DATA VISUALIZATION USING PYTHON WITH SPECIAL REFERENCE TO MATPLOTLIB AND SEABORN

Abstract

In the contemporary world, enormous data is 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. Representing the analyzed data or processed data through charts, graphs and animations is known as data visualization. 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. Hence representing data in a visual form makes it easier to analyze. The significance of visualization is that correlations and trends that might not be detected can be easily recognized in an understandable format.

Keywords: Visualization; Python; Matplotlib; Seaborn

Authors

Reny Jose
Assistant Professor
PG Department of Computer Applications
Marian College Kuttikanam Autonomous
Kuttikanam, India
reny.jose@mariancollege.org

Kochumol Abraham
P G Department of Computer Applications
Marian College Kuttikanam Autonomous
Kuttikanam, India
kochumol.abraham@mariancollege.org

Win Mathew John
Associate Professor
PG Department of Computer Applications
Marian College Kuttikanam Autonomous
Kuttikanam, India
win.mathew@mariancollege.org

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 relevance of visualizing data. Visualization provides a systematic view which helps in easily analyzing and understanding data. 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 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. High-performance applications can be developed using Python. External libraries are used to perform scientific computing. Some 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.

Figure 1: List of attributes

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. One of the the most extensively used library of python for data visualization is Matplotlib 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 an open source module in which license servers are not 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.

1. **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.

```
# 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))
```

Figure 2: Code for Line Chart Data Preprocessing

```
# 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 3: Code for Line Chart Visualization

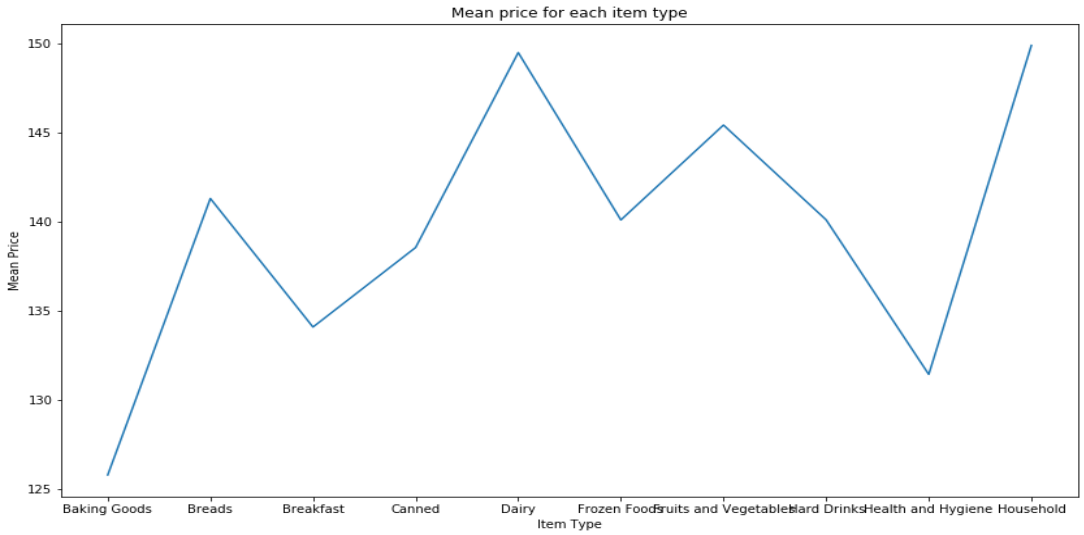

Figure 4: Line chart to denote the mean price per item

2. **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.

