Data Visualization using Python with special reference to Matplotlib and Seaborn

Reny Jose, Assistant Professor Kochumol Abraham, Assistant Professor MCA Department MCA Department

Marian College Kuttikanam Autonomous Marian College Kuttikanam Autonomous

Kuttikanam, India Kuttikanam, India

reny.jose@mariancollege.org kochumol.abraham@mariancollege.org

Win Mathew John, Associate Professor

MCA Department

Marian College Kuttikanam Autonomous

Kuttikanam, India

win.mathew@mariancollege.org

## ABSTRACT

In the [contemporary world](https://thesaurus.yourdictionary.com/contemporary-world), a lot of data is being generated per diem. The extraction of information from this data is a tedious process. To represent this information different methods are available in the real-world scenario. To produce an impact on the viewers it is always better to have a visualization of these data. Data visualization is the process of representing the analyzed data or processed data through charts, graphs, animations etc. To convey an idea or to represent a policy we can use visualizations. The importance of visualization increases as the visual impact on humans increases. Our eyes are more lenient with colors and patterns. Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends, and correlations that might not otherwise be detected can be exposed. Data visualization provides a good, organized pictorial representation of the data which makes it easier to understand, observe, and analyze.

**Keywords**— Visualization; python; matplotlib; Seaborn

## I. INTRODUCTION

With the advancement of technology, the collection and storage of data has become much easier. The Analysis of data becomes intricate if data exists in its raw format. Here comes the importance of data visualization. Visualization provides an organized pictorial representation of data which makes it easier to understand, observe, and analyze. Python provides various libraries that come with different features for visualizing data. All these libraries come with different features and can support various types of graphs. Here, we will be discussing two such libraries Matplotlib and Seaborn. This paper focuses on creating basic plots using Matplotlib and Seaborn as well as how to use some specific features of each library. Here we focus on the syntax and not on the interpretation of the graphs.

**II. PYTHON PACKAGES FOR DATA VISUALIZATION**

Python is a simple, dominant, and well-organized interpreted language. Python is used to develop a very high performance scientific related application and it is used to develop an application for Numeric computation, it is frequently used. Using python we can develop high-performance applications. Python often utilizes external libraries to perform scientific computing. The most important libraries used are NumPy, SciPy, and Matplotlib to perform scientific and numeric applications. The Python libraries are open-source tag-on modules, which do additional frequent mathematical and numerical routines in pre-compiled, high-speed tasks.

## III. DATA SET

The dataset used in this chapter is Big Mart Sales from Kaggle [1]. The dataset consists of 2013 sales data for 1559 products across 10 stores in different cities. The main attributes are [2]

* Item\_Identifier: Unique product ID,
* Item\_Weight: Weight of the product
* Item\_Fat\_Content: Whether the product is low fat or not
* Item\_Visibility: The % of the total display area of all products in a store allocated to the particular product
* Item\_Type: The category to which the product belongs
* Item\_MRP: Maximum Retail Price (list price) of the product
* Outlet\_Identifier : Unique store ID
* Outlet\_Establishment\_Year: The year in which the store was established
* Outlet\_Size: The size of the store in terms of ground area covered
* Outlet\_Location\_Type: The type of city in which the store is located
* Outlet\_Type: Whether the outlet is just a grocery store or some sort of supermarket
* Item\_Outlet\_Sales: Sales of the product in the particulate store. This is the outcome variable to be predicted.

## IV. VISUALIZATION USING MATPLOTLIB

Matplotlib is a Python package for creating simple and complex plots with few lines of code. It helps in generating quality graphs required for scientific and numeric plotting. Matplotlib is the most extensively used library of python for data visualization due to its high flexibility and extensive functionality that it provides. It is designed to create simple and complex two-dimensional plots. It’s an amazing multi-platform visualization library designed to work with a broader SciPy stack. Several plots like lines, bars, scatter, histograms, etc. can be created using this library. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt. It can be used in Python, IPython shells, Jupyter notebook, and web application servers. Matplotlib is open source and no license servers are required. Here we focus on the syntax and implementation of four different charts as Line chart, Bar chart, Histogram, and Box plot. Here we have taken the Big Mart Sales dataset from GitHub as the use case for plotting. It’s a regression practice problem wherein they predict sales product-wise and store-wise.

## LINE CHART

## Line charts are used to show trends in data by plotting data points connected with a line. Matplotlib allows a number of different formatting options on charts. To make it more informative, axis labels, chart title, and markers are there for data points on the chart.

