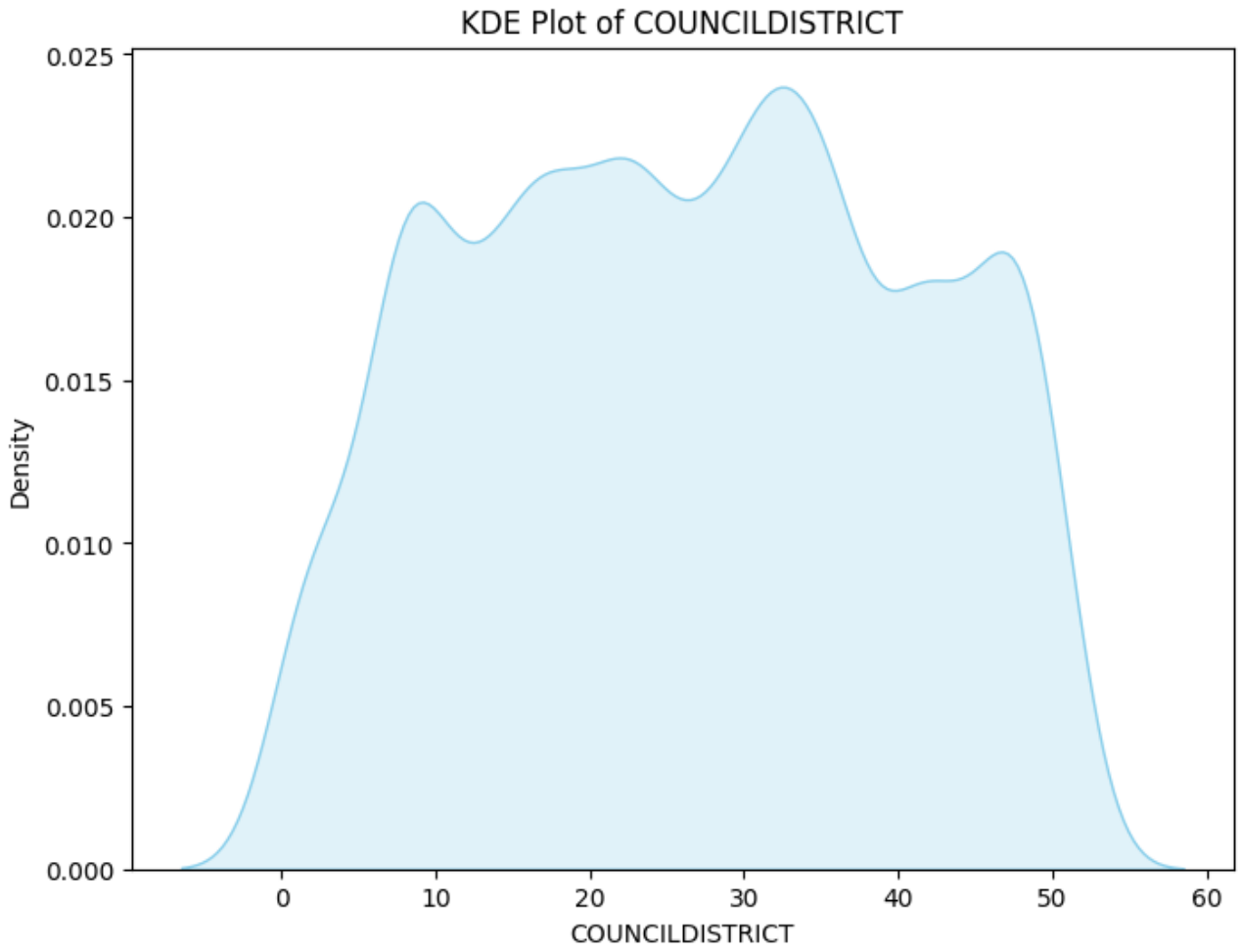


Figure 3.28: The above figure depicts the KDE Plot of the Cleaned Data for the *COUNCILDISTRICT* Column.

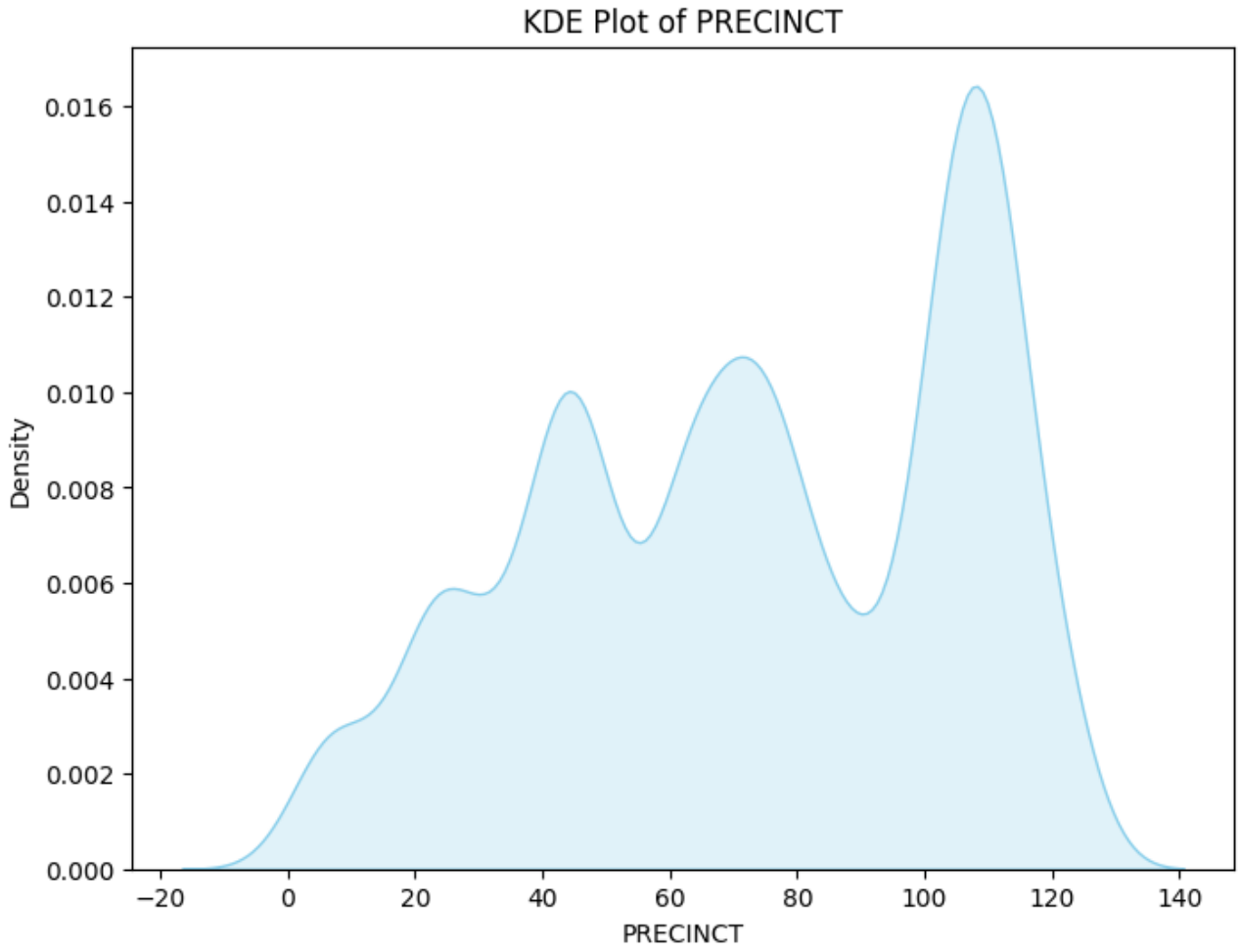


Figure 3.29: The above figure depicts the KDE Plot of the Cleaned Data for the *PRECINCT* Column.

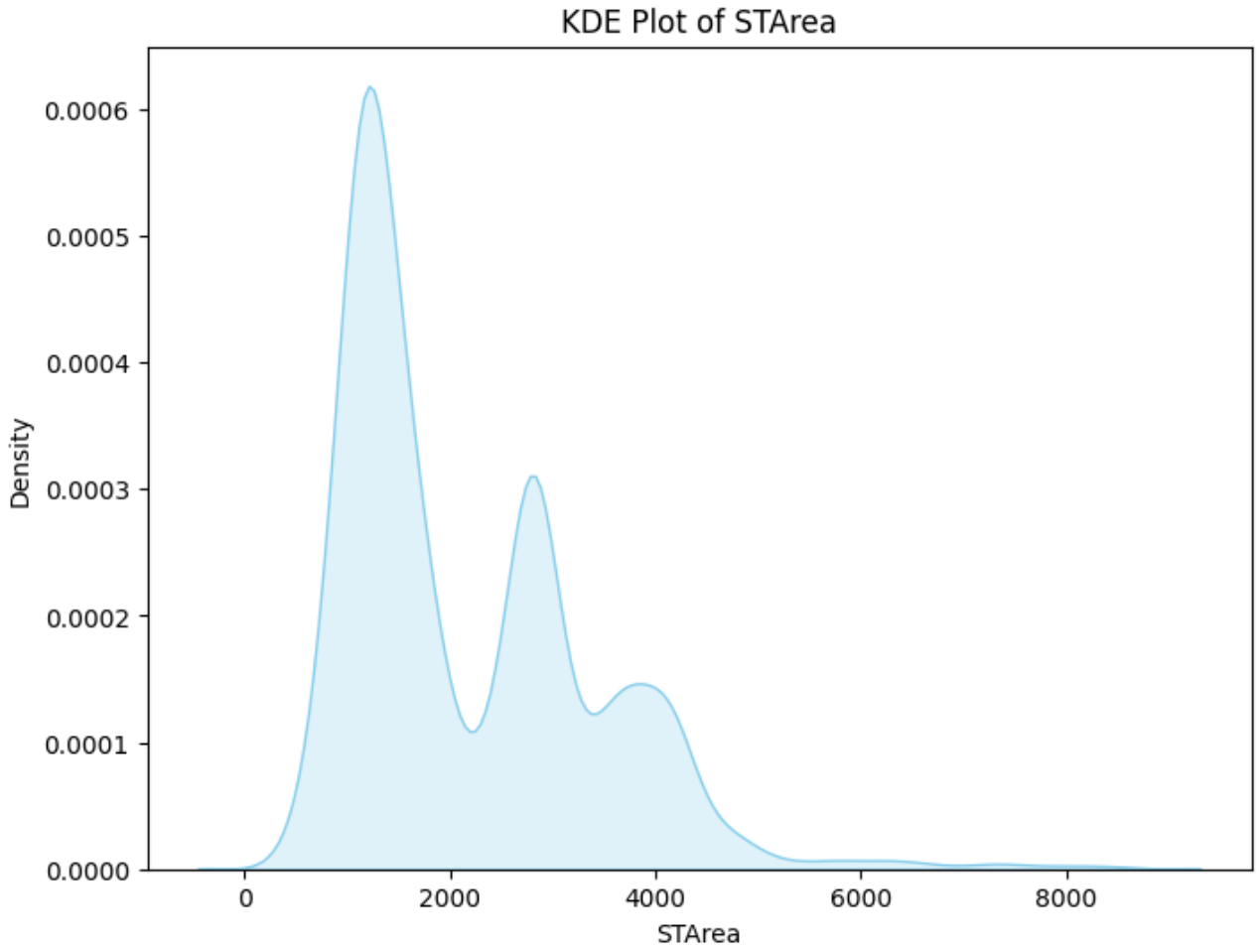


Figure 3.30: The above figure depicts the KDE Plot of the Cleaned Data for the *STArea* Column.

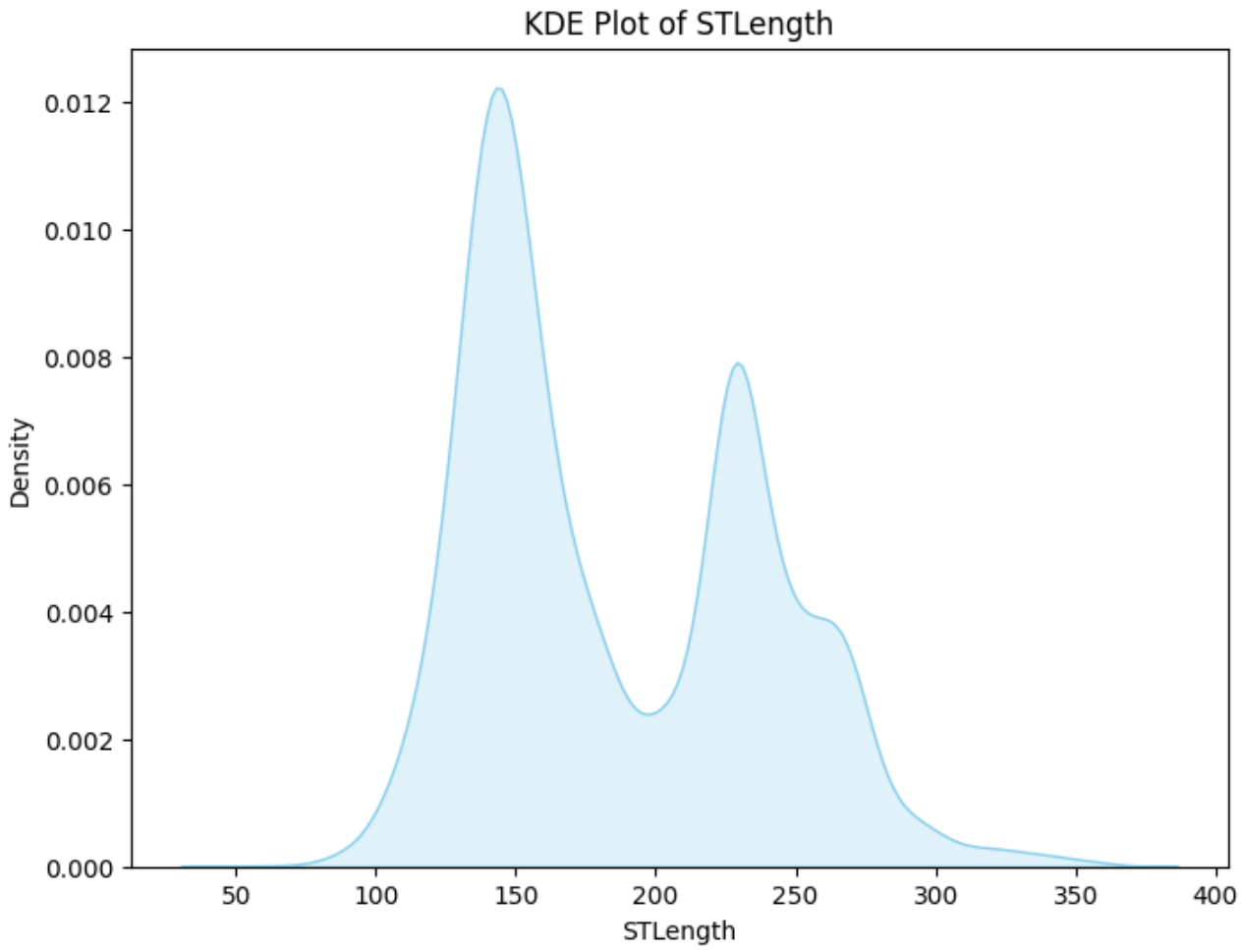


Figure 3.31: The above figure depicts the KDE Plot of the Cleaned Data for the *STLength* Column.