```
# 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()
```

Figure 5: Code for Preprocessing

```
# 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 6: Code for Bar Chart Visualization

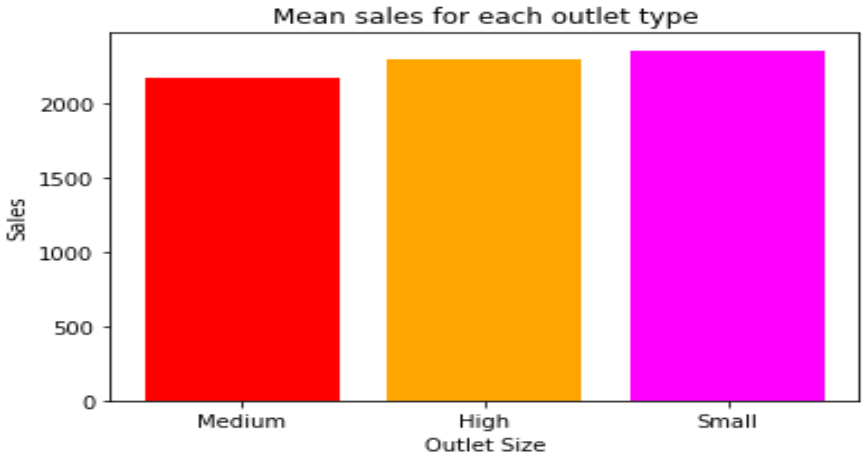

Figure 7: Bar chart to denote mean sales for each outlet type

3. **Histogram:** Histograms are a very common type of plotting method which is used to plot data which is continuous in nature. Here 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. The function '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 8: Code for plotting a histogram

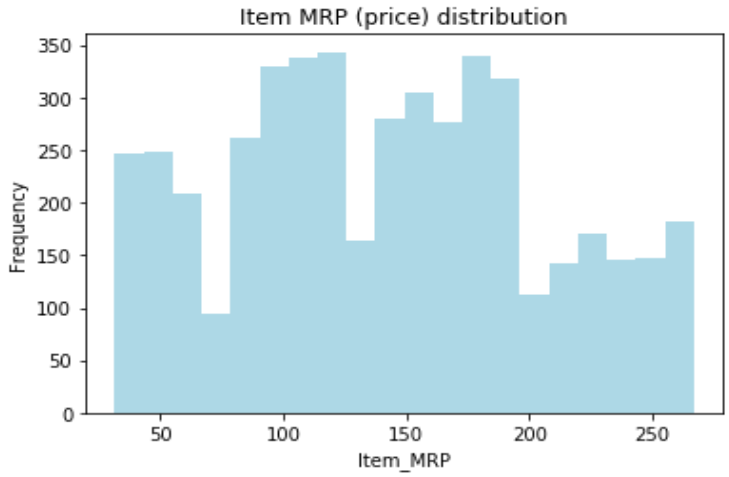

Figure 9: Histogram to denote the Distribution of prices

4. **Box plot:** A Box plot represents a five-number summary of a set of data. The five-number summary indicates minimum, first quartile, median, third quartile, and maximum. Here, we draw a box from the first quartile to the third quartile. Each value in the box plot corresponds to an actual observation in the data. Let's try to visualize the distribution of the outlet sales of Items.

```
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 10: Code for plotting a Box Plot

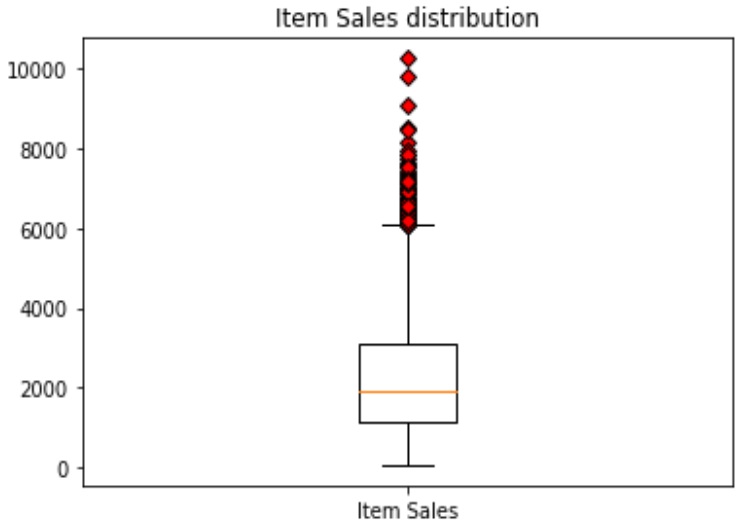

Figure 11: Box Plot to denote the distribution of Outlet Sales of items.

V. VISUALIZATION USING SEABORNE

Seaborn is a visualization library in python used for visualizing statistical data. An API is used to create plots in a timely manner and provides a variety of color palettes and statistical styles [3]. It is based on Matplotlib and provides an interface for drawing statistical graphics by selecting plot style and color and integrating with the functionality provided by Pandas Data Frames. The data stored in an array, list, or any data structure can be represented in graphical form using Seaborn. The main idea is to explore statistical data and model fitting. It is a useful tool for financial analysts, business analysts, data scientists, etc. [4].

1. **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