**CODE**

# Importing matplotlib

import matplotlib.pyplot as plt

# read the dataset

data\_BM = pd.read\_csv('bigmart\_data.csv')

# drop the null values

data\_BM = data\_BM.dropna(how="any")

# view the top results

data\_BM.head()

# Mean price based on item type

price\_by\_item = data\_BM.groupby ('Item\_Type').Item\_MRP.mean() [:10]

x = price\_by\_item.index.tolist()

y = price\_by\_item.values.tolist()

# set figure size

plt.figure(figsize=(14, 8))

# set title

plt.title('Mean price for each item type')

# set axis labels

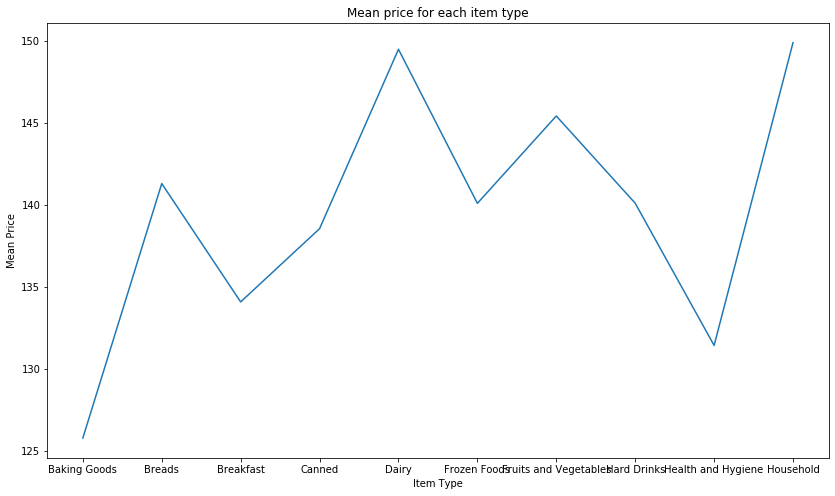
plt.xlabel('Item Type')

plt.ylabel('Mean Price')

# set xticks

plt.xticks(labels=x, ticks=np.arange(len(x)))

plt.plot(x, y)



**Figure 1: Line chart to denote the mean price per item**

## B. BAR CHART

A bar chart is a simple type of visualization that is used for categorical variables. Assume that we need to determine the mean sales for each outlet type. In such cases, we can use bar diagrams to plot the category.

**CODE**

# sales by outlet size

sales\_by\_outlet\_size = data\_BM.groupby('Outlet\_Size').Item\_Outlet\_Sales.mean()

# sort by sales

sales\_by\_outlet\_size.sort\_values(inplace=True)

x = sales\_by\_outlet\_size.index.tolist()

y = sales\_by\_outlet\_size.values.tolist()

# set axis labels

plt.xlabel('Outlet Size')

plt.ylabel('Sales')

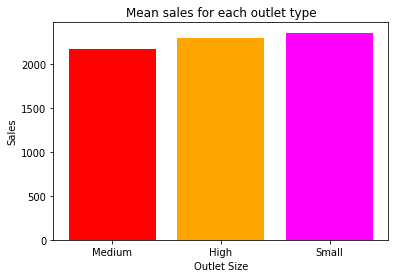
# set title

plt.title('Mean sales for each outlet type')

# set xticks

plt.xticks(labels=x, ticks=np.arange(len(x)))

plt.bar(x, y, color=['red', 'orange', 'magenta'])



**Figure 2: Bar chart to denote the mean sales for each outlet type**

## C. HISTOGRAM

Histograms are a very common type of plotting method when we focus on data like height and weight, stock prices, waiting time for a customer, etc which are continuous in nature. The histogram’s data is plotted within a range against its frequency. Histograms are very commonly occurring graphs in probability and statistics and form the basis for various distributions like the normal -distribution, t-distribution, etc.  plt. hist() is used to draw a histogram. It provides many parameters to adjust the plot. Here we use the histogram to represent the Distribution of prices.

# title

plt.title('Item MRP (price) distribution')

# xlabel

plt.xlabel('Item\_MRP')

# ylabel

plt.ylabel('Frequency')

# plot histogram

plt.hist(data\_BM['Item\_MRP'], bins=20, color='lightblue')



**Figure 3: Histogram to denote the Distribution of prices**

## D. BOX PLOT

Box plot shows the three quartile values of the distribution along with extreme values. The whiskers go from each quartile to the minimum or maximum and then observations that fall outside this range are displayed independently. This means that each value in the boxplot corresponds to an actual observation in the data. Let's try to visualize the distribution of the outlet sales of Items.