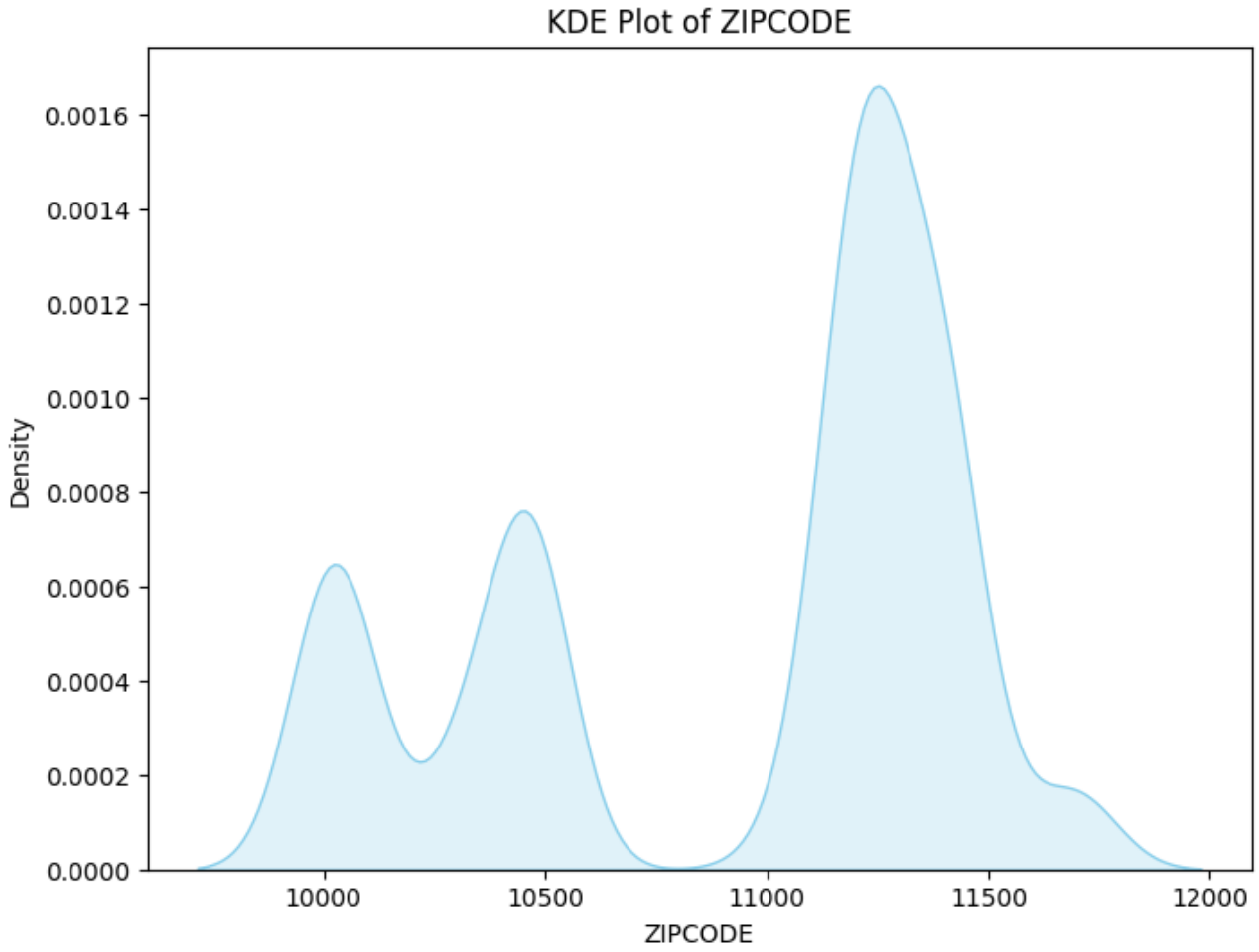


Figure 3.32: The above figure depicts the KDE Plot of the Cleaned Data for the *ZIPCODE* Column.

3.2 Multivariate analysis

In multivariate analysis, we will analyze multiple variables in the dataset together. The goal of the multivariate analysis is to understand the distribution of variables and examine their relational properties and dependencies. The main techniques involved in this are:

- Scatter Plots
- Stacked Bar Charts
- Grouped Bar Charts
- Tree Maps
- Point Plots
- Correlation Map
- Pair Plots
- Joint Plot

3.2.1 Scatter Plots

Scatter plot uses dots on a Cartesian plane to represent values for two different numerical data. They are used to observe and deduce relationships between the two variables. The following are some examples of Scatter charts during this study:

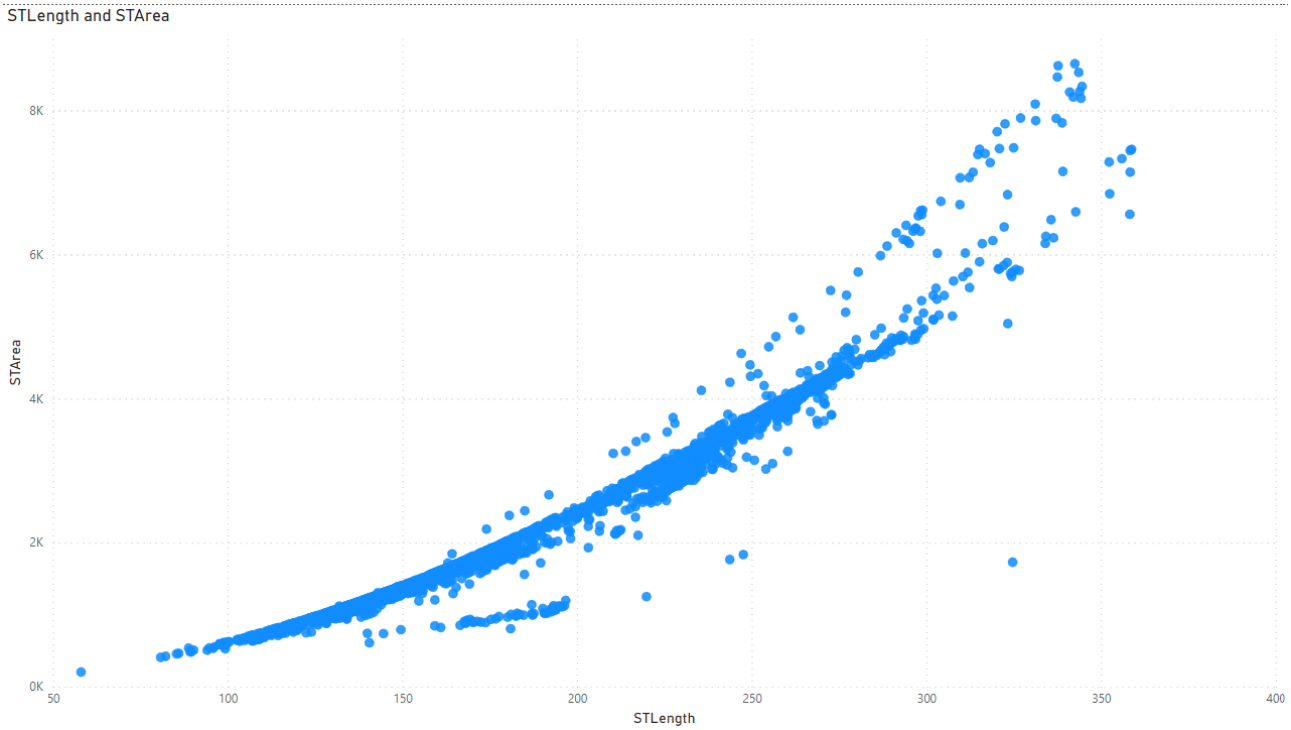


Figure 3.33: The above figure depicts a Scatter Plot showing the plot between *STLength* and *STArea*. It helps to ascertain any form of relation they may have. For example, by referring to the above chart we can infer a quadratic relation between *STLength* and *STArea*.

3.2.2 Stacked Bar Charts

Stacked Bar Chart is a form of the bar chart. It uses the method of displaying the data of multiple related or similar variables with a single bar chart by stacking them on top each other. The following are some examples of Stacked Bar Chart during this study:

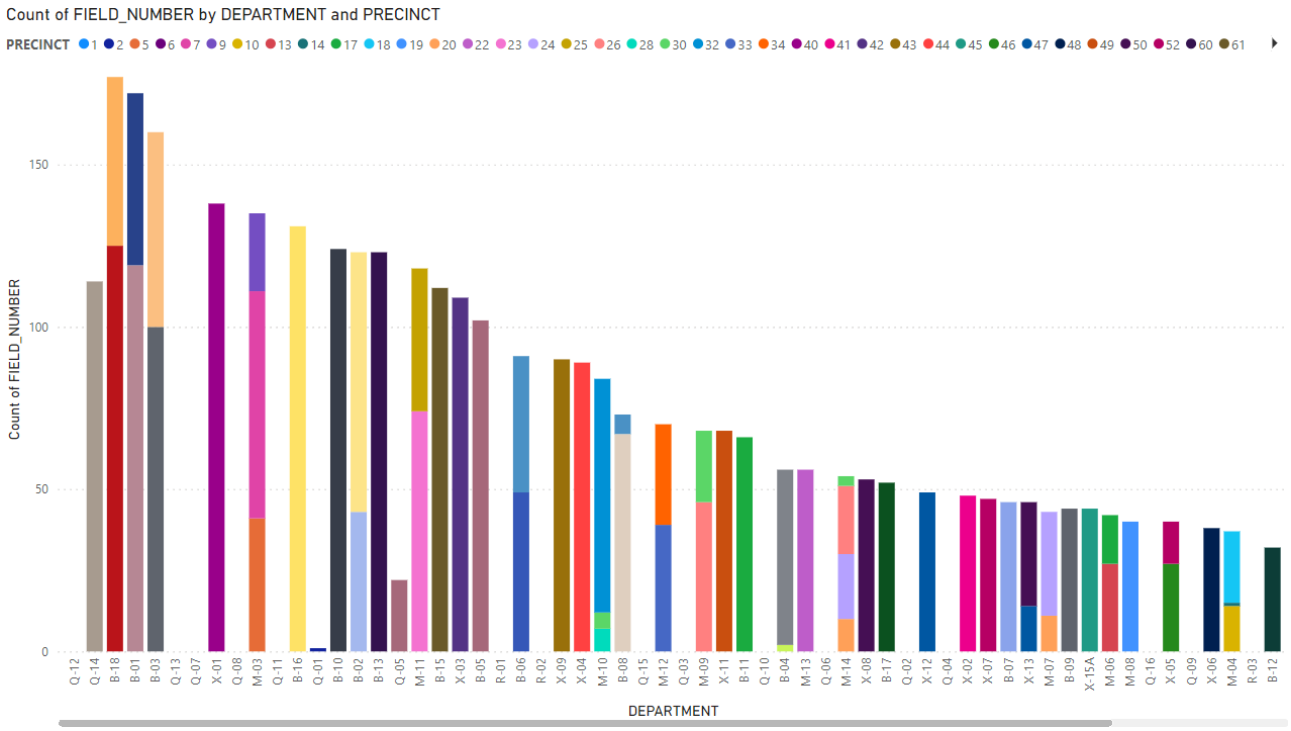


Figure 3.34: The above figure depicts a Stacked Bar Chart showing the distribution of fields of various *PRECINCT* in different *DEPARMENT*.

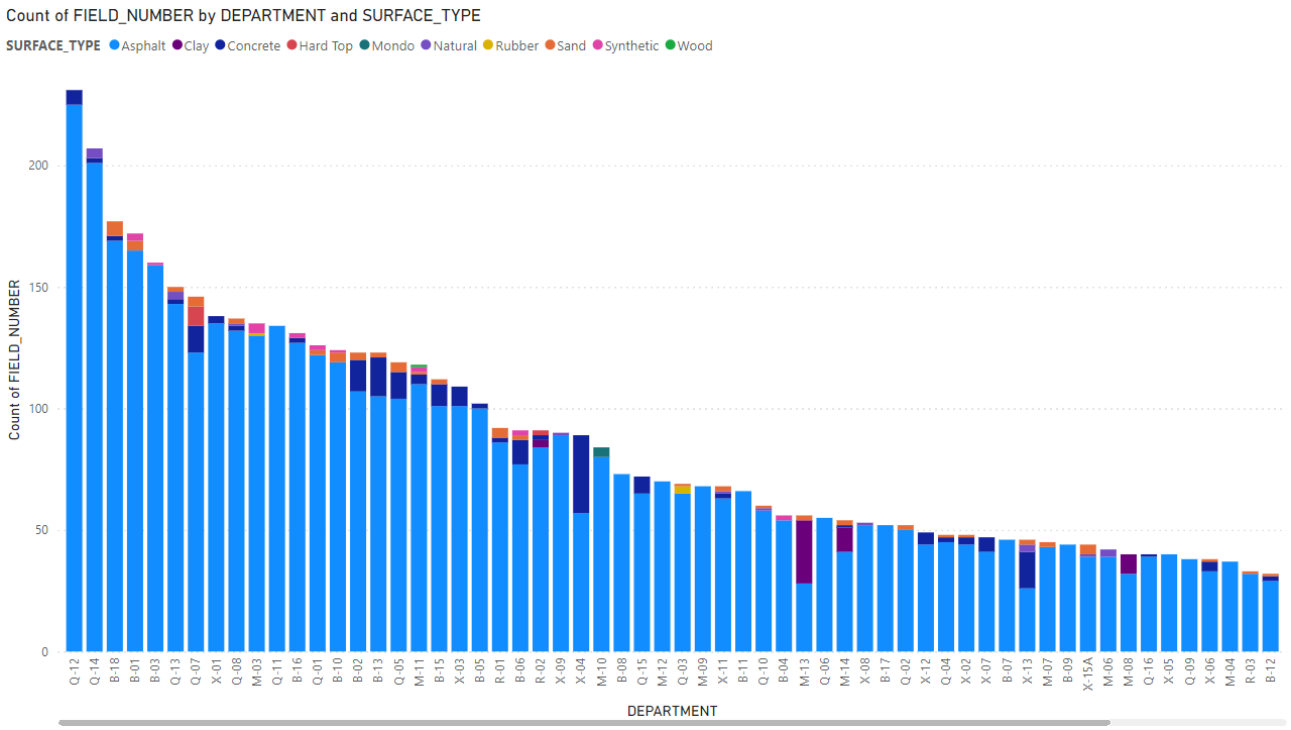


Figure 3.35: The above figure depicts a Stacked Bar Chart showing the distribution of fields of various *SURFACE_TYPE* in different *DEPARMENT*. Once again, it shows the huge ratio of field grounds made of Asphalt.

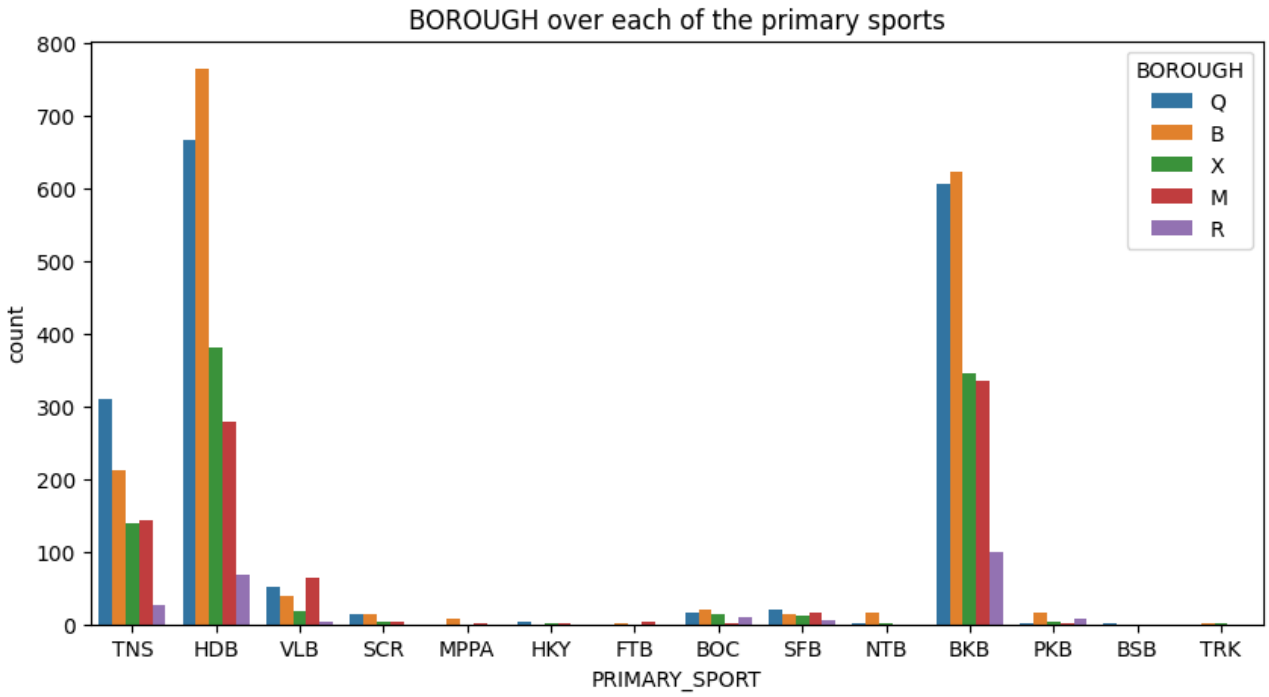


Figure 3.36: The above figure depicts a Grouped Bar Chart showing the distribution of *BOROUGH* over each *PRIMARY_SPORT*.

