A
PROJECT REPORT
ON
"DATA ANALYSIS AND SALES PREDICTION OF BIG MART"

SUBMITTED BY
MEGHNA SHANTIPADA MIDYA
(228972)

UNDER THE GUIDANCE OF
PROF. DR. MANISHA ABHYANKAR

SUBMITTED IN THE PARTIAL FULFILMENT FOR
THE DEGREE OF
MASTER OF SCIENCE (COMPUTER SCIENCE)
2022–2023

DEPARTMENT OF COMPUTER SCIENCE
KARMAVEER BHAURAO PATIL COLLEGE,
VASHI (AUTONOMOUS)
PKC ROAD, SECTOR 15-A, JUHU NAGAR, VASHI,
NAVI MUMBAI, MAHARASHTRA – 400703

“Education through self-help is our Motto” – KARMAVEER

RAYAT SHIKSHAN SANSTHA’S
KARMAVEER BHAURAO PATIL COLLEGE
VASHI, NAVI MUMBAI – 400 703
[AUTONOMOUS COLLEGE]

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

THIS IS TO CERTIFY THAT THE PROJECT TITLED

“Data Analysis And Sales Prediction Of Big Mart”

IS UNDERTAKEN BY

Meghna Shantipada Midya

In partial fulfillment of the Master of Science (Computer Science) Degree (Semester IV) examination in the academic year 2022–2023 and has not been submitted for any other examination and does not form part of any other course undergone by the candidate. It is further certified that she has completed all the required phases of the Project.

Roll No.: 228972

_____
Project Guide

_____
Internal Examiner

_____
External Examiner

_____
Head of Department

_____
Principal

Date: _____

College Seal

ACKNOWLEDGEMENT

I have taken efforts in this project research. It would not have been possible without the kind support and help of many individuals and organizations. I would like to express my sincere gratitude to the people who helped me in completing the project. I heartily thank my project guide, Prof. Dr. Manisha Abhyankar Ma'am, for her valuable guidance, cooperation, encouragement and time spent on this project work. I also thank her for showing faith in me throughout the course of the project. I extend my sincere thanks to the principal of our college and all faculty members of the MSc CS department who tuned my knowledge in the field of Computer Science. I am also grateful to my family members for constantly supporting me and all those who guided me directly or indirectly during my project, encouraging me to put my best efforts into the continual improvement of the project.

DECLARATION

I hereby declare that the Project entitled, “Data Analysis And Sales Prediction Of Big Mart” done at Karmaveer Bhaurao Patil College, Vashi, Navi Mumbai – 400703, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other University. The Project is done in partial fulfillment for the award of the degree of Master of Science (Computer Science) to be submitted as a project for part of our curriculum.

ABSTRACT

In recent years, shopping malls and Big Marts manage and keep a record of their sales data for any and every unique item with the goal to estimate client demand in future periods. Every firm and organization is working on meeting customer needs while also improving their inventory management in the present era of achieving new heights of progress. In a data warehouse, these data stores hold a vast quantity of customer data and specific item information. More importantly, abnormalities and regular patterns are discovered through mining the data warehouse. The resulting data may be utilized to forecast future sales volume using various machine-learning algorithms. In this project, we provide a linear regression based predictive model to forecast sales for a firm like Big Mart. A detailed comparison of the model to others in terms of performance measures is also provided. With the use of numerous datasets gathered for Big Mart and the process used to create a predictive model, very accurate findings are produced, and these observations may be used to make decisions to increase sales.

INDEX

Sr. No.	Title	Page No.
I	Acknowledgement	i
II	Declaration	ii
III	Abstract	iii
1	Introduction	01 – 02
2	Project Description ➔ Problem Statement ➔ Proposed System ➔ Objectives	03
3	Dataset Description	04
4	Hardware and Software Requirements	05
5	Software Specification	06 – 07
6	Experimentation Environment	08 – 12
7	Methodology	13 – 16
8	Exploratory Data Analysis	17
9	Project ➔ Importing Libraries and Extracting Dataset ➔ Exploratory Data Analysis and Data Preprocessing ➔ Converting and Splitting Dataset ➔ Model Building ➔ Predicting Data	18 – 51
10	Conclusion	52
11	References	53