**CODE**

data = data\_BM[['Item\_Outlet\_Sales']]

# create outlier point shape

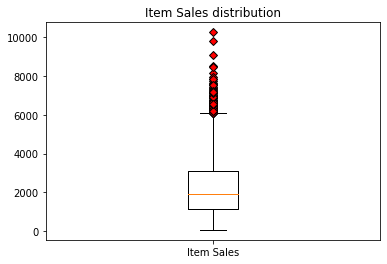
red\_diamond = dict(markerfacecolor='r', marker='D')

# set title

plt.title('Item Sales distribution')

# make the boxplot

plt.boxplot(data.values, labels=['Item Sales'], flierprops=red\_diamond);



**Figure 4: Box Plot to denote the distribution of Outlet Sales of items.**

## V. VISUALIZATION USING SEABORNE

Seaborn, the most widely used visualization library in the python library for statistical data visualization plots. An API is used to create plots in a timely manner and provides a variety of color palettes and statistical styles [3]. It works based on matplotlib and provides a high-level interface for drawing statistical graphics by selecting plot t style and color defaults defining simple high-level functions for common statistical plot types, and integrating with the functionality provided by Pandas Data Frames. Using Seaborn we can represent the data stored in an array, list, or any data structure in graphical form. No need to understand the internal details, just pass the data, it will automatically calculate the values and plot the corresponding graphs. The main idea is to explore statistical data and model fitting. It is a useful tool for financial analysts, business analysts, data analysts, data scientists, etc. [4].

**A.COUNT PLOT**

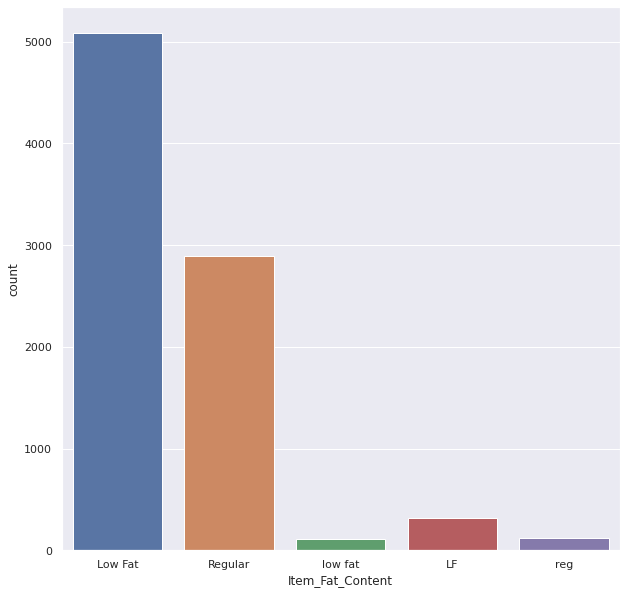
A count plot is useful to deal with categorical values by plotting the frequency of the different categories. The column Item Fat Content contains categorical data [4].

**CODE**



**Figure5: Python code for count plot**

In figure 5 we import the library and in line two we read the dataset to the variable called data\_BM. On line three is the code for the count plot whose basic parameter is the x value for the categorical attribute, here it is Item\_Fat\_Content and data is the corresponding dataset.



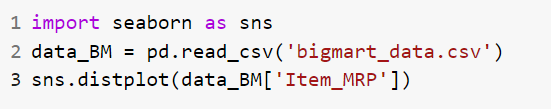
**Figure 6: Count Plot Based Fat Content Of An Item**

We can observe from the graph that the number of items having low fat is significantly higher than the number of regular fat items.

**B. HISTOGRAM**

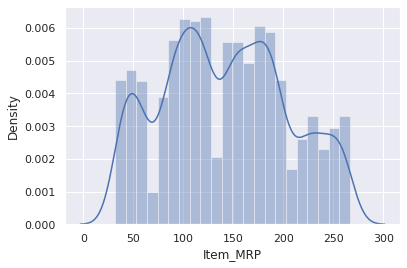
Histograms are used to represent the distribution of a set of continuous data by forming bins along the range of the data in the X-axis.[5] It displays bars to show the number of observations in each bin. In Seaborn, we create a histogram using the distplot().

**CODE**



**Figure 7: python code to display the histogram**

In figure 7 we import the library and in line two we read the dataset to the variable called data\_BM. On line three is the code for a histogram. We use distplot one attribute, here it is Item\_MRP from the corresponding dataset.