3.2.3 Grouped Bar Charts

Grouped Bar Chart is a form of the bar chart. It uses the method of displaying the data of multiple related or similar variables in a plot by grouping them under common values. The following are some examples of Grouped Bar Chart during this study:

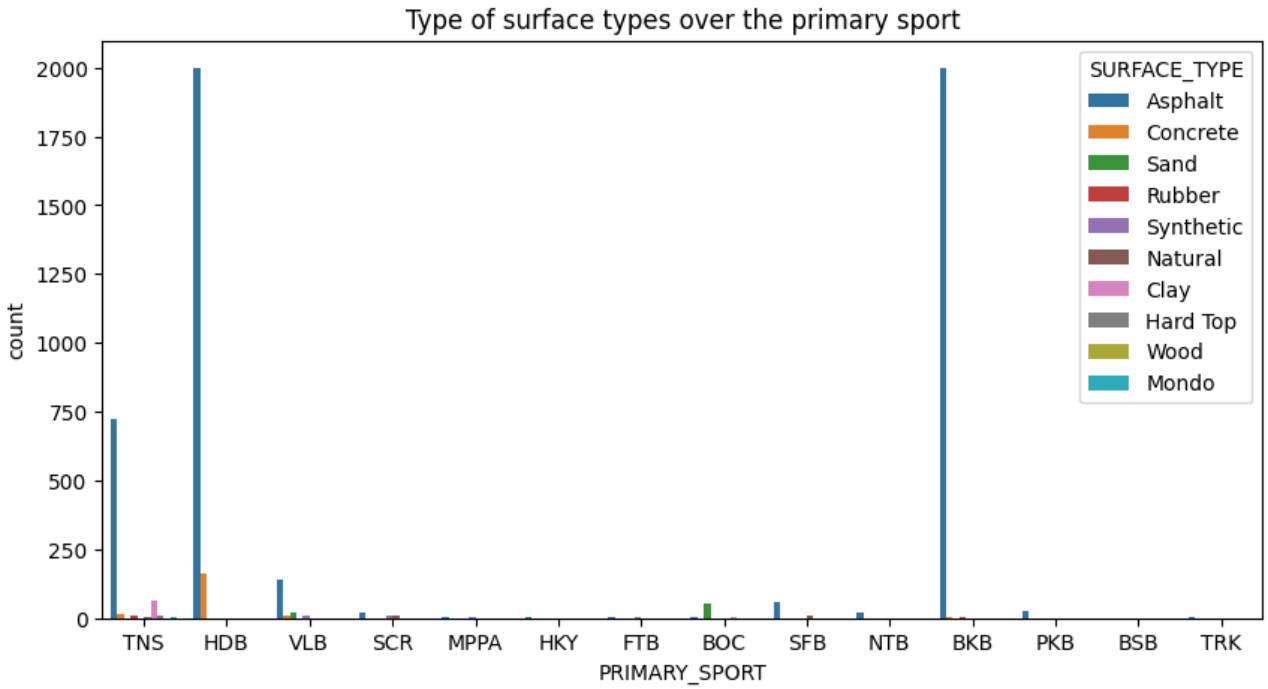


Figure 3.37: The above figure depicts a Grouped Bar Chart showing the distribution of *SURFACE_TYPE* over each *PRIMARY_SPORT*.

3.2.4 Tree Maps

A treemap is a data visualization type that displays hierarchical data as a set of nested rectangles. Each branch of the hierarchy is presented as a rectangle that is divided into smaller rectangles for each sub-branch. The rectangle size corresponds to the data being measured. The following are some examples of Tree Maps during this study:

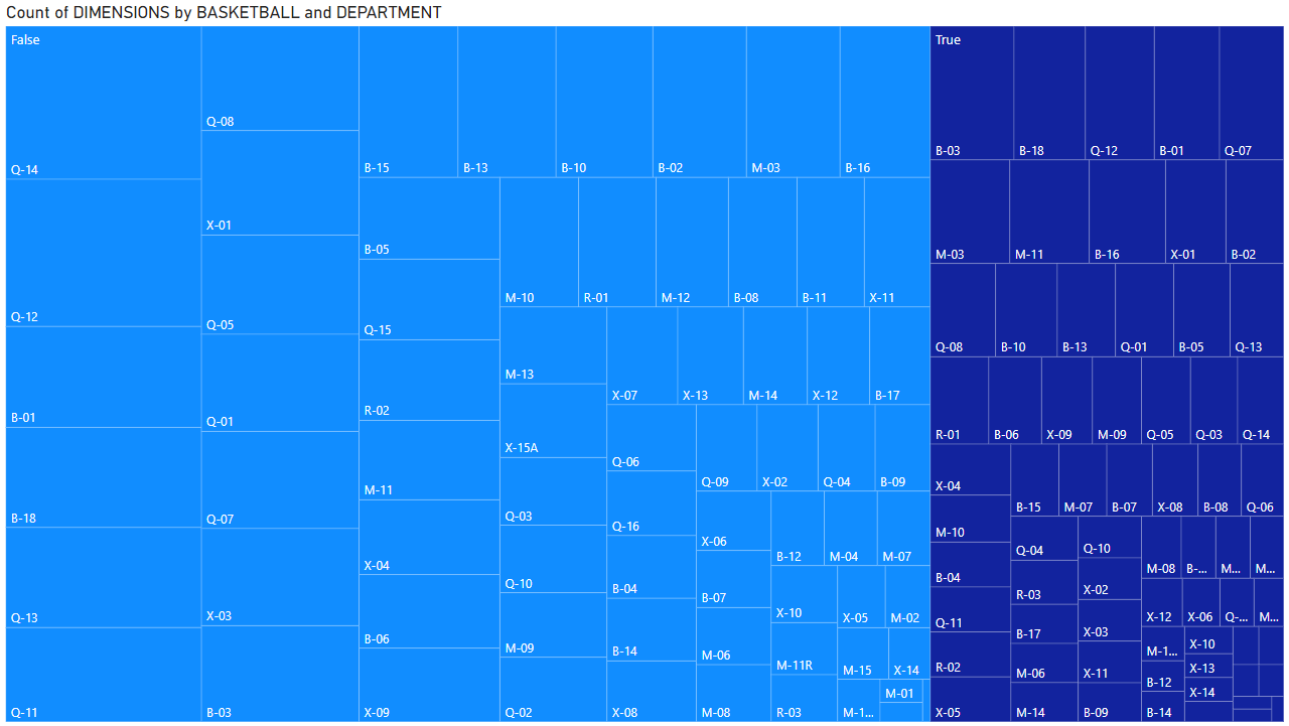


Figure 3.38: The above figure depicts a Tree Chart showing the distribution of availability of *BASKETBALL* fields of different *DIMENSIONS* in various *DEPARTMENTS*.

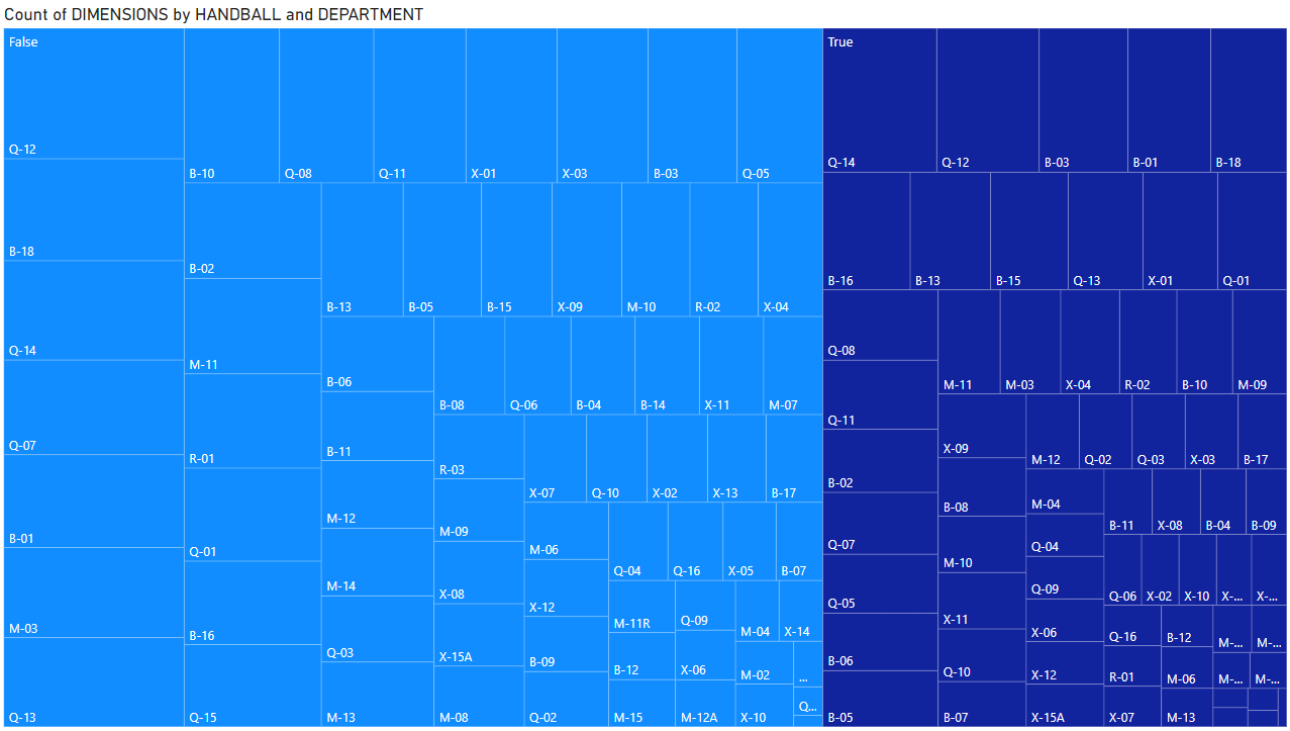


Figure 3.39: The above figure depicts a Tree Chart showing the distribution of availability of *HANDBALL* fields of different *DIMENSIONS* in various *DEPARTMENTS*.

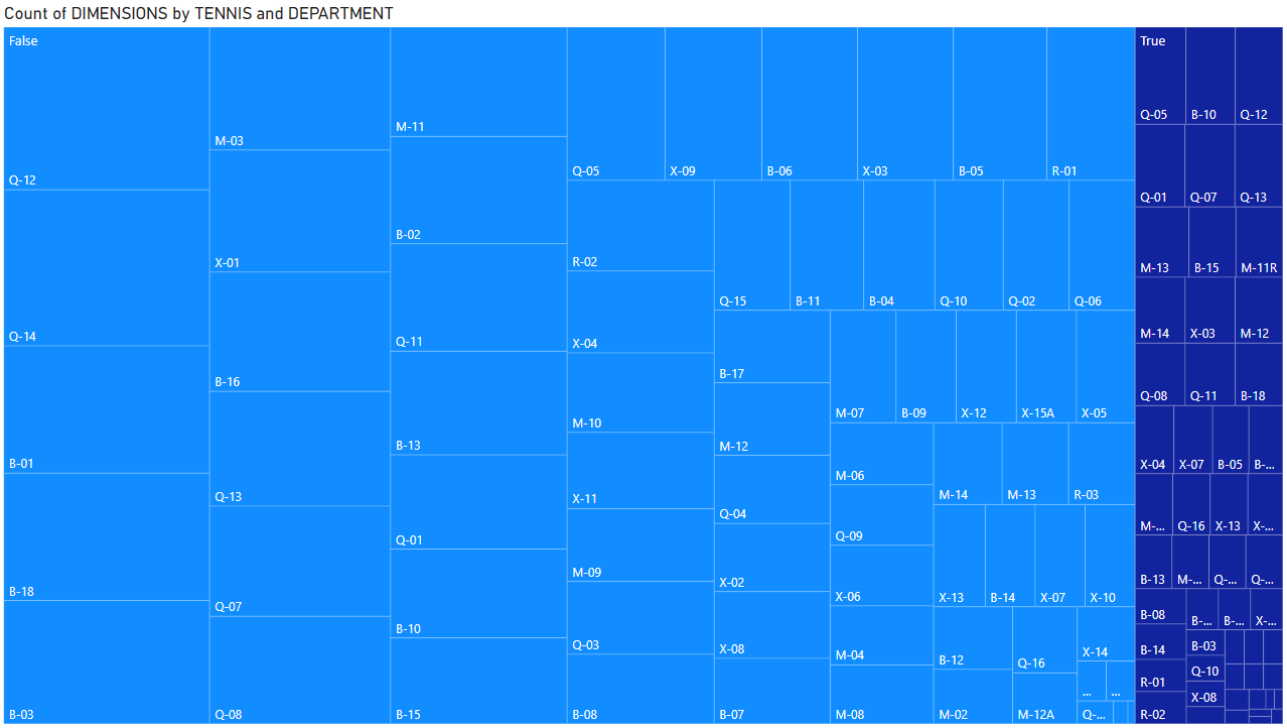


Figure 3.40: The above figure depicts a Tree Chart showing the distribution of availability of *TENNIS* fields of different *DIMENSIONS* in various *DEPARTMENTS*.

3.2.5 Point Plots

Point plots provide use when we want to estimate measures of central tendency for numeric variables.

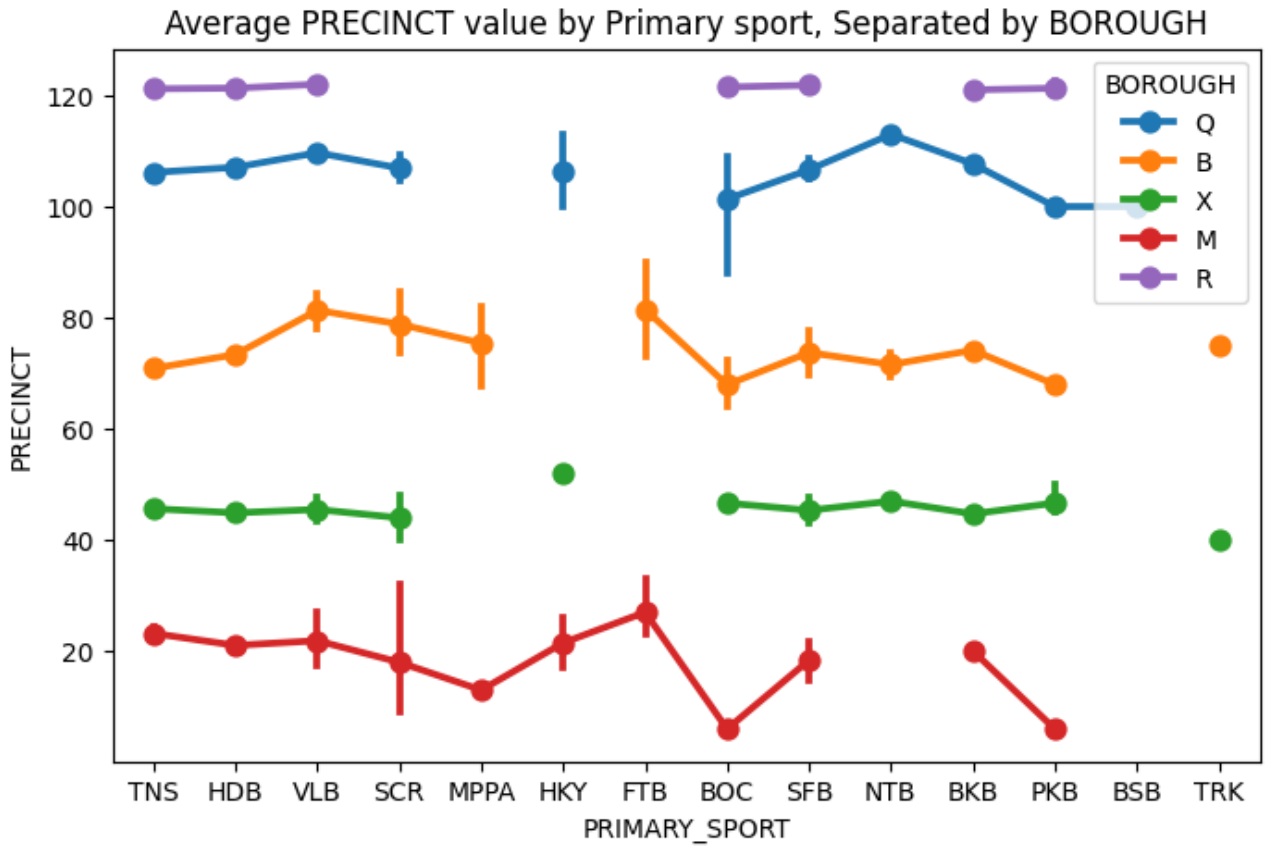


Figure 3.41: The above figure depicts a Point Plot showing the *PRECINCT* of various *PRIMARY_SPORT* separated by *BOROUGH*.