```
1 import seaborn as sns
2 data_BM = pd.read_csv('bigmart_data.csv')
3 sns.countplot(x='Item_Fat_Content',data=data_BM)
```

Figure 12: Python code for count plot

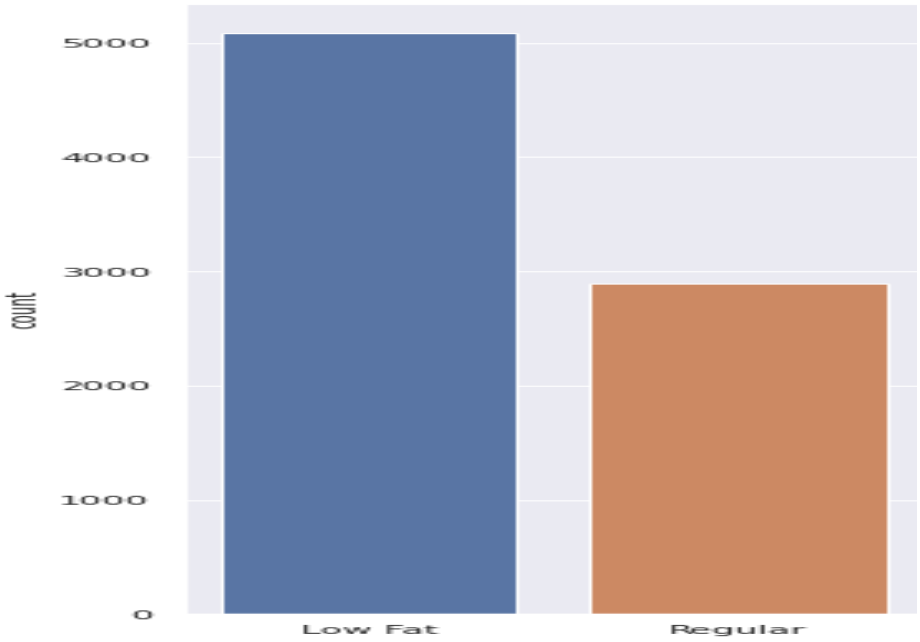

Figure 13: Count Plot-Based Fat Content of an Item

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

2. **Histogram:** Histograms are used to represent the distribution of a set of continuous data by forming bins along the range of the data on 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