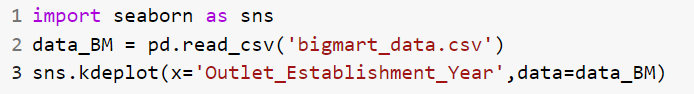
**Figure 8: Histogram-Based Item MRP**

We can observe from the graph that items having MRP is having the highest peak and items having MRP between 150 and 200t is in the next highest peak. That means the majority of items are in the range between the MRP 100 and 200.

**C.KDE PLOT**

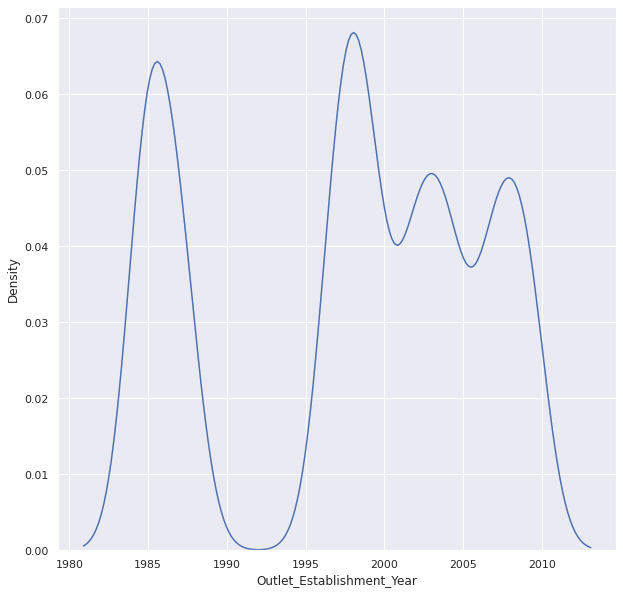
A Kernel Density Estimate (KDE) Plot is used to show the distribution of continuous data. It depicts the probability density function of the continuous or non-parametric data variables[6]. . It can be considered as a smoothed histogram

**CODE**



**Figure9: Python Code For KDE Plot**

In figure 9 we import the library and in line two we read the dataset to the variable called data\_BM. On line three is the code for the KDE plot whose basic parameter is the x value for the column name, here it is Outlet\_Establishment \_Year and data is the corresponding dataset.



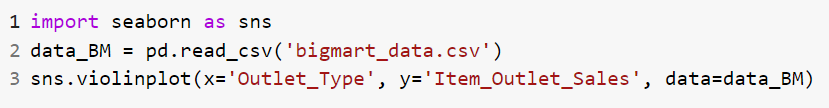
**Figure 10: KDE Plot Based On Above Code**

The peak of the above graph is between 1980 to 1990 and 1995 to 2000 so we can conclude that most outlets are established between 1995 to 2000 and next is between 1980 and 1990.

**D. VIOLIN PLOT**

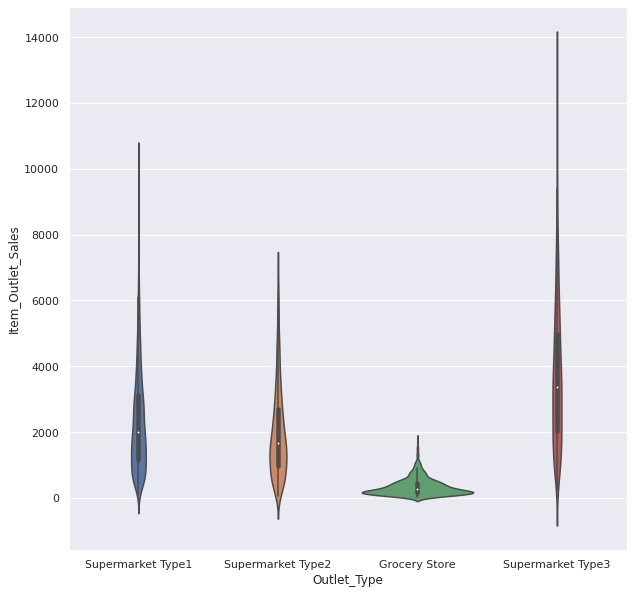
A violin plot is similar to a box and whisker plot. It is used to compare the distribution of quantitative data on one (or more) categorical variables. It features a kernel density estimation of the underlying distribution.[7] To create a violin plot in Seaborn using the violin plot()[8].