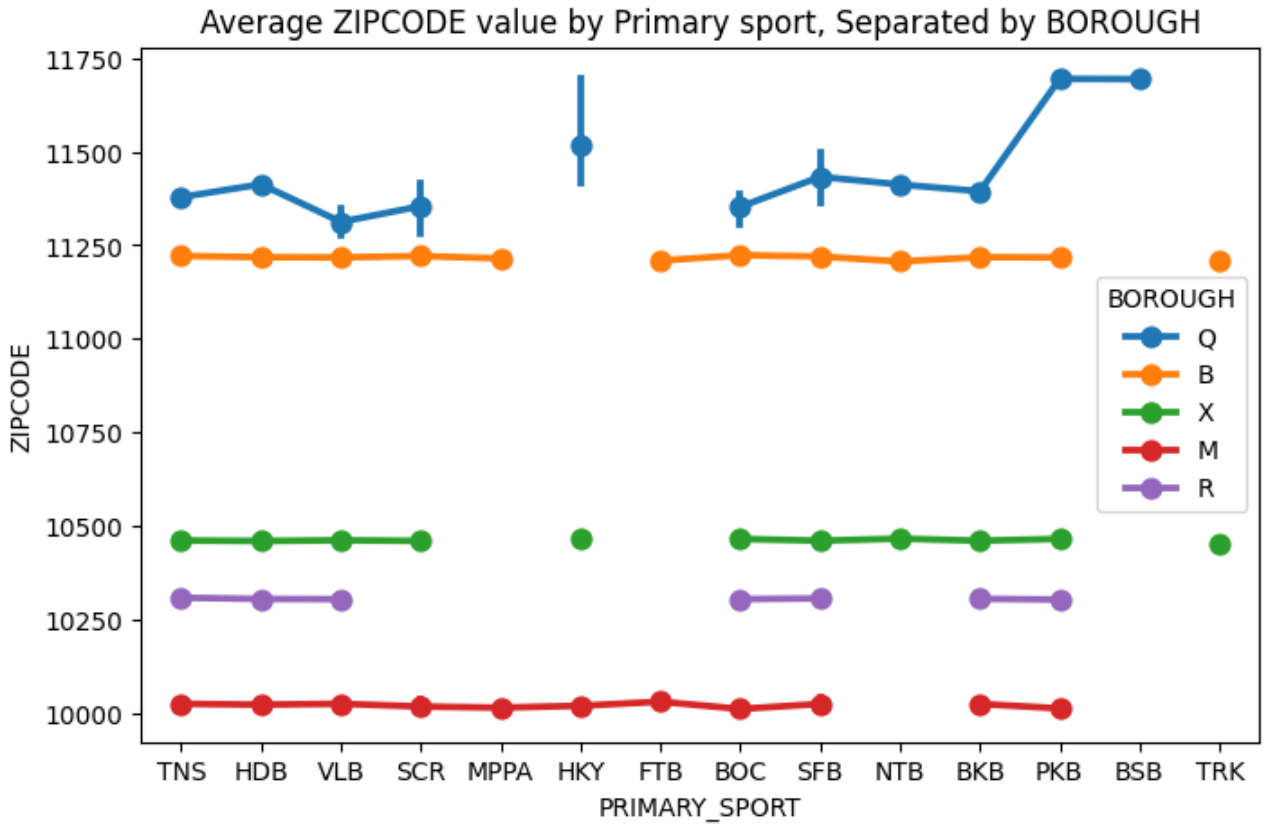


Figure 3.42: The above figure depicts a Point Plot showing the *ZIPCODE* of various *PRIMARY_SPORT* separated by *BOROUGH*.

3.2.6 Correlation Map

The Correlation Map visualizes the calculated correlation coefficients for more than two variables. You can quickly determine whether there is a relationship between the variables or not.

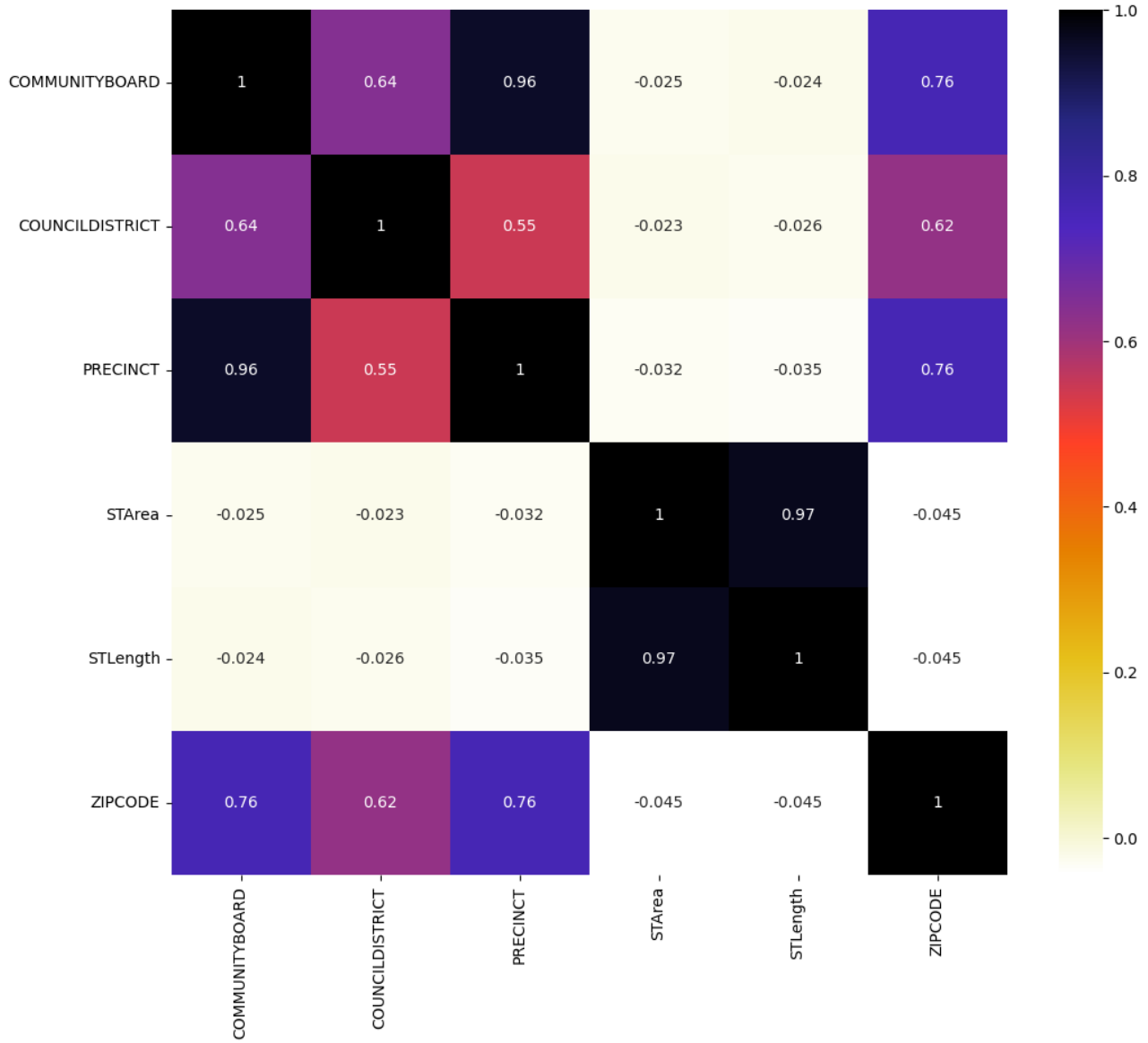


Figure 3.43: The above figure depicts a Correlation Map showing the correlation between the numerical columns in a Heat Map form. Through this, we can infer the dependencies between them.

3.2.7 Pair Plot

A pair plot, is a matrix of graphs that enables the visualization of the relationship between each pair of variables in a dataset. It can be seen as the histogram plus scatter plot version of the correlation plot.

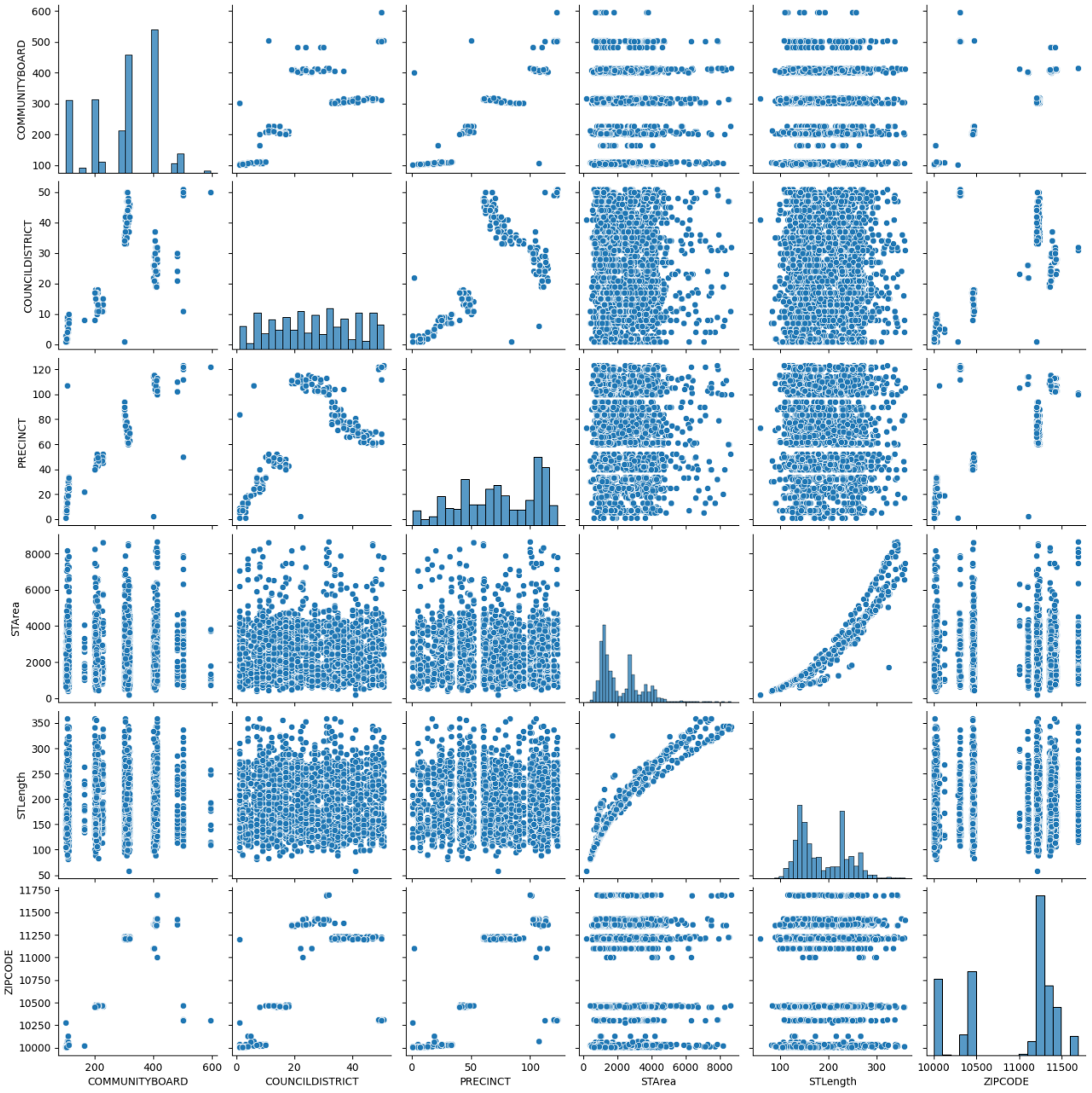


Figure 3.44: The above figure depicts a Pair Plot showing the correlation between the numerical columns in form of Scatter Plots and Bar Plots. Through this, we can infer the dependencies between them.

3.2.8 Joint Plot

A Joint Plot can be visualized as a mutilated version of the pair plot. Unlike the pair plot where the correlation of multiple variables (more than two) can be analyzed, Joint plot only takes two variables and integrates their pair plot in s single graph.

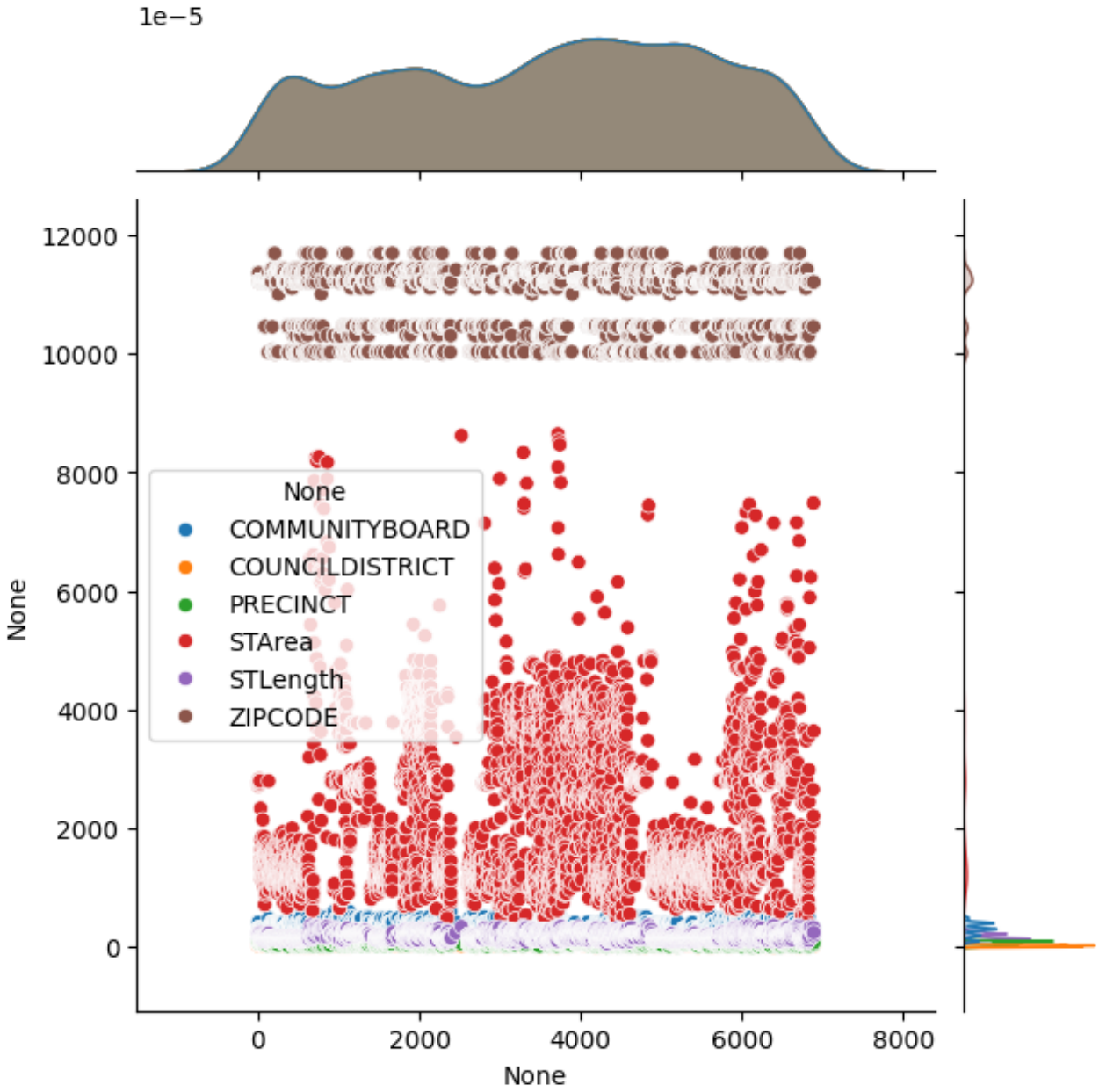


Figure 3.45: The above figure depicts a Joint Plot between the numerical columns.