```
1 import seaborn as sns
2 data_BM = pd.read_csv('bigmart_data.csv')
3 sns.distplot(data_BM['Item_MRP'])
```

Figure 14: python code to display the histogram

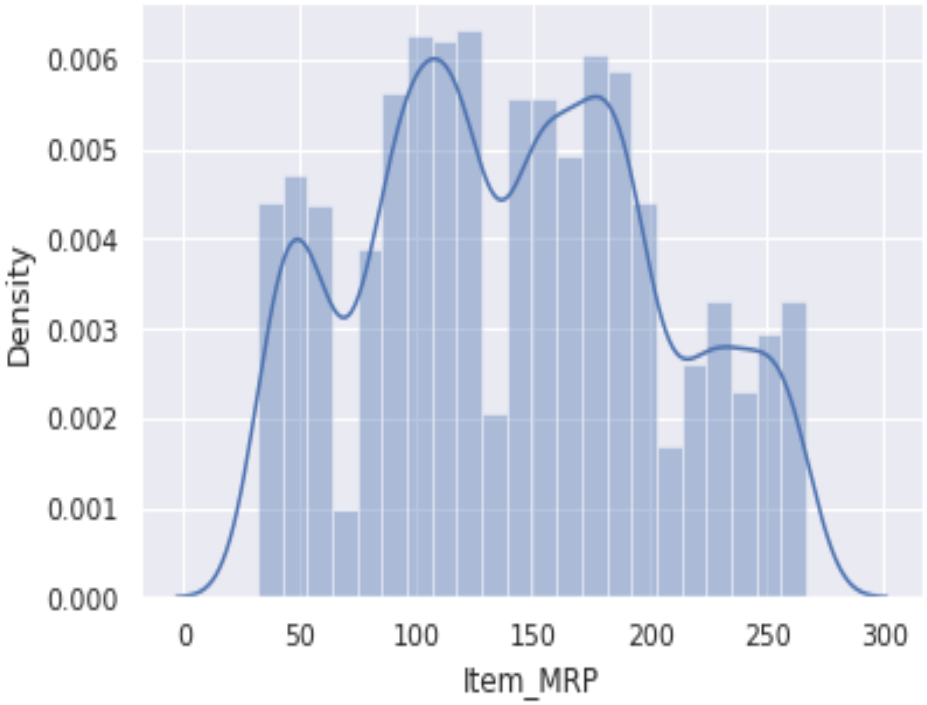

Figure 15: Histogram-Based Item MRP

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

3. **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

```
1 import seaborn as sns
2 data_BM = pd.read_csv('bigmart_data.csv')
3 sns.kdeplot(x='Outlet_Establishment_Year',data=data_BM)
```

Figure 16: Python Code for KDE Plot

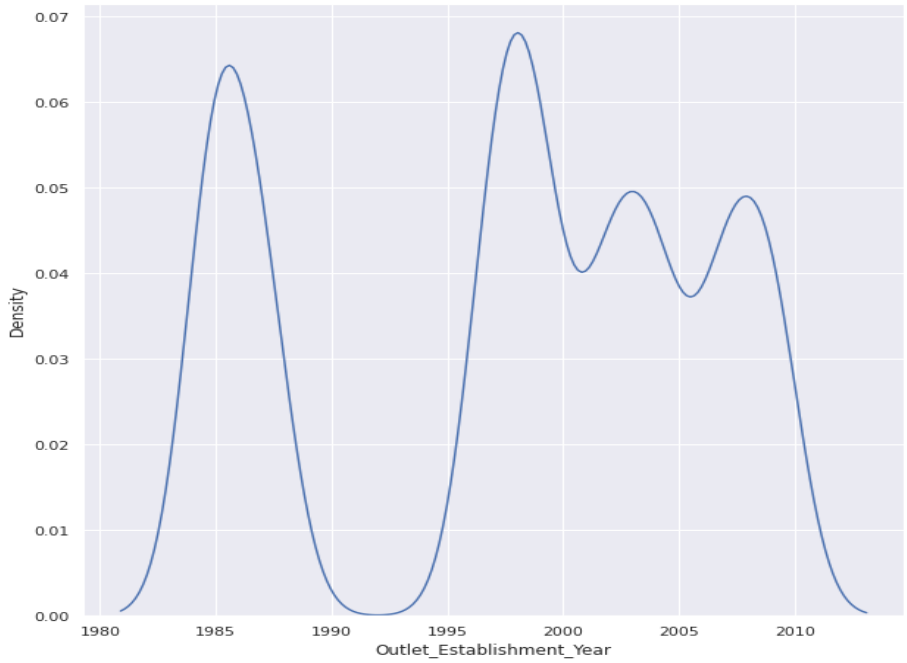

Figure 17: KDE Plot Based on Above Code

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.

4. **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]

Code