**CODE**



**Figure 11. Python Code For Violin Plot**

In figure 11. we import the library and in line two we read the dataset to the variable called data\_BM. On line three is the code for violin plot whose basic parameter is the x value for the column name, here it is Outlet\_Type, d y value for column attribute, here it is Item\_Outlet\_Sales and data is the corresponding dataset.



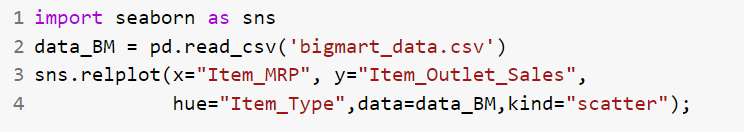
**Figure 12: Violin Plot Based On Outlet Type and Sales**

We can observe from the graph that sales in the different outlets are in the range 0 to 14000. The supermarket type 3 has the largest sales. The average sales of each outlet is between 1000 and 15000. The grocery store is having fewer sales. The small white dot in the middle shows the median value

**E. SCATTER PLOT**

Scatter Plot depicts the distribution of two variables using a cloud of points, where each point represents an observation in the dataset[9]. This depiction allows the eye to infer a substantial amount of information about whether there is any meaningful relationship between them. In Seaborn, we use replot() with the option of kind=scatter to plot a scatter plot.

**CODE**



**Figure13: Python Code for Scatter Plot**

In figure 13 we import the library and in line two we read the dataset to the variable called data\_BM. On line three is the code for scatter plot whose basic parameter is the x value for the column name, here it is Item\_MRP, y value for column attribute, here it is Item\_Outlet\_Sales, hue is used to plot by coloring the points according to a third variable, here we use Item\_Type, data is the corresponding dataset and kind is a type of plot, here it is scatter.



**Figure 14. Scatter Plot Based On Sales MPR And Item Types**

We can observe from the graph that fruits and vegetables is having the maximum sales. Items having MRP between 150 and 259 have average sales between 2000 and 6000. Items having less MRP are having fewer sales.

**F. STRIP PLOT**

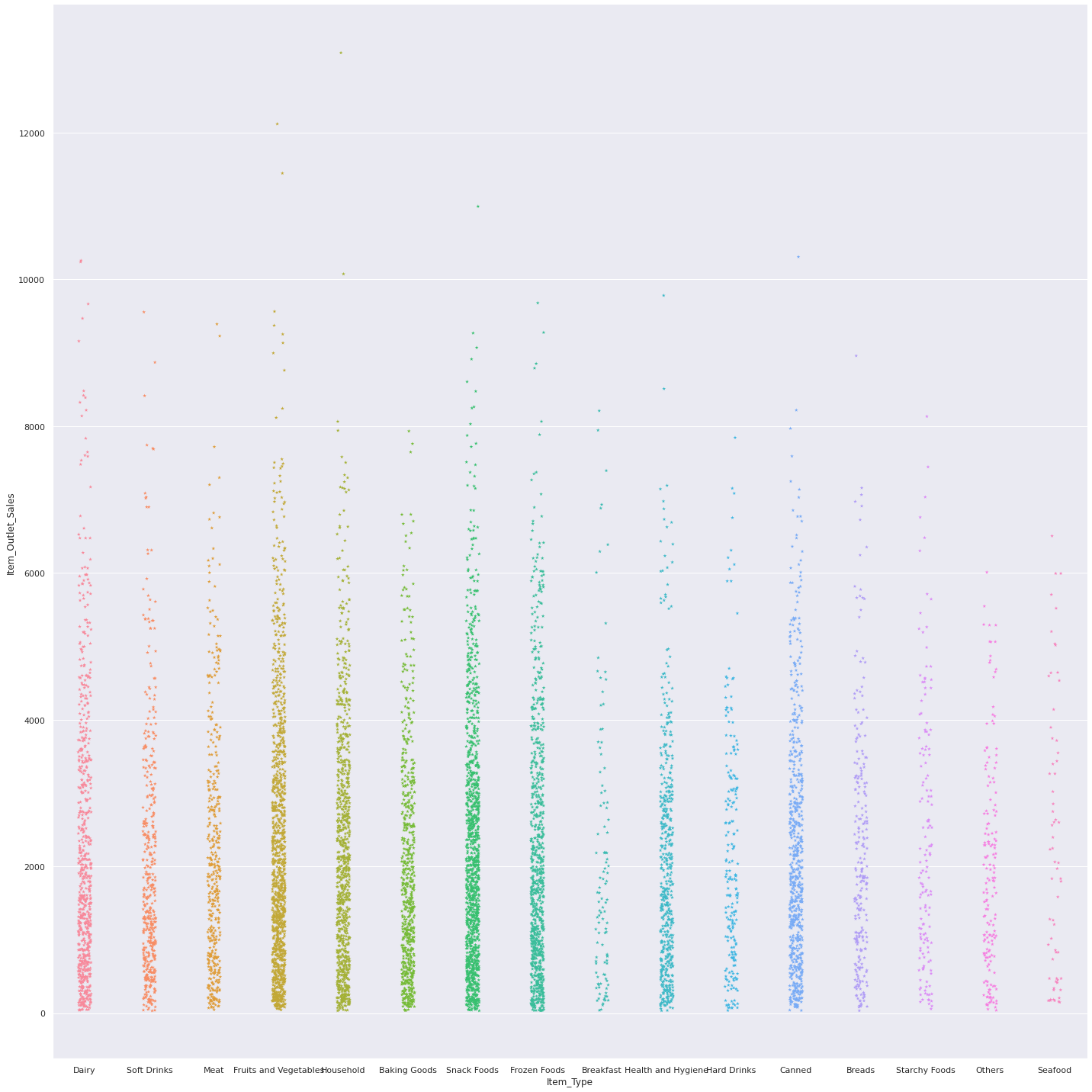
A strip plot is a scatter plot depending on a category. In a single graph, all the observations and collected data that are visualized are shown, side-by-side. It is a good alternative to a box plot or a violin plot[10]. Using catplot(), we can create this by passing kind=strip [11].