Chapter 4. Performing Some Basic Problems In Our Cleaned Dataset

So we have included many basic problems to analyze the revised cleaned dataset. So the problems included will make our understanding of the dataset more clearer.

4.1 1^{st} Problem : Which is the most popular sport ?

☞ Here is the code and its output :

Which is the most popular sport?

```
most_popular_sport = df['PRIMARY_SPORT'].mode()[0]
most_popular_sport_count = df['PRIMARY_SPORT'].value_counts().max()

print("\nMost Popular Sport:", most_popular_sport)
print("Number of Occurrences:", most_popular_sport_count)
```

```
Most Popular Sport: HDB
Number of Occurrences: 2159
```

Figure 4.1: Code and output of 1^{st} problem

Okay, first, our code calculates the mode of the 'PRIMARY SPORT' column in the df DataFrame mode() function. Then it calculates how many times the most popular sport appeared and extracts the maximum count and finally prints out the most popular sport and occurrences .

So we got "HDB"(Handball) sport as the most popular sport as it has the most occurrences i.e. 2159.

4.2 2^{nd} Problem : What is the count of basketball facilities: Get the total count of entries where BASKETBALL is present ?

☞ Here is the code and its output :

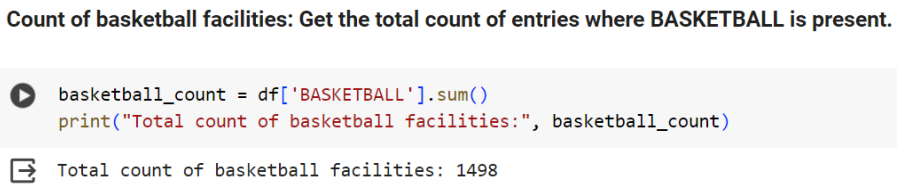


Figure 4.2: Code and output of 2^{nd} problem

The above code snippet computes the total count of entries with the presence of the 'BASKETBALL' facility in the dataset. The count of entries or basketball facilities among all the dataset entries is 1498 .

Therefore, there are 1498 entries where the presence of the 'BASKETBALL' facility is in the entire dataset.

4.3 3^{rd} Problem: What is the percentage of accessible facilities with basketball?

☞ Here is the code and its output :

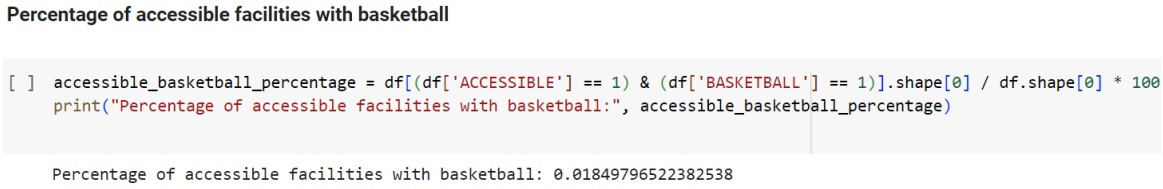


Figure 4.3: Code and output of 3^{rd} problem

This code snippet computes the proportion of accessible facilities which also have basketball facilities. First, it filters the DataFrame to choose rows whose 'ACCESSIBLE' and 'BASKETBALL' columns are True.

It then computes the percentage, which is: The percentage of accessible facilities with basketball facilities is around 0.0185%; put differently, this implies that only a minuscule proportion of accessible facilities out of the total number in the dataset have basketball facilities.

4.4 4^{th} Problem: Calculate the density of athletic facilities per zip-code?

☞ Here is the code and its output :

Calculate the density of athletic facilities per zipcode

```
zipcode_sport_counts = {}

for zipcode in df['ZIPCODE'].unique():
    df_zipcode = df[df['ZIPCODE'] == zipcode]

    sport_counts = {}

    for column in df_zipcode.columns:
        if df_zipcode[column].dtype == bool:
            true_count = df_zipcode[column].sum()
            sport_counts[column] = true_count

    max_sports = [sport for sport, count in sport_counts.items() if count == max(sport_counts.values())]

    zipcode_sport_counts[zipcode] = {"sports": max_sports, "true_count": max(sport_counts.values())}

for zipcode, data in zipcode_sport_counts.items():
    print("ZIP code:", zipcode)
    print("Sport(s) with highest true count:", data["sports"])
    print("True count:", data["true_count"])
    print()
```

Figure 4.4: Code of 4^{th} problem

```
ZIP code: 11361.0
Sport(s) with highest true count: ['TENNIS']
True count: 12

ZIP code: 11209.0
Sport(s) with highest true count: ['HANDBALL']
True count: 17

ZIP code: 11228.0
Sport(s) with highest true count: ['HANDBALL']
True count: 20

ZIP code: 11217.0
Sport(s) with highest true count: ['HANDBALL']
True count: 13

ZIP code: 11201.0
Sport(s) with highest true count: ['HANDBALL']
True count: 29

ZIP code: 11213.0
Sport(s) with highest true count: ['HANDBALL']
True count: 29

ZIP code: 11220.0
Sport(s) with highest true count: ['BASKETBALL']
True count: 22

ZIP code: 11232.0
Sport(s) with highest true count: ['BASKETBALL', 'HANDBALL']
True count: 8

ZIP code: 11203.0
Sport(s) with highest true count: ['HANDBALL']
True count: 20

ZIP code: 11238.0
Sport(s) with highest true count: ['BASKETBALL', 'HANDBALL']
True count: 6

ZIP code: 11225.0
Sport(s) with highest true count: ['BASKETBALL']
True count: 9

ZIP code: 11219.0
Sport(s) with highest true count: ['HANDBALL']
True count: 2

ZIP code: 11214.0
Sport(s) with highest true count: ['HANDBALL']
True count: 15

ZIP code: 11224.0
Sport(s) with highest true count: ['HANDBALL']
True count: 47

ZIP code: 11223.0
Sport(s) with highest true count: ['HANDBALL']
True count: 12
```

Figure 4.5: Output of 4^{th} problem.Here we have included the output of a few of the entries only.

This code snippet computes the proportion of accessible facilities that also have basketball facilities. First, it filters the DataFrame to choose rows whose 'ACCESSIBLE' and 'BASKETBALL' columns are True.

This code snippet contains the count of true values for each sport in each ZIP code and sport with the most instances of true values.

It utilizes a for-loop to iterate through unique ZIP codes and create a subset DataFrame with selected ZIP codes. It also utilizes a for-loop to iterate through columns and count for each sport. It finds out which sport has the largest count of true and stores that count and sport in the dictionary. This code snippet contains the print statement for each ZIP Code result.

For instance, in ZIP code 11361.0, the most popular sport is tennis with 12 true counts.

4.5 5^{th} Problem: Find the sport with the maximum total area and the sport with the maximum total length among active sports, and display their respective total areas and lengths

☞ Here is the code and its output :

Find the sport with the maximum total area and the sport with the maximum total length among active sports, and display their respective total areas and lengths

```
[ ]  active_sports = df[df['FEATURESTATUS'] == 'Active']
     result = active_sports.groupby('PRIMARY_SPORT').agg({'STArea': 'sum', 'STLength': 'sum'}).reset_index()
     max_area_sport = result.loc[result['STArea'].idxmax()]
     max_length_sport = result.loc[result['STLength'].idxmax()]

     print("Sport with maximum total STArea (Active):", max_area_sport['PRIMARY_SPORT'], "Total STArea:", max_area_sport['STArea'])
     print("Sport with maximum total STLength (Active):", max_length_sport['PRIMARY_SPORT'], "Total STLength:", max_length_sport['STLength'])

     Sport with maximum total STArea (Active): BKB Total STArea: 4782239.8940765
     Sport with maximum total STLength (Active): BKB Total STLength: 363053.57075475
```

Figure 4.6: Code and Output of 5^{th} problem

The code first filters the dataframe to only use entries where the feature status is equal to “Active” and then groups the filtered dataframe by primary sport and finds the sum of area and length values for each sport. Then, it finds the row with the maximum total area by using the idxmax() function on the column “STArea” of the result dataframe. Finally, it finds the row with the maximum total length by using idxmax() function on the column “STLength” of the result dataframe .

Basketball has the maximum total area for all sports, which is approximately 4,782,239.89 square units, and it has also the maximum total length among active sports, which is approximately 363,053.57 square units.

Chapter 5. Feature Engineering

Now,after visualizing the dataset after removing outliers, we do feature selection and model training. Feature engineering constitutes one of the critical components in the machine learning pipeline.

It is defined as the process of creating, modifying, or selecting features that are most suited for predictive models. Moreover, it is stated that good features make good models. Hence, feature engineering is an essential step in creating effective machine learning systems.

Feature engineering involves a wide range of techniques to garner valuable insights and information from raw data and present it in a manner that can be better learned by the model. The below can be achieved in Feature Engineering:

- Handling Categorical Variables
- Feature Creation
- Dimensionality Reduction
- Feature Scaling
- Feature Selection
- Feature Transformation

→ In our analysis , we have included handling categorical variables,feature scaling and feature selection.

5.1 Handling Categorical Variables

Meanwhile, when it comes to our categorical features, we analyzed all of them, first of all. In our dataset, categorical variables do not have a particular order, or in other words, all of these variables are not ordinal in terms of EDA, so they are nominal.

Therefore, using One-Hot encoding, we will convert categorical features into numerical ones.

One-hot encoding is used to transform each unique category of a categorical column into a new binary column where that column's unique category is represented by "1" and other columns are represented by "0".

This can be done using the pd.get_dummies()function. Then, before applying standardization and normalization, we removed null values, classifications; after that, we applied one -hot encoding.

Here, I displayed the code and the result :

```
categorical_cols = df.select_dtypes(include=['object']).columns
print(categorical_cols)
df_before_encoded = pd.get_dummies(df, columns=categorical_cols,drop_first = True)

print(df_before_encoded.shape)
print(df_before_encoded.sample(n=10))
```

Figure 5.1: Code of Encoding for handling categorical columns

```
Index(['BOROUGH', 'DEPARTMENT', 'DIMENSIONS', 'FEATURESTATUS', 'FIELD_NUMBER',
       'GISPROPNUM', 'PRIMARY_SPORT', 'SURFACE_TYPE', 'SYSTEM'],
      dtype='object')
(5406, 6372)
      ACCESSIBLE  ADULT_BASEBALL  ADULT_FOOTBALL  ADULT_SOFTBALL  BASKETBALL  \
6594       False           False           False           False       False
1212       False           False           False           False       False
5719       False           False           False           False       False
5043       False           False           False           False       False
852        False           False           False           False       False
5210       False           False           False           False       False
2924       False           False           False           False       False
601        False           False           False           False       False
3912       False           False           False           False        True
5346       False           False           False           False       False

      BOCCE  COMMUNITYBOARD  COUNCILDISTRICT  CRICKET  FIELD_LIGHTED  ...  \
6594  False             411               19    False          False  ...
1212  False             112                7    False          False  ...
5719  False             305               37    False          False  ...
5043  False             411               23    False          False  ...
852   False             405               30    False          False  ...
5210  False             104                3    False          False  ...
2924   True             501               49    False          False  ...
601   False             201               17    False          False  ...
3912  False             409               28    False          False  ...
5346  False             209               18    False          False  ...

      SYSTEM_X344-TENNIS-2  SYSTEM_X344-TENNIS-3  SYSTEM_X344-TENNIS-4  \
6594                 False                 False                 False
1212                 False                 False                 False
5719                 False                 False                 False
5043                 False                 False                 False
852                  False                 False                 False
5210                 False                 False                 False
2924                 False                 False                 False
601                  False                 False                 False
3912                 False                 False                 False
5346                 False                 False                 False

      SYSTEM_X344-TENNIS-5  SYSTEM_X344-TENNIS-6  SYSTEM_X344-TENNIS-7  \
6594                 False                 False                 False
1212                 False                 False                 False
5719                 False                 False                 False
5043                 False                 False                 False
852                  False                 False                 False
5210                 False                 False                 False
2924                 False                 False                 False
601                  False                 False                 False
3912                 False                 False                 False
5346                 False                 False                 False

      SYSTEM_X344-TENNIS-8  SYSTEM_X344-TENNIS-9  SYSTEM_x214-BASKETBALL-2  \
6594                 False                 False                     False
1212                 False                 False                     False
5719                 False                 False                     False
5043                 False                 False                     False
852                  False                 False                     False
5210                 False                 False                     False
2924                 False                 False                     False
601                  False                 False                     False
3912                 False                 False                     False
```

Figure 5.2: Output of Encoding for handling categorical columns

5.2 Feature Scaling

Feature scaling is a type of data preprocessing that helps standardize or normalize the range of all features within a given dataset.

The purpose of feature scaling is to ensure that all features have the same-scale because, for many machine learning algorithms, it is necessary for them to function properly.

As numerous machine learning algorithms are sensitive to the scale of features, when there are disparities in the scale of features, large-scale features have the potential to dominate small-scale features, leading to bias in the analysis.

Hence, scaling features help algorithms to optimize faster, reduce feature magnitudes disparities, and make the optimization process more stable. Two feature scaling methods were used in our analysis were standardization and normalization.

1.) **Standardization**:-

Standardization is also referred to as *z*-score normalization, which transforms features so that they are centered with mean of 0 and standard deviation of 1.

This is achieved by subtracting the feature mean from the data, and then dividing by the feature standard deviation. This approach is appropriate if there is a Gaussian distribution of features.

We have presented the code and the results of the implementation on our dataset.(In this, we have used the StandardScaler() function from sklearn.preprocessing inbuilt library) .

Please do note here that the outputs and code don't have the one-hot encoding. We have printed it to just get the insight of normalization and standardization in our dataset.

```
non_numeric_columns = data_types[data_types == 'object'].index
print("Non-numeric columns:")
print(non_numeric_columns)
df_non_numeric = df.drop(columns=numeric_columns)
numeric_data = df[numeric_columns]
numeric_data.reset_index(drop=True, inplace=True)
df_non_numeric.reset_index(drop=True, inplace=True)

# Concatenate df_non_numeric with scaled_df_standardized or scaled_df_normalized
df_scaled_standardized = pd.concat([df_non_numeric, scaled_df_standardized], axis=1)
df_scaled_normalized = pd.concat([df_non_numeric, scaled_df_normalized], axis=1)
```

Figure 5.3: Code for Standardization Normalization

```
print("Scaled DataFrame after standardization:")
print(df_scaled_standardized.sample(n=10))
```

Figure 5.4: Code for Printing the dataset after standardization

```
Scaled DataFrame after standardization:
     BOROUGH DEPARTMENT  DIMENSIONS FEATURESTATUS FIELD_NUMBER GISPROPNUM  \
182        Q       Q-14    Handball        Active            2       Q367
3698       Q       Q-10       Tennis        Active            1       Q092
1431       M       M-11  Full Court        Active            1       M047
3566       B       B-15       Tennis        Active            5       B051
4489       R       R-01    Handball      Inactive            3       R034
553        M       M-12    Handball        Active            3       M199
84         B       B-03    Handball        Active            3       B294
1493       X       X-13  Full Court        Active            1       X092
2620       M       M-10  Half Court        Active          01A       M110
2894       B       B-16  Full Court        Active            3       B219

     PRIMARY_SPORT SURFACE_TYPE               SYSTEM  ACCESSIBLE  ...  RUGBY  \
182            HDB      Asphalt       Q367-HANDBALL-1   -0.019238  ...    0.0
3698           TNS      Asphalt         Q092-TENNIS-1   -0.019238  ...    0.0
1431           BKB      Asphalt  M047-03-BASKETBALL-1   -0.019238  ...    0.0
3566           TNS      Asphalt         B051-TENNIS-7   -0.019238  ...    0.0
4489           HDB      Asphalt       R034-HANDBALL-2   -0.019238  ...    0.0
553            HDB      Asphalt       M199-HANDBALL-1   -0.019238  ...    0.0
84             HDB      Asphalt       B294-HANDBALL-1   -0.019238  ...    0.0
1493           BKB      Asphalt  X092-06-BASKETBALL-1   -0.019238  ...    0.0
2620           BKB      Asphalt     M110-BASKETBALL-2   -0.019238  ...    0.0
2894           BKB      Asphalt     B219-BASKETBALL-3   -0.019238  ...    0.0

        STArea  STLength    TENNIS  TRACK_AND_FIELD    T_BALL  VOLLEYBALL  \
182  -0.490784 -0.416660 -0.361233              0.0 -0.047167   -0.146772
3698  0.494295  0.779808  2.768296              0.0 -0.047167   -0.146772
1431  1.446689  1.479784 -0.361233              0.0 -0.047167   -0.146772
3566  0.509385  0.807082  2.768296              0.0 -0.047167   -0.146772
4489 -0.935710 -1.072387 -0.361233              0.0 -0.047167   -0.146772
553  -1.005928 -1.211516 -0.361233              0.0 -0.047167   -0.146772
84   -1.057983 -1.251005 -0.361233              0.0 -0.047167   -0.146772
1493  1.603502  1.585876 -0.361233              0.0 -0.047167   -0.146772
2620 -1.289698 -1.712284 -0.361233              0.0 -0.047167   -0.146772
2894  1.558275  1.554082 -0.361233              0.0 -0.047167   -0.146772

      WHEELCHAIRFOOTBALL  YOUTH_FOOTBALL   ZIPCODE
182                  0.0       -0.030426  1.452207
3698                 0.0       -0.030426  0.938635
1431                 0.0       -0.030426 -1.678138
3566                 0.0       -0.030426  0.579323
4489                 0.0       -0.030426 -1.149516
553                  0.0       -0.030426 -1.668732
84                   0.0       -0.030426  0.586847
1493                 0.0       -0.030426 -0.848521
2620                 0.0       -0.030426 -1.663088
2894                 0.0       -0.030426  0.586847

[10 rows x 44 columns]
```

Figure 5.5: Output for Printing the dataset after standardization

2.) **Normalization** :-

Normalization , aka min:max scaling, scales features to a given range, usually 0 to 1. This is done by subtracting the minimal possible value for each element from the data, and then dividing it by the range, which is the maximal value minus the minimal value.