```
1 import seaborn as sns
2 data_BM = pd.read_csv('bigmart_data.csv')
3 sns.violinplot(x='Outlet_Type', y='Item_Outlet_Sales', data=data_BM)
```

Figure 18: Python Code for Violin Plot

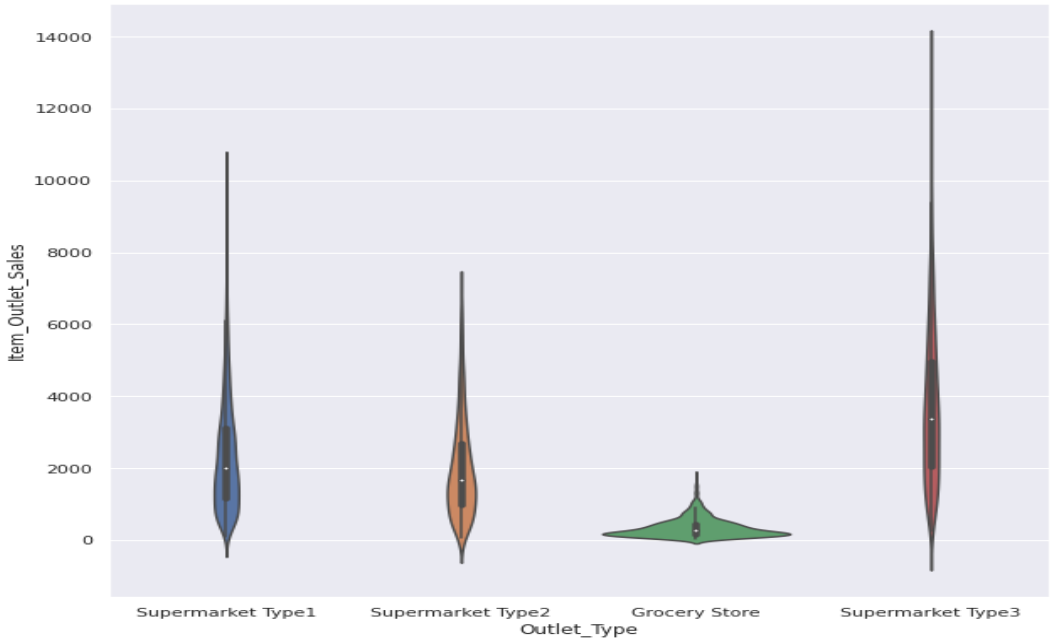

Figure 19: Violin Plot Based on Outlet Type and Sales

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

5. **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 plot helps to infer whether there is any meaningful relationship between data. In Seaborn, we use replot() with the option of kind=scatter to plot a scatter plot.

Code