**CODE**



**Figure15: Python Code For Strip Plot**

In figure 15 we import the library and in line two we read the dataset to the variable called data\_BM. On line three is the code for striper plot whose basic parameter is the x value for the column name, here it is Item\_Type, y value for column attribute, here it is Item\_Outlet\_Sales, the marker is the shape of the point, and kind is a type of plot, here it is strip, size is the size of the figure, data is the corresponding dataset.



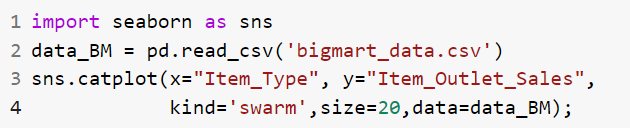
**Figure 16. Strip Diagrambased On Sales And Item Type**

We can observe from the graph that only fruits and vegetables and snacks have above 10000 sales

**G. SWARM PLOT**

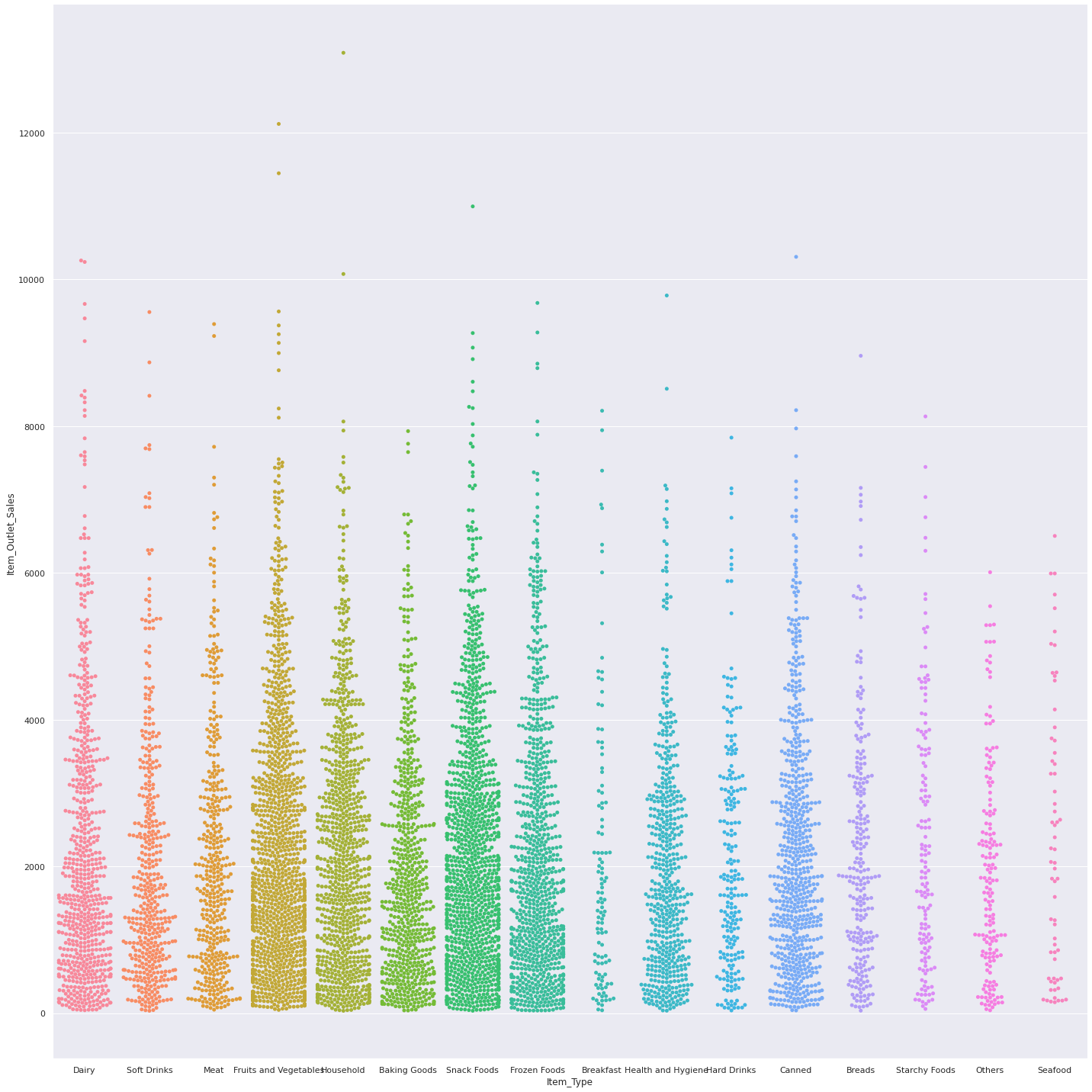
A swarm plot is a type of scatter plot that is used for representing categorical values. It is very similar to the strip plot, but it avoids the overlapping of points but the points are adjusted (only along the categorical axis) [12]. This gives a better representation of the distribution of values, but it does not scale well to large numbers of observations. This style of the plot is sometimes called a “beeswarm”.[13] You can create this by passing kind=swarm in the catplot().

**CODE**



**Figure 17: Python Code For Swam Plot**

In figure 17 we import the library and in line two we read the dataset to the variable called data\_BM. On line three is the code for striper plot whose basic parameter is the x value for the column name, here it is Item\_Type, y value for column attribute, here it is Item\_Outlet\_Sales, kind is a type of plot, here it is a swarm, size is the size of the figure, data is the corresponding dataset.