It is generally useful in case of skewed or when the algorithm explicitly requires the features to follow in a bounded interval.

We have presented the code and the results of the implementation on our dataset. In this case, we used the MinMaxScaler() function.

```
print("\nScaled DataFrame after normalization:")
print(df_scaled_normalized.sample(n=10))
```

Figure 5.6: Code for Printing the dataset after normalization

```
Scaled DataFrame after normalization:
     BOROUGH DEPARTMENT  DIMENSIONS       FEATURESTATUS FIELD_NUMBER  \
3328       B       B-12  Half Court              Active            1
170        M       M-11    Handball              Active            7
301        Q       Q-08    Handball              Active            2
1209       B       B-13    Handball              Active           12
1765       B       B-05    Handball              Active            4
2581       B       B-02  Full Court              Active            1
1970       B       B-16    Handball  Closed Temporarily            9
1808       M       M-14    Handball              Active            4
2901       B       B-16  Full Court              Active            2
2251       Q       Q-01  Full Court              Active            1

     GISPROPNUM PRIMARY_SPORT SURFACE_TYPE                SYSTEM  ACCESSIBLE  \
3328       B297           BKB      Asphalt      B297-BASKETBALL-2         0.0
170        M192           HDB      Asphalt        M192-HANDBALL-3         0.0
301        Q354           HDB      Asphalt        Q354-HANDBALL-1         0.0
1209       B129           HDB      Asphalt  B129-ZN01-HANDBALL-10         0.0
1765       B214           HDB      Asphalt     B214-01-HANDBALL-5         0.0
2581       B292           BKB      Asphalt      B292-BASKETBALL-1         0.0
1970       B008           HDB      Asphalt     B008-03-HANDBALL-1         0.0
1808       M071           HDB      Asphalt     M071-21-HANDBALL-5         0.0
2901       B153           BKB      Asphalt      B153-BASKETBALL-2         0.0
2251       Q298           BKB      Asphalt      Q298-BASKETBALL-1         0.0

      ...  RUGBY    STArea  STLength  TENNIS  TRACK_AND_FIELD  T_BALL  \
3328  ...    0.0  0.126307  0.285927     0.0              0.0     0.0
170   ...    0.0  0.100744  0.249155     0.0              0.0     0.0
301   ...    0.0  0.138404  0.304021     0.0              0.0     0.0
1209  ...    0.0  0.164853  0.340297     0.0              0.0     0.0
1765  ...    0.0  0.105089  0.258929     0.0              0.0     0.0
2581  ...    0.0  0.422590  0.665122     0.0              0.0     0.0
1970  ...    0.0  0.135743  0.302005     0.0              0.0     0.0
1808  ...    0.0  0.100825  0.258916     0.0              0.0     0.0
2901  ...    0.0  0.449438  0.680926     0.0              0.0     0.0
2251  ...    0.0  0.355991  0.590367     0.0              0.0     0.0

      VOLLEYBALL  WHEELCHAIRFOOTBALL  YOUTH_FOOTBALL   ZIPCODE
3328         0.0                 0.0             0.0  0.718842
170          0.0                 0.0             0.0  0.016539
301          0.0                 0.0             0.0  0.806261
1209         0.0                 0.0             0.0  0.722386
1765         0.0                 0.0             0.0  0.712345
2581         0.0                 0.0             0.0  0.711164
1970         0.0                 0.0             0.0  0.715298
1808         0.0                 0.0             0.0  0.013585
2901         0.0                 0.0             0.0  0.727702
2251         0.0                 0.0             0.0  0.652097

[10 rows x 44 columns]
```

Figure 5.7: Output for Printing the dataset after normalization

We have also made a data frame called df_encoded in which null values and outliers are removed and after that normalization is performed to numerical columns and afterwards categorical variables are converted into numeric variables by one-hot encoding.We have attached the output of the same :

```
Index(['BOROUGH', 'DEPARTMENT', 'DIMENSIONS', 'FEATURESTATUS', 'FIELD_NUMBER',
       'GISPROPNUM', 'PRIMARY_SPORT', 'SURFACE_TYPE', 'SYSTEM'],
      dtype='object')
(5406, 6372)
      ACCESSIBLE  ADULT_BASEBALL  ADULT_FOOTBALL  ADULT_SOFTBALL  BASKETBALL  \
5217         0.0             0.0             0.0             0.0         0.0
4834         0.0             0.0             0.0             0.0         0.0
2358         0.0             0.0             0.0             0.0         1.0
1866         0.0             0.0             0.0             0.0         0.0
2374         0.0             0.0             0.0             0.0         1.0
1049         0.0             0.0             0.0             0.0         0.0
2904         0.0             0.0             0.0             0.0         0.0
2837         0.0             0.0             0.0             0.0         1.0
1939         0.0             0.0             0.0             0.0         0.0
10           0.0             0.0             0.0             0.0         0.0

      BOCCE  COMMUNITYBOARD  COUNCILDISTRICT  CRICKET  FIELD_LIGHTED  ...  \
5217    0.0        0.439271             0.90      0.0            0.0  ...
4834    0.0        0.439271             0.82      0.0            0.0  ...
2358    0.0        0.627530             0.44      0.0            0.0  ...
1866    0.0        0.020243             0.14      0.0            0.0  ...
2374    0.0        0.631579             0.44      0.0            0.0  ...
1049    0.0        0.257085             0.24      0.0            0.0  ...
2904    0.0        0.435223             0.72      0.0            0.0  ...
2837    0.0        0.631579             0.44      0.0            0.0  ...
1939    0.0        0.629555             0.52      0.0            0.0  ...
10      0.0        0.423077             0.84      0.0            0.0  ...

      SYSTEM_X344-TENNIS-2  SYSTEM_X344-TENNIS-3  SYSTEM_X344-TENNIS-4  \
5217                 False                 False                 False
4834                 False                 False                 False
2358                 False                 False                 False
1866                 False                 False                 False
2374                 False                 False                 False
1049                 False                 False                 False
2904                 False                 False                 False
2837                 False                 False                 False
1939                 False                 False                 False
10                   False                 False                 False

      SYSTEM_X344-TENNIS-5  SYSTEM_X344-TENNIS-6  SYSTEM_X344-TENNIS-7  \
5217                 False                 False                 False
4834                 False                 False                 False
2358                 False                 False                 False
1866                 False                 False                 False
2374                 False                 False                 False
1049                 False                 False                 False
2904                 False                 False                 False
2837                 False                 False                 False
1939                 False                 False                 False
10                   False                 False                 False

      SYSTEM_X344-TENNIS-8  SYSTEM_X344-TENNIS-9  SYSTEM_x214-BASKETBALL-2  \
5217                 False                 False                     False
4834                 False                 False                     False
2358                 False                 False                     False
1866                 False                 False                     False
2374                 False                 False                     False
1049                 False                 False                     False
2904                 False                 False                     False
2837                 False                 False                     False
1939                 False                 False                     False
```

Figure 5.8: Output for Printing the dataset after normalization and removing outliers and null values and after applying one-hot encoding

5.3 Feature selection

Feature selection is an essential procedure in machine learning and data analysis in which one identifies the most relevant and crucial features and discards the rest as irrelevant or redundant .

Feature selection adequately boosts an ML model's performance by reducing overfitting, minimizing computational complexity and boosting the model's interpretability.

It helps in identifying the valuable features of your model hence ensure that the ML model is focused on the core areas of the data, leading to high generalization and inference accuracy.

We have included 2 methods of feature selection in our analysis.

- Mutual Information Feature Selection Technique
- Variance Thresholding Feature Selection Technique

5.3.1 Mutual Information Feature Selection Method: Target Variable: STArea

So first of all we will choose the Target Variable. So In our analysis, We have taken STArea as the target variable and applied Mutual Information Regression method to select features as it has continuous values we will discuss the method in the next paragraph .

So we also removed the columns named Gispropnum and System column as they don't have any relation to the columns named STArea and then also apply the one-hot encoding.

So we get the shape of revised data frame (5406,221) . The following code shows the same :

```
Xtemp = df.drop(columns = ['SYSTEM','GISPROPNUM'])
categorical_cols = Xtemp.select_dtypes(include=['object']).columns
Xtemp_before_encoded = pd.get_dummies(Xtemp, columns=categorical_cols,drop_first = True)
print(Xtemp_before_encoded)


X = Xtemp_before_encoded.drop(columns=['STArea', 'STLength'])
y_area = Xtemp_before_encoded['STArea']
y_length = Xtemp_before_encoded['STLength']
mi_area = mutual_info_regression(X, y_area)
# mi_length = mutual_info_regression(X, y_length)
```

Figure 5.9: Code of Encoding of the revised dataset for performing Mutual Information Feature Selection

```
      PRIMARY_SPORT_VLB  SURFACE_TYPE_Clay  SURFACE_TYPE_Concrete  \
0                 False              False                  False
1                 False              False                  False
2                 False              False                  False
3                 False              False                  False
4                 False              False                  False
...                 ...                ...                    ...
6878              False              False                  False
6880              False              False                  False
6883              False              False                  False
6884              False              False                  False
6887              False              False                  False

      SURFACE_TYPE_Hard Top  SURFACE_TYPE_Mondo  SURFACE_TYPE_Natural  \
0                     False               False                 False
1                     False               False                 False
2                     False               False                 False
3                     False               False                 False
4                     False               False                 False
...                     ...                 ...                   ...
6878                  False               False                 False
6880                  False               False                 False
6883                  False               False                 False
6884                  False               False                 False
6887                  False               False                 False

      SURFACE_TYPE_Rubber  SURFACE_TYPE_Sand  SURFACE_TYPE_Synthetic  \
0                   False              False                   False
1                   False              False                   False
2                   False              False                   False
3                   False              False                   False
4                   False              False                   False
...                   ...                ...                     ...
6878                False              False                   False
6880                False              False                   False
6883                False              False                   False
6884                False              False                   False
6887                False              False                   False

      SURFACE_TYPE_Wood
0                 False
1                 False
2                 False
3                 False
4                 False
...                 ...
6878              False
6880              False
6883              False
6884              False
6887              False

[5406 rows x 221 columns]
```

Figure 5.10: Output of Encoding of the revised dataset for performing Mutual Information Feature Selection.Here we have included only few columns for displaying.

Overview On Mutual Information Feature Selection Technique :

- Mutual Information is a measurement of the relationships between two variables. The quantity of information is achieved regarding one variable via the other variables.

- When applied to feature selection, mutual information calculates the quantity of information gained concerning the target variable.

- That is, features that provide a lot of information concerning the target variable are more informative and likely to be relevant when predicting the target result .

- Mutual information is utilized as choice for feature selection to pinpoint the most critical features within a dataset.

- It orders features dependent on the level of attachment to the target variable and is frequently utilized as a part of feature selection techniques to elevate machine learning models' presentation.

Afterwards we will visualize the top 10 features or variables which have high mutual information with our target variable :

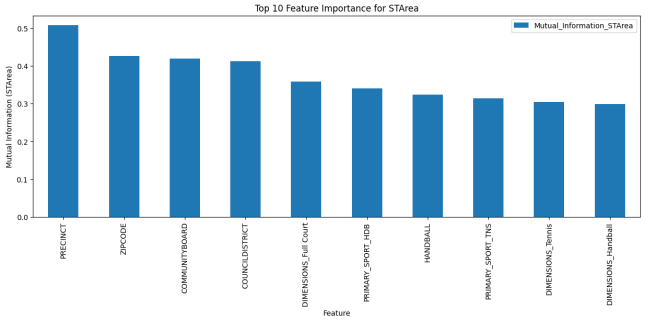


Figure 5.11: Top 10 features with higher mutual information with STArea

The following output shows the feature importance of all numerical columns of the revised dataframe by the mutual information selection technique :

```
print("Feature Importance for STArea:")
print(feature_importance_area)
```

```
Feature Importance for STArea:
                    Feature  Mutual_Information_STArea
22                 PRECINCT                   0.508358
32                  ZIPCODE                   0.427194
6           COMMUNITYBOARD                   0.419734
7           COUNCILDISTRICT                  0.413238
111  DIMENSIONS_Full Court                   0.358278
..                     ...                        ...
140       FIELD_NUMBER_04A                   0.000000
191        FIELD_NUMBER_48                   0.000000
163        FIELD_NUMBER_23                   0.000000
153        FIELD_NUMBER_15                   0.000000
159   FIELD_NUMBER_2 and 4                   0.000000

[219 rows x 2 columns]
```

Figure 5.12: List of all features with mutual information with our target variable STArea

So we can clearly see that the PRECINCT column has the highest mutual information with STArea (0.508358) depicting that it has the highest role in the feature selection of the target variable STArea.

Afterwards as we got the mutual information of all the numerical columns with STArea , then we will apply the SelectPercentile() function from sklearn.feature_selection.Here we have selected top 20 percentile of our total features or total numerical columns .(This 20 we have selected and it is user based parameter or we can say it is hyperparamter).You can see from our code :

```
## Selecting the top 20 percentile
selected_top_columns_area = SelectPercentile(score_func=mutual_info_regression, percentile=20)
selected_top_columns_area.fit(X, y_area)
```

```
SelectPercentile(percentile=20,
                 score_func=<function mutual_info_regression at 0x7a50f892ed40>)
```

Figure 5.13: Code of top 20 percentile feature selection from 221 available features by comparing the mutual information of features with STArea

So after fitting and by using get_support() functions, we will get the columns of the selected features which are top 20 percentile. So in total we got 44 features . So this is the output :

```
X.columns[selected_top_columns_area.get_support()]
```

```
Index(['BASKETBALL', 'COMMUNITYBOARD', 'COUNCILDISTRICT', 'FIELD_LIGHTED',
       'HANDBALL', 'LL_SOFTBALL', 'MAINTENANCEAGREEMENT', 'PRECINCT', 'TENNIS',
       'VOLLEYBALL', 'ZIPCODE', 'BOROUGH_M', 'BOROUGH_Q', 'BOROUGH_R',
       'BOROUGH_X', 'DEPARTMENT_B-02', 'DEPARTMENT_B-13', 'DEPARTMENT_B-14',
       'DEPARTMENT_B-18', 'DEPARTMENT_M-11R', 'DEPARTMENT_Q-05',
       'DEPARTMENT_Q-13', 'DEPARTMENT_Q-14', 'DEPARTMENT_Q-15',
       'DEPARTMENT_Q-16', 'DEPARTMENT_R-02', 'DEPARTMENT_X-04',
       'DIMENSIONS_Bocce', 'DIMENSIONS_Full Court', 'DIMENSIONS_Half Court',
       'DIMENSIONS_Handball', 'DIMENSIONS_Tennis', 'DIMENSIONS_Volleyball',
       'FEATURESTATUS_Inactive', 'FIELD_NUMBER_1', 'PRIMARY_SPORT_BOC',
       'PRIMARY_SPORT_HDB', 'PRIMARY_SPORT_SCR', 'PRIMARY_SPORT_SFB',
       'PRIMARY_SPORT_TNS', 'PRIMARY_SPORT_VLB', 'SURFACE_TYPE_Clay',
       'SURFACE_TYPE_Concrete', 'SURFACE_TYPE_Sand'],
      dtype='object')
```

```
len(X.columns[selected_top_columns_area.get_support()])
```

```
44
```

Figure 5.14: Output of Top 20 percentile feature selection from 221 available features by comparing the mutual information of features with STArea

5.3.2 Variance Threshold Feature Selection Method : Target Variable: STLength

So first of all we will choose the Target Variable. So In our analysis, We have taken STLength as the target variable and applied Variance Thresholding Feature Seelction method to select features as it has continuous values we will discuss the method in the next paragraph .