```
import seaborn as sns
data_BM = pd.read_csv('bigmart_data.csv')
sns.relplot(x="Item_MRP", y="Item_Outlet_Sales",
            hue="Item_Type",data=data_BM,kind="scatter");
```

Figure 20: Python Code for Scatter Plot

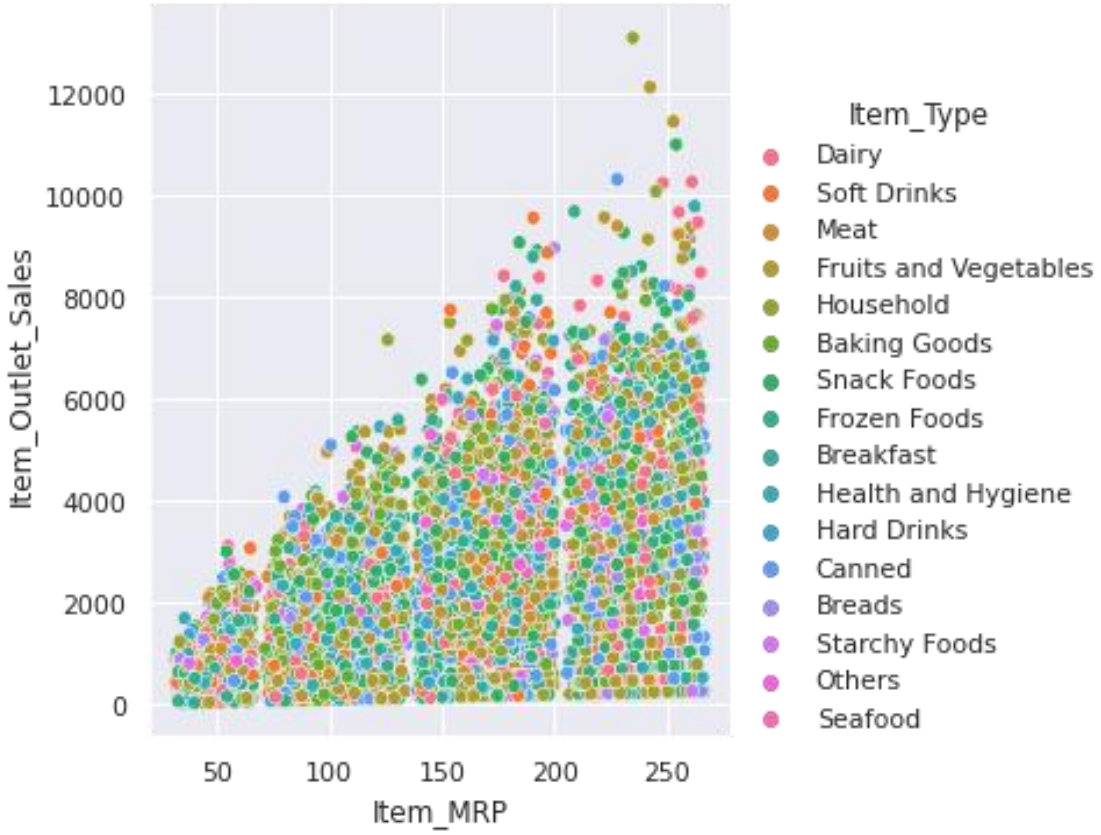

Figure 21: Scatter Plot Based On Sales MPR and Item Types

We can observe from the graph that fruits and vegetables are 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.

6. **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

```
1 import seaborn as sns
2 data_BM = pd.read_csv('bigmart_data.csv')
3 sns.catplot(x="Item_Type", y="Item_Outlet_Sales",
4             marker="*",kind='strip',size=20,data=data_BM);
```

Figure 22: Python Code for Strip Plot

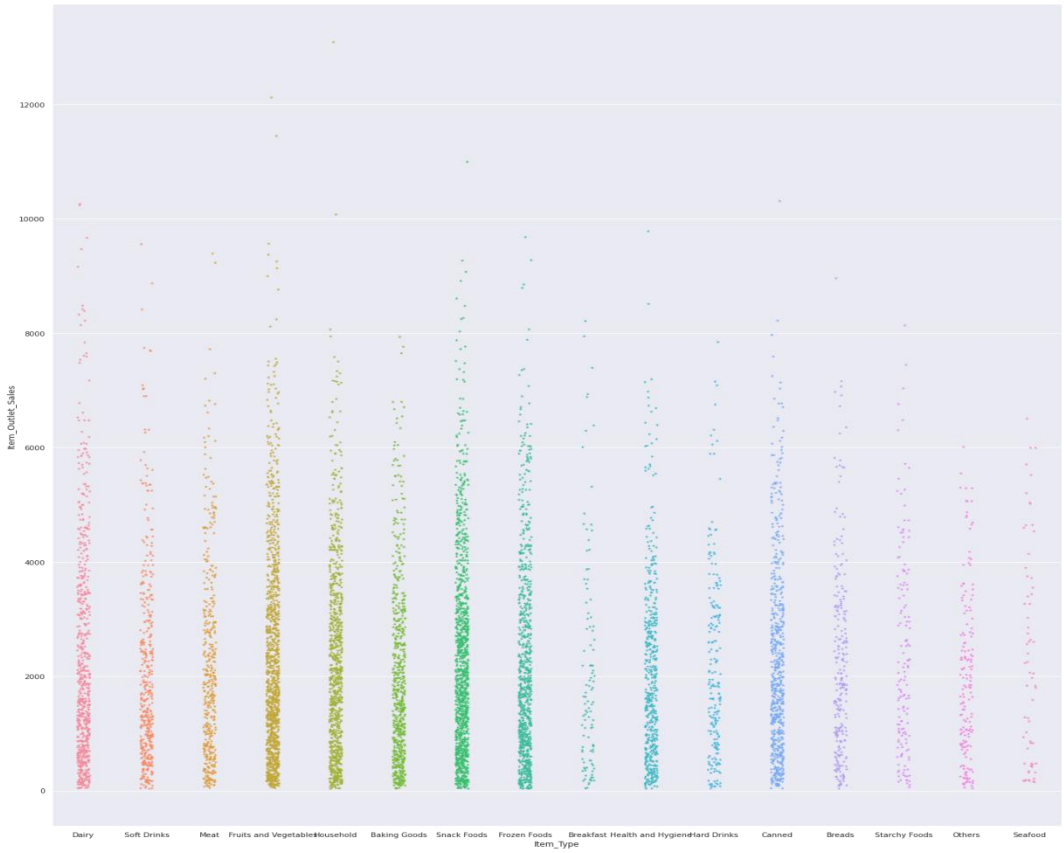

Figure 23: Strip Diagrambased on Sales and Item Type

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

7. **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]. It gives a better representation of the distribution of values. For, large numbers of observations, it does not scale well too. This style of the plot is called a "bee swarm".[13] You can create this by passing "kind=swarm" in the catplot().

Code