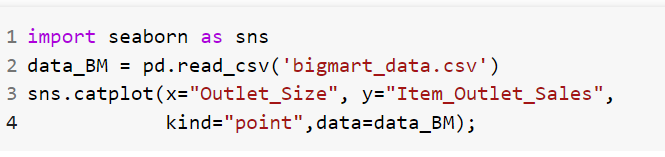
**Figure 18: Swarm Plot Based On Sales And Item Type**

We can observe from the graph about the distribution of data we get a clear idea about the number of items sold in each category i.e. item type. Between 0 and 2000 the maximum sales happen items like fruits & vegetable and snacks and minimum for seafood.

**H. POINT PLOT**

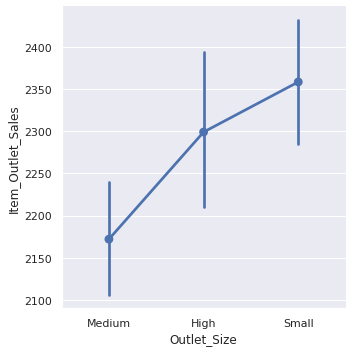
A point plot represents an estimate of central tendency[14]. It provides some indication of the uncertainty around that estimate using error bars. A point plot uses scatter plot glyphs to visualize features like point estimates and confidence intervals.[15]

**CODE**



**Figure 19: Python Code for Point Plot**

In figure 19 we import the library and in line two we read the dataset to the variable called data\_BM. On line three is the code for catplot for point plot whose basic parameter is the x value for the column name, here it is Outlet\_size, y value for column attribute, here it is Item\_Outlet\_Sales, kind is a type of plot, here it is a point, data is the corresponding dataset. There is a line joining each point from the same level of the hue level which shows the interaction between two categorical variables which is easier for us than comparing the heights of several groups of points or bars.