So we also removed the columns named Gispropnum and System column as they don't have any relation to the columns named STLength and then also apply the one-hot encoding.

So we get the shape of revised data frame (5406,221) .We have done the same thing previously in Mutual Information initially .

Overview On Variance Thresholding Feature Selection Technique :

- Variance thresholding is a straightforward feature selection method, discarding features having a variance lower than a pre-set threshold. The threshold, in practice, will be assumed based on domain knowledge or cross-validated.
- A low threshold threshold = 0.1 is applied in this case, at which features with a variance difference lower than this fractional value are discarded.
- The main objective of the variance threshold is to reduce the dimension by discarding identical features, and returning features that do not vary much within the samples, lower variance means a lower power to discriminate and is less important for the predictive model output.

In our analysis, we also set the threshold to 0.1. Therefore, once we use the fit and get_support() function again for applying our set threshold, we will get selected features, which have a variance of more than 0.1. So, a total of 19 features were acquired, and the output with the final result of selected features via Variance thresholding feature selection is:

```
[ ]  selector_vt = VarianceThreshold(threshold=0.1)
     X_selected_vt = selector_vt.fit_transform(X)

[ ]  selected_features_vt = X.columns[selector_vt.get_support()]

     print("Selected Features after Variance Thresholding:")
     print(selected_features_vt)

     Selected Features after Variance Thresholding:
     Index(['BASKETBALL', 'COMMUNITYBOARD', 'COUNCILDISTRICT', 'HANDBALL',
            'PRECINCT', 'TENNIS', 'ZIPCODE', 'BOROUGH_M', 'BOROUGH_Q', 'BOROUGH_X',
            'DIMENSIONS_Full Court', 'DIMENSIONS_Half Court', 'DIMENSIONS_Handball',
            'DIMENSIONS_Tennis', 'FIELD_NUMBER_1', 'FIELD_NUMBER_2',
            'FIELD_NUMBER_3', 'PRIMARY_SPORT_HDB', 'PRIMARY_SPORT_TNS'],
           dtype='object')

[ ]  len(X.columns[selector_vt.get_support()])

     19
```

Figure 5.15: 19 features selected from available 221 features who have variance value higher than threshold = 0.1

Chapter 6. Model fitting and Evaluation

Model fitting consists of finding the optimal parameters of a mathematical or statistical model based on the input variables to predict a target variable .

The final goal of model fitting is to find the parameter values that minimize the divergence of the observed data from the data produced by the model .

Model fitting consists of the following steps :

- **Model selection**

  ☞It selects the fitting model or model class, and estimates the values of its parameters.

  ☞It is usually done by optimization of a predefined objective function .

  ☞Model fitting may be achieved via optimization methods such as optimization algorithms – iterative methods for finding optimal parameter values.

  ☞Some examples include gradient descent; generalized gradient descent, also known as stochastic gradient descent; or other algorithms specific to model classes .

- **Performance evaluation**

  ☞ After fitting the model on the training data, it is necessary to evaluate it on the test data – the same data on which model fitting has not been performed .

  ☞Model performance evaluation is based on specific metrics – classification task: accuracy; for binary classification tasks, it is frequently combined with other metrics such as precision and recall; the F1-score, for instance and for regression problems there are evaluation metrics such as root mean squared error, R2 score etc.

Here we have used **regression** as well as **classification** methods for model fitting.

So now the question arises when to apply regression and when to apply classification ?

1.) Regression :-

- Regression is employed when the target variable is continuous and numeric.
- Regression is employed when the target variable is continuous and numeric. Some examples of regression tasks are house prices prediction, stock prices forecasting, temperature estimation, or sales revenue prediction.
- Regression models are designed to learn the real relationship between the input features and the continuous target variable to forecast future or unseen values of the target variable where they fall within the range of the target.
- Some regression techniques are linear regression, polynomial regression, SVR – support vector regression, decision tree regression, random forest regression , and regression by neural network .

2.) Classification :-

- Classification is applied when the target variable is categorical and discrete.
- Spam detection, customer satisfaction prediction, disease diagnosis, or image classification are classification tasks.
- Classification firstly defines the class or category to which the given data belongs or will be allocated. This process is executed based on the input attribute values.
- Logistic regression, decision trees, random forests, SVM – support vector machine , k-NN – k-nearest neighbors, naive Bayes, or neural network for classification are common classification techniques .

Now we will analyze both regression as well as classification in our dataset try to do the model fitting and compare the evaluation metrics of different regression and classification techniques.

6.1 Regression Problem

6.1.1 1st Problem : Predicting the STArea of unseen data(test data) by the help of features selected from Mutual Information Feature Selection Technique.

So as we know STArea has numerical values and it has continuous values .

So we can apply Regression to this variable .So in total we have applied 2 regression models i.e. Linear Regression and Random Forest Regression and afterwards calculates the R2 score of both of them .

In this we have taken only selected features i.e. only after applying the Mutual Information Feature Selection i.e. taken 44 features and our model's goal or our regression task is predicting STArea of test data provided i.e. unseen data .This is our first problem.

So we have to note here that we can't apply Classification problem as our target variable can be categorized discrete classes .

So now we will look at basic overview of **Linear Regression** and **Random Forest Regression**.

1.) Linear Regression :-

- Linear regression is pervasive method of modelling the linear relationship between the dependent variable and one or more independent variables.
- This is the assumption that the target variable can be represented as linear combination of the input vectors. Linear regression estimates the weight or coefficient of the input feature and finds the linear relationship's strength and direction characteristic.
- Linear regression is primarily used for forecasting and predicting continuous numeric values called target variables.
- For instance, in finance, economics, and social sciences, linear regression is widely used. Since the model is interpretable and easy, it has been a manual and a benchmark for governing more sophisticated models.
- Linear regression assumes that the independent variable has a linear relationship, but this assumption from reality may not be sustainable.

2.) Random Forest Regression :-

- Random Forest Regression is an ensemble learning technique that builds a regression model on several decision trees .
- The method fits a number of decision trees on subsets created randomly from data and combines predictions of several individual trees to provide more accurate final prediction.

- Because random forest regression can model nonlinear correlation between predictors and output, it is a more flexible and robust algorithm than linear regression.
- Moreover, it minimizes the overfitting problem by reducing the variance of individual trees. Random forest regression is applied for regression and classification problems on large datasets with numerous input features.
- The algorithm is not based on the assumption of linearity and requires standardization, making it more independent of data distribution.
- However, random forest regression is not as interpretable as a linear regression since multiple trees are gathering. It also requires more time and computational power than linear regression.

So , we have applied both these methods by importing LinearRegression inbuilt function from sklearn.linear_model and importing RandomForestRegressor inbuilt function from sklearn.ensemble.

We have displayed the code of it as follows :

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import accuracy_score, mean_squared_error,r2_score

# Split the data into training and testing sets
X_selected_area = X[X.columns[selected_top_columns_area.get_support()]]
# print(X_selected_area)
X_train, X_test, y_train, y_test = train_test_split(X_selected_area, y_area, test_size=0.2, random_state=42)


# Logistic Regression can't be applied because the target variable is not continuous
# logistic_model = LogisticRegression()
# logistic_model.fit(X_train, y_train)
# logistic_pred = logistic_model.predict(X_test)
# logistic_accuracy = accuracy_score(y_test, logistic_pred)
# print("Logistic Regression Accuracy:", logistic_accuracy)

# Train and evaluate Linear Regression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
linear_pred = linear_model.predict(X_test)
linear_mse = mean_squared_error(y_test, linear_pred)
linear_r2 = r2_score(y_test, linear_pred)
print("Linear Regression : R2 score:", linear_r2)
# print("Linear Regression Mean Squared Error:", linear_mse)

# # Train and evaluate Random Forest Classifier
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
rf_mse = mean_squared_error(y_test, rf_pred)
rf_r2 = r2_score(y_test, rf_pred)
# print("Random Forest Mean Squared Error:", rf_mse)
print(" Random Forest Regression : R2_Score",rf_r2)
```

Figure 6.1: Code showing the appliance of Linear Regression and Random Forest Regressor for the target varible STArea.Note we can apply Regression as STArea has continuous values.

The output of the following code is as follows :

```
Linear Regression : R2 score: 0.6973729153929771
 Random Forest Regression : R2_Score 0.7158996549844858
```

Figure 6.2: Output showing the R2 score of Linear and Random Forest Regression.

We have the output of R2 score of both these regression models . Let's have a summary on R2 score :

R2 score describes goodness-of-fit of the model by the comparison of variance of predicted values to the variance of the actual values of the response variable.

Ideally for a perfect model , R2 score =1.

We have Random Forest Regressor's R2 score near to 1 so it is better model to fit.

Then we have predicted values vs true predictions graphically of both the models as follows :

Here is the code of plotting the linear regression as well as random forest regression predictions vs true predictions :

```
import matplotlib.pyplot as plt


# Plotting Linear Regression predictions
plt.figure(figsize=(8, 6))
plt.scatter(y_test, linear_pred, color='orange', label='Linear Regression Predictions')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--', label='Ideal Predictions')
plt.title('Linear Regression Predictions vs Actual')
plt.xlabel('Actual STArea')
plt.ylabel('Predicted STArea')
plt.legend()
plt.show()

# Plotting Random Forest predictions
plt.figure(figsize=(8, 6))
plt.scatter(y_test, rf_pred, color='purple', label='Random Forest Predictions')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--', label='Ideal Predictions')
plt.title('Random Forest Predictions vs Actual')
plt.xlabel('Actual STArea')
plt.ylabel('Predicted STArea')
plt.legend()
plt.show()
```

Figure 6.3: Code of plotting the linear regression as well as random forest regression predictions vs true predictions for STArea

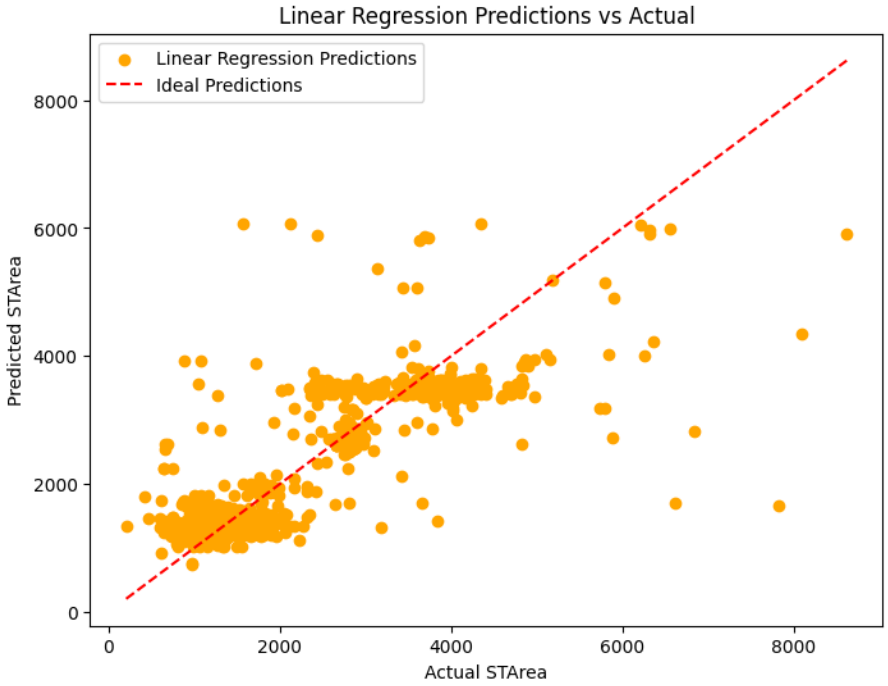


Figure 6.4: Plot of linear regression predicted values vs true values for STArea

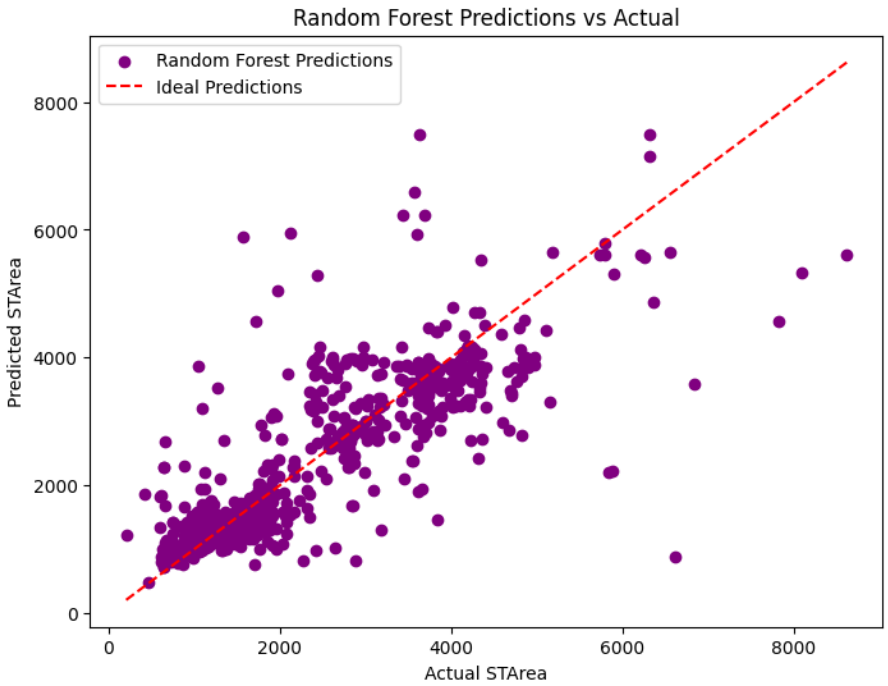


Figure 6.5: Plot of random forest regression predicted values vs true values for STArea

So we can see from the above graphs that Random Forest Regression can fit or predict the unseen values or test values of STArea more accurately than Linear Regression.

6.1.2 2nd Problem: Predicting the STLength of unseen data(test data) with the help of features selected from Variance Threshold Feature Selection Technique

Likewise, we know STLength has numerical values and it also has continuous values. So we can do Regression on this variable.

Thus, we have done 2 regression models i.e. Linear Regression, Random Forest Regression, and then calculated the R2 score of both.

In this we have taken the selected Features after the variance threshold is implemented i.e., we took 19 features and then our model's goal is to predict the STLength of unseen data. Unseen data means our test data. This is our second problem.

So, here also STLength has continuous values and even we can't solve it as a classification problem.

As we already discussed Linear Regression and random Forest Regression, are now the same as we're applying for our STLength Prediction Model but with our Variance Thresholding.

So, we have applied both these methods by importing LinearRegression inbuilt function from sklearn.linear_model and importing RandomForestRegressor inbuilt function from sklearn.ensemble.

We have displayed the code of it as follows :

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import accuracy_score, mean_squared_error,r2_score

# Split the data into training and testing sets
X_selected_length = X[X.columns[selector_vt.get_support()]]
# print(X_selected_area)
X_train, X_test, y_train, y_test = train_test_split(X_selected_length, y_length, test_size=0.2, random_state=42)


# Logistic Regression can't be applied because the target variable is not continuous
# logistic_model = LogisticRegression()
# logistic_model.fit(X_train, y_train)
# logistic_pred = logistic_model.predict(X_test)
# logistic_accuracy = accuracy_score(y_test, logistic_pred)
# print("Logistic Regression Accuracy:", logistic_accuracy)

# Train and evaluate Linear Regression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
linear_pred = linear_model.predict(X_test)
linear_mse = mean_squared_error(y_test, linear_pred)
linear_r2 = r2_score(y_test, linear_pred)
print("Linear Regression : R2 score:", linear_r2)
# print("Linear Regression Mean Squared Error:", linear_mse)

# # Train and evaluate Random Forest Classifier
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
rf_mse = mean_squared_error(y_test, rf_pred)
rf_r2 = r2_score(y_test, rf_pred)
# print("Random Forest Mean Squared Error:", rf_mse)
print(" Random Forest Regression : R2_Score",rf_r2)
```

Figure 6.6: Code showing the appliance of Linear Regression and Random Forest Regressor for the target variable STLength.Note we can apply Regression as STLength has continuous values.