```
import seaborn as sns
data_BM = pd.read_csv('bigmart_data.csv')
sns.catplot(x="Item_Type", y="Item_Outlet_Sales",
            kind='swarm',size=20,data=data_BM);
```

Figure 24: Python Code for Swam Plot

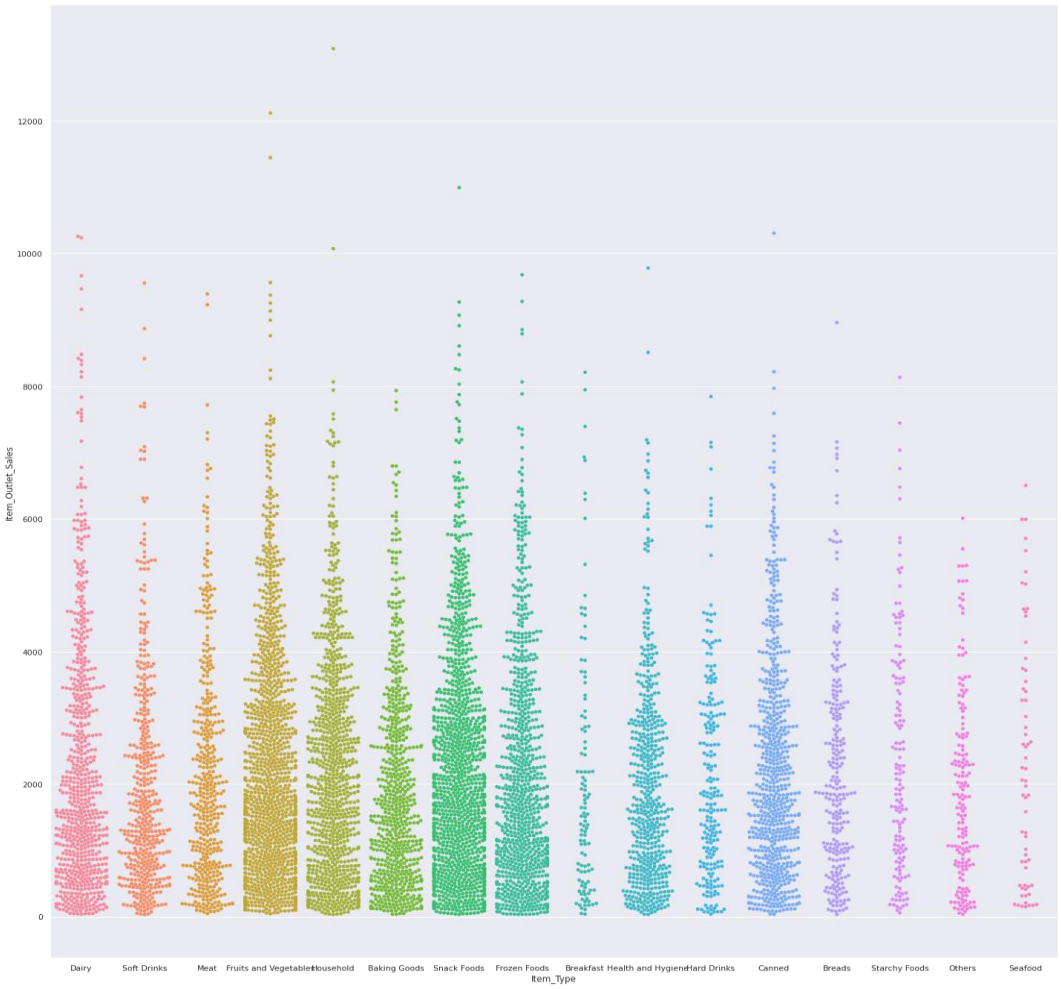

Figure 25: Swarm Plot Based on Sales and Item Type

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 & vegetables and snacks and minimum for seafood.

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

Code