**Figure 20: Point Plot based On Outlet Size And Sales**

In the point plot above, the dot markers are placed against the mean number of sales2175,2300 and,2360) in different outlets based on size(medium, high and small). The error bars denoting the standard deviations are drawn above and below these dot markers.

**VI. CONCLUSION**

In our application, we have a large amount of data that needs to be represented to the user in some readable and undertakable format, so Seaborn and Matplotlib helps the user visualize data in the form of graphics which makes it attractive and efficient. It enhances the exploration process. Matplotlib and Seaborn to powerful libraries in python for data visualization [16]. In this chapter we covered how to implement Matplotlib plotting using line plots, bar charts, histograms, scatter plots, etc, and using the Seaborn strip plots, box plots, and swarm plots, by explaining the code and diagram. If a data scientist wants to visualize large chunks of datasets then Seaborn will be a better option, but if you are looking for basic visualization patterns then matplotlib would be a better choice for beginners and starters in the field of data visualization & computational modeling. [17]

## REFERENCES

[1] “Big Mart Sales | Kaggle.” [Online]. Available: https://www.kaggle.com/datasets/mrmorj/big-mart-sales. [Accessed: 14-Aug-2022].

[2] “GitHub - akki8087/Big-Mart-Sales.” [Online]. Available: https://github.com/akki8087/Big-Mart-Sales. [Accessed: 14-Aug-2022].

[3] M. Waskom, “seaborn: statistical data visualization,” *J. Open Source Softw.*, vol. 6, no. 60, p. 3021, Apr. 2021.

[4] “How to use Seaborn for Data Visualization | Engineering Education (EngEd) Program | Section.” [Online]. Available: https://www.section.io/engineering-education/seaborn-tutorial/. [Accessed: 14-Aug-2022].

[5] “Basic histogram with Seaborn.” [Online]. Available: https://www.python-graph-gallery.com/20-basic-histogram-seaborn. [Accessed: 14-Aug-2022].

[6] “Seaborn Kdeplot - A Comprehensive Guide | DigitalOcean.” [Online]. Available: https://www.digitalocean.com/community/tutorials/seaborn-kdeplot. [Accessed: 14-Aug-2022].

[7] “Details of Violinplot and Relplot in Seaborn – Regenerative.” [Online]. Available: https://regenerativetoday.com/details-of-violinplot-and-relplot-in-seaborn/. [Accessed: 14-Aug-2022].

[8] “seaborn.violinplot — seaborn 0.11.2 documentation.” [Online]. Available: https://seaborn.pydata.org/generated/seaborn.violinplot.html. [Accessed: 14-Aug-2022].

[9] “Scatterplot.” [Online]. Available: https://www.python-graph-gallery.com/scatter-plot/. [Accessed: 14-Aug-2022].

[10] “The Seaborn stripplot() method in Python - A Quick Guide - AskPython.” [Online]. Available: https://www.askpython.com/python-modules/seaborn-stripplot-method. [Accessed: 15-Aug-2022].

[11] “What is seaborn.stripplot?” [Online]. Available: https://www.educative.io/answers/what-is-seabornstripplot. [Accessed: 15-Aug-2022].

[12] “data\_visualization\_using\_seaborn 0.0.1 Data Visualization using Seaborn (a Python library) Tutorial by: Navie Narula, Digital Centers Teaching Intern Created for: Research Data Services at Columbia University Libraries Resources used to create tutorial: DataCamp’s Introductory Tutorial Pandey’s Visualization Examples Seaborn Py-Data Swarm Plots Seaborn PyData Heat Maps List of Colors in Python,” 2018.

[13] “The Ultimate Python Seaborn Tutorial: Gotta Catch ’Em All.” [Online]. Available: https://elitedatascience.com/python-seaborn-tutorial. [Accessed: 15-Aug-2022].

[14] “Python Seaborn Categorical estimate plots: Point Plot.” [Online]. Available: https://www.programsbuzz.com/article/python-seaborn-categorical-estimate-plots-point-plot. [Accessed: 15-Aug-2022].

[15] “Drawing a Point Plot using Seaborn | Pythontic.com.” [Online]. Available: https://pythontic.com/visualization/seaborn/pointplot. [Accessed: 15-Aug-2022].

[16] S. K. A. Fahad and A. E. Yahya, “Big Data Visualization: Allotting by R and Python with GUI Tools,” *2018 Int. Conf. Smart Comput. Electron. Enterp. ICSCEE 2018*, Nov. 2018.

[17] “Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python Related papers.”