The output of the following code is as follows :

```
Linear Regression : R2 score: 0.7105962149189429
 Random Forest Regression : R2_Score 0.7273728941995254
```

Figure 6.7: Output showing the R2 score of Linear and Random Forest Regression.

We have the output of R2 score of both these regression models. Let's have a summary on the R2 score :

R2 score describes goodness-of-fit of model by the comparison of variance of predicted values to the variance of the actual values of the response variable.

Ideally for a perfect model , R2 score =1.

We have Random Forest Regressor's R2 score near 1 so it is a better model to fit.

Then we have predicted values vs true predictions graphically of both the models as follows:

Here is the code for plotting the linear regression as well as random forest regression predictions vs true predictions :

```
import matplotlib.pyplot as plt


# Plotting Linear Regression predictions
plt.figure(figsize=(8, 6))
plt.scatter(y_test, linear_pred, color='orange', label='Linear Regression Predictions')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--', label='Ideal Predictions')
plt.title('Linear Regression Predictions vs Actual')
plt.xlabel('Actual STLength')
plt.ylabel('Predicted STLength')
plt.legend()
plt.show()

# Plotting Random Forest predictions
plt.figure(figsize=(8, 6))
plt.scatter(y_test, rf_pred, color='purple', label='Random Forest Predictions')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--', label='Ideal Predictions')
plt.title('Random Forest Predictions vs Actual')
plt.xlabel('Actual STLength')
plt.ylabel('Predicted STLength')
plt.legend()
plt.show()
```

Figure 6.8: Code of plotting the linear regression as well as random forest regression predictions vs true predictions for STLength

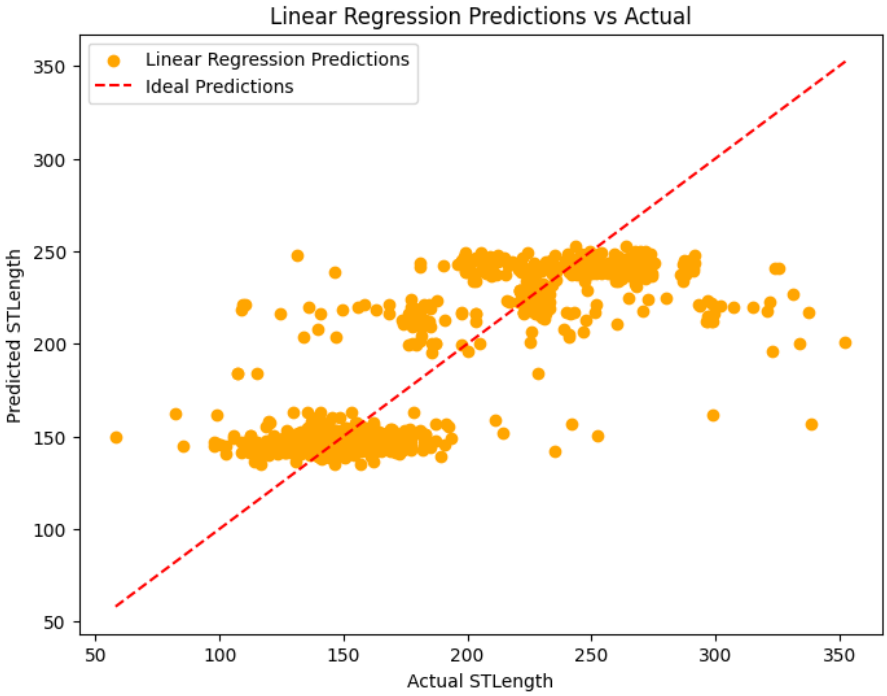


Figure 6.9: Plot of linear regression predicted values vs true values for STLength

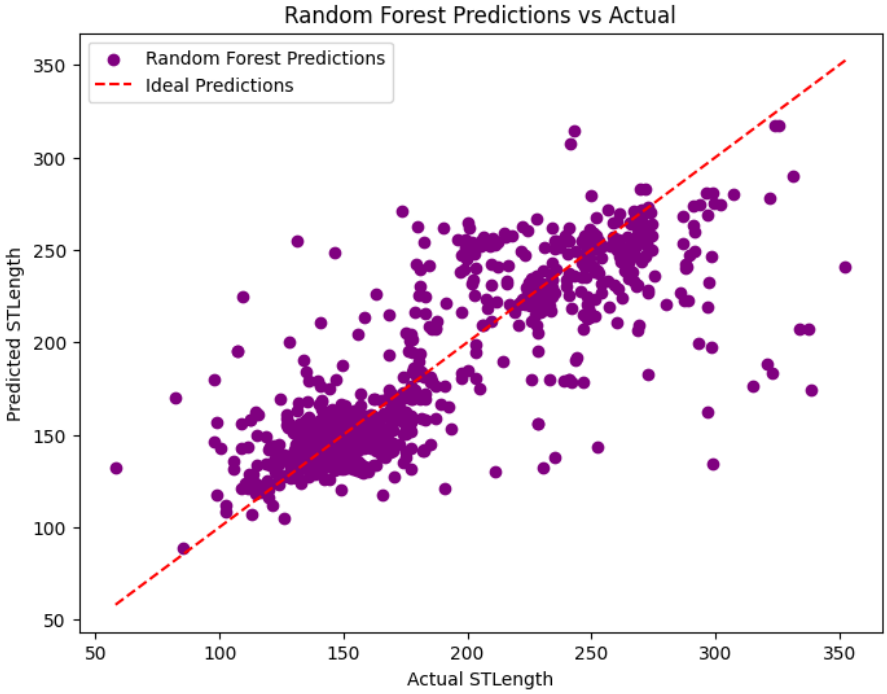


Figure 6.10: Plot of random forest regression predicted values vs true values for STLength

So we can see from the above graphs that Random Forest Regression can fit or predict the unseen values or test values of STLengt more accurately than Linear Regression.

6.2 Classification Problem

In our analysis, as discussed earlier we have many bool datatype variables.So as we know bool has 2 values either false or true .So it can be classified into 2 classes or we can say it can be classified into discrete classes . So here we can successfully implement the classification problem. So our query is as follows :

6.2.1 Predicting the column called BASKETBALL of unseen data(test data) which is of boolean datatype.

So here in our analysis we didn't perform the feature selection and we have directly fitted various classification models .We have included Logistic Regression , Random Forest Classifier and Support Vector Machine Classifier and after we have calculated all the evaluation metric of Classification problem such as accuracy,precision,recall and F1 score.

So let's first take a overview of **Logistic Regression,Random Forest Classifier** and **SVM classification** :

1.) Logistic Regression :

- A logistic regression models the relationship between the input features and the target as the logistic function, or the sigmoid function.
- In particular, the logistic function scales any real-valued input to the range 0, 1, where 0 represents the negative class ; 1 represents the positive class.
- Specifically, during training, we learn the coefficients of the logistic function, which we use to draw the likelihood of each instance belonging to class.
- At last, we classify each instance to class based on their probabilities using a threshold,such as 0.5.

2.) Random Forest Classifier :

- Random Forest Classifier builds a collection of decision trees in this practice throughout the fit stage, each of which is constructed on a random subset of training data and a random subset of characteristics.
- The ultimate forecast is established by accumulating the forecast of all the individual trees . In most circumstances, a voting system is used for classification difficulties.

3.) SVM Classification Technique :

- The SVM selects the optimal hyperplane for separating several data points into classes.
- The hyperplane is enforced to maximize the margin, which is the space between the hyperplane and the nearest data points of multiple categories that are known as support vectors .
- These models can create linear boundaries between classes and nonlinear by multiple kernels, such as linear, polynomial, radial basis function , and the sigmoid kernel.

☞ So afterwards we have counted the accuracy , precision, recall and F1 score of all 3 models.This is the code and output :

```
from sklearn.metrics import roc_curve, auc
target_variable = 'BASKETBALL'
X = df_encoded.drop(target_variable, axis=1)
y = df_encoded[target_variable]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature Engineering (You can add more feature engineering steps here)

# Feature Selection (You can add feature selection techniques here)

models = {
    'Logistic Regression': LogisticRegression(),
    'Random Forest Classifier': RandomForestClassifier(),
    'Support Vector Machine': SVC()
}

results = {}

for model_name, model in models.items():
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    results[model_name] = {'Accuracy': accuracy,'Precision': precision,'Recall': recall,'F1 Score': f1}

for model_name, scores in results.items():
    print(f"Model: {model_name}")
    print(f"Accuracy: {scores['Accuracy']:.4f}")
    print(f"Precision: {scores['Precision']:.4f}")
    print(f"Recall: {scores['Recall']:.4f}")
    print(f"F1 Score: {scores['F1 Score']:.4f}")
    print()
```

Figure 6.11: Code of different Classification Techniques with target variable named Basketball.We can see that Random Forest Classifier has good evaluation metric as compared to SVM and logistic regression

```
Model: Logistic Regression
Accuracy: 0.9723
Precision: 0.9262
Recall: 0.9718
F1 Score: 0.9485

Model: Random Forest Classifier
Accuracy: 0.9750
Precision: 0.9269
Recall: 0.9824
F1 Score: 0.9538

Model: Support Vector Machine
Accuracy: 0.9695
Precision: 0.9169
Recall: 0.9718
F1 Score: 0.9436
```

Figure 6.12: Output of evaluation metric of all 3 classification techniques used.We can see that Random Forest Classifier has good evaluation metric as compared to SVM and logistic regression

So as we can see following results:

- For example, the Random Forest Classifier demonstrated the best accuracy on the test dataset, 0.975, with Logistic Regression being slightly behind with its achievement of 0.9723 and Support Vector Machine having the result of 0.9695.
- Although the accuracies are not absolutely the same, all three of them are relatively high, demonstrating the capability of making correct predictions by using the development of the test data.
- Precision, on the other hand, explains the proportion of actual positive predictions within all the positive outcomes predicted by the model.
- In this case, the Random Forest Classifier and Logistic Regression have slightly relatively same precision of 0.9269 and 0.9262, respectively, whereas the Support Vector Machine showed the lowest result of 0.9169.
- This implies that the SVM model demonstrates a higher rate of false positives determined by false negatives, as well.
- As for the definition of recall, it would be better to introduce this metric as the proportion of actual positive predictions within all the positive instances in the dataset.
- In this case, the Random Forest Classifier gained the highest result of 0.9824, thus being the most effective in foresighting the outcomes.
- Lastly, one could consider the F1 Score, which is harmonic mean of precision and recall. In this case, the Random Forest Classifier got the best result of 0.9538, while the Logistic Regression and Support Vector Machine got 0.9485 and 0.9436. Thus, they all seem useful, especially given the imbalanced nature of the datasets.

So that was all about model fitting !

Chapter 7. Conclusion & future scope

In conclusion, We have performed a detailed Exploratory Data Analysis on the Athletic Facilities dataset. The analysis has facilitated the understanding of the distribution, characteristics, and the status of the athletic facilities.

Such information may be helpful in the area of data-driven decision-making with regard to matters such as urban planning, administration, and policy making. More specifically, one of the main aims of urban planning is ensuring that communities, cities, and towns become more amenable to an active and healthy lifestyle which can be done by understanding the current situation of Athletic Facilities.

However, our analysis provides great insights, future scope for this could be including:

- Spatial Analysis - Spacial analysis techniques could be used to know better the geographical distribution of athletic facilities and recognize areas with inadequate access to facilities.
- Community Engagement - To understand the preferences and opinions of the residents regarding the provision of such facilities and locations desired.
- Impact Assessment - Study can be conducted to analyze the impact of the facilities on the health, physical and social life of the residents.

7.1 Findings/observations

Key findings of the analysis includes:

- Our dataset originally had 6897 rows and 44 columns.
- The cleaning and preprocessing of the data was done by first visualizing(using missing no package) and then removing the missing values (imputation techniques) and same was done for outliers. The missing data was of MCAR type.
- Most of the facilities were concentrated in the departments under 3 major regions. Although nearly a seventh of them are non-operational.
- Asphalt is the most widely used material for the facility surfaces.
- Feature Selection techniques : Mutual Information Feature Selection Method was used for STArea and top 20 percentile with high mutual information were selected. Total of 44 features were chosen.
  Variance Threshold Selection Method : This technique was applied to STLength and Threshold of 0.1 was chosen. Total of 19 features were obtained through this method.

- Handball came out to be the most popular sport.
- Random Forest Regression was better as compared to Linear Regression for predicting STArea and STLength.
  Random Forest Classifier gave best results as compared to Logistic Regression and SVM for classifying the column BASKETBALL.

7.2 Challenges

During the course of this project few challenges that were encountered were-

- Finding the dataset- It was hard to get the perfect dataset as it had to be relevant in terms of scope, detailed enough, and up-to-date.
- Data Interpretation- The interpretation of some features like borough, etc. needed knowlegde of the field. Thus, accurate interpretation was paramount to allow meaningful conclusions.
- Data cleaning- The dataset contains strings which requires careful handling. Cleaning of such a dataset can be a little challenging.

Despite these challenges, our project was succesfull in covering valuable information and insights for the athletic Facilities.

7.3 Future plan

We will maintain and keep updating the dataset by continuing to gather data and enhance our analysis. Exploring additional data sources will also help us go more in this direction.

We will also explore advanced analysis techniques including predictive modelling to go more deeper into our analysis.

Group Contribution

All the three members have contributed equally to the EDA Final Project. Total Work was divided into three parts and each member was given a certain time period to complete the task. In this way the project was carried out phase wise within one month.

Member 1

Introduction and how big is the data (shape,etc.), Visualisations

Member 2

Model Training, Statistical Analysis and Data Cleaning

Member 3

Defining Problems and other queries, Data Collection, Feature Engineering, Data Transformation

Short Bio

1. **Keerrtivardhan Goyal** I am Keerrtivardhan Goyal(ID: 202103007) from B.Tech MnC branch in 3rd Year currently studying in DAIICT. I have had an interest in mathematics since childhood and later developed interest in computers. This led to me choosing for an engineering course for my graduate studies.

I like football, badminton and reading novels. I am proficient in Python, Jupyter Notebook and LaTex and some grasp over Excel. I am have an interest in Data Analytics and hope to make a career in it.

2. **Yash Mashru** I am Yash Mashru from MnC batch 2021,student ID 202103045. I am very much passionate and have a keen interest in the field of mathematics from my childhood and that's why I have taken the Mathematics and Computing branch in DAIICT. My hometown is in Jamnagar,Gujarat.

I like playing chess, basketball,cricket and football and have a keen interest in singing and dancing too.

In my school days , I have participated in various cultural activities like Drama, Elocution,Declamation,Debate etc.

I have won various state level prizes in the Chess Under 17 men's category and I have been also awarded the best basketball player in our district Jamnagar.

3. **Sanchit Satija** I am Sanchit Satija(ID: 202103054) from B.Tech MnC branch in 3rd Year currently studying in DAIICT. I am from Gurgaon, Haryana. Some of my hobbies are exploring different places, playing cricket and going to the gym.

I am familiar with many softwares such Google data studio, Jupyter Notebook, LaTeX, Python, R, Spark, Excel.

I have contributed to various open source projects. I am excellent in debating skills as well. I have won various medals in Athletics during my school life.

References

[1] Official Data Catalog Website of USA. *URL:* https://catalog.data.gov/dataset

[2] Little MCAR Test. *URL:* https://stackoverflow.com/questions/58144737/mcar-littles-test-in-python