```
1 import seaborn as sns
2 data_BM = pd.read_csv('bigmart_data.csv')
3 sns.catplot(x="Outlet_Size", y="Item_Outlet_Sales",
4              kind="point",data=data_BM);
```

Figure 26: Python Code for Point Plot

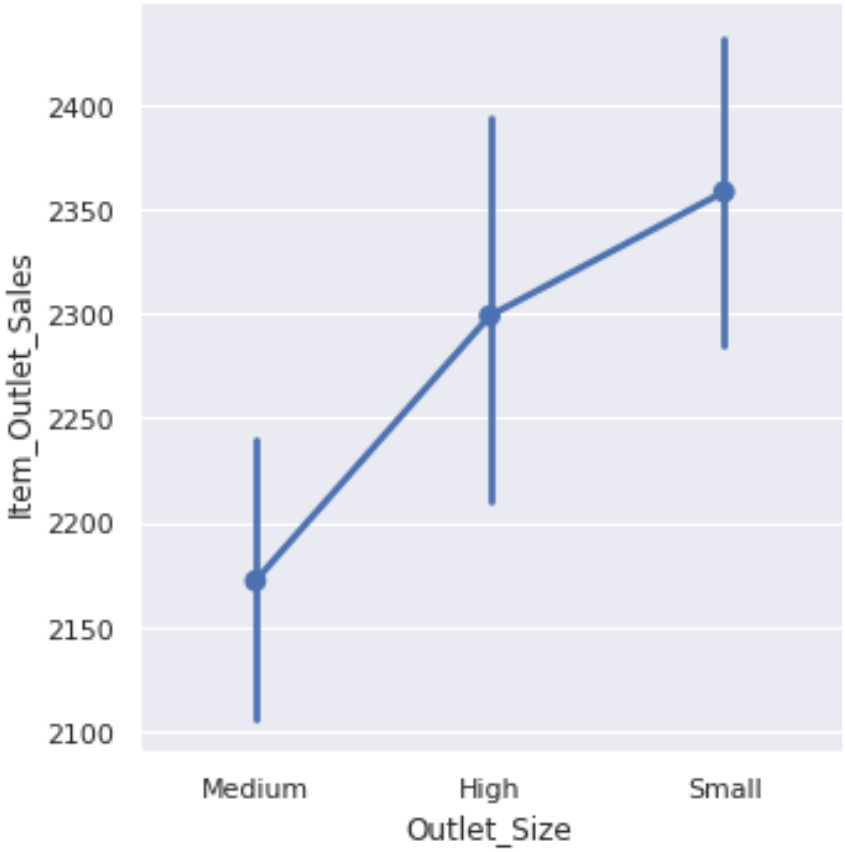

Figure 27: Point Plot based on Outlet Size and Sales

In the point plot above, the dot markers are placed against the mean number of sales 2175,2300 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

To visualize a large amount of data in readable forms and to infer the meaning, the data scientists use Seaborn and Matplotlib. It helps the user to understand the data and represent it in different attractive and efficient formats. 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. Seaborn is good for visualizing. 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."