INTRODUCTION

Day-by-day competition among different shopping malls as well as big marts is getting more serious only due to the rapid growth of global malls and online shopping. In order to draw in more consumers depending on the moment, every mall or market tries to give unique, limited-time deals. This way, the volume of sales for each item can be forecasted for inventory management of the organization, logistics, transport service, etc. The collecting of sales data for commodities or products together with their different dependent or independent aspects is a crucial step in today's contemporary world for helping to estimate future demand and inventory management. Large shopping centers like major malls and marts are examples of this. The dataset, which was created utilizing a range of dependent and independent variables, is made up of information on the attributes of the items and information acquired from customers and inventory management information maintained in a data warehouse. The data is then changed in order to generate accurate forecasts as well as to collect new and exciting results that shed new light on our comprehension of the task's data. In order to predict forthcoming sales, this might then be utilized in conjunction with machine learning techniques. In this project, we address the problem of anticipating big-mart sales or projecting an item based on the consumers' expected future demand in various big-mart stores across various locations and goods. For the prediction or forecasting of sales volume, several machine learning methods, such as linear regression analysis, random forest, etc., are applied. In any retail center, the ability to predict sales is crucial since strong sales are the lifeblood of every business. Always a better prediction is helpful, to develop as well as to enhance the strategies of business about the marketplace which is also helpful to improve the knowledge of marketplaces. A conventional sales prediction research may assist in carefully analyzing prior scenarios or conditions and the inference can then be used to client acquisition, money insufficiency and strengths before making a budget and marketing strategies for the following year. In other words, sales forecasting is based on the resources that were accessible in the past. In-depth knowledge of the past is required for enhancing and improving the likelihood of marketplaces irrespective of any

circumstances, especially the external circumstance, which allows for the preparation of the upcoming needs of the business.

Data Analysis:

Data visualization tools and technologies are crucial in the space of big data to analyze vast volumes of information. Data visualization is the depiction of data using conventional graphics like infographics, plots and even animations. These informational visual representations make complicated data relationships and data-driven insights simple to comprehend.

Machine Learning:

The amount of data accessible is growing daily, and this massive volume of unprocessed data must be carefully analyzed in order to provide outcomes that meet the current standards for being highly useful and flawlessly pure. Data is incredibly important in current aspects, therefore as technology advances, so will the analysis and interpretation of data to provide effective outcomes. Both supervised and unsupervised types of tasks are dealt with in machine learning, and typically, a classification-type issue serves as a source for knowledge acquisition. Machine Learning appears in many guises. Various machine learning algorithm includes:

1. Linear Regression

   Linear regression is a popular supervised machine learning algorithm used for predicting numerical values. It models the relationship between an independent variable (input feature) and a dependent variable (output or target variable) as a linear equation. The goal of linear regression is to find the best-fit line or hyperplane that minimizes the error between the predicted values and the actual values of the target variable.

2. Lasso Regression

   The term Lasso regression is an acronym for Least Absolute Shrinkage and Selection Operator. Lasso regression is a type of regularization. It is preferred over regression approaches for more precise prediction. This model makes advantage of shrinkage. Shrinkage is the process of shrinking data values towards a central point known as the mean.

PROJECT DESCRIPTION

PROBLEM STATEMENT

Sales forecasting is a quite common real-world challenge that every organization experiences at some point in its entire existence. If done correctly, it may have a substantial influence on the company's success and performance. Big Mart's data scientists gathered 2013 sales data for numerous items across stores in several cities. Certain characteristics of each product and shop have also been determined. The goal is to create a predictive model that predicts the sales of each product at a certain location.

PROPOSED SYSTEM

The project is about building a model to predict accurate results for the dataset of Big Mart sales. This undergoes several sequences of steps as mentioned in the methodology and in this work, we propose a model using the Linear Regression technique. Every step plays a vital role in building the proposed model. In our model, we have used the 2013 Big Mart dataset. After preprocessing and filling in missing values, we used an ensemble classifier using Linear Regression, Ridge Regression and Lasso regression. Both the R^2 score and RSME are used as accuracy metrics for predicting sales in Big Mart.

OBJECTIVES

- Converting data into an appropriate form using various preprocessing techniques for the implementation of Machine Learning algorithms.
- The objective of this project is to predict future sales from given data of the previous years using Machine Learning Techniques.
- To determine and conclude the best model which is more efficient and gives fast and accurate results.

DATASET DESCRIPTION

The data which was required for the project is collected through a Kaggle Dataset. In the project, I have used the 2013 Sales data of Big Mart as the dataset. Big Mart's data scientists collected sales data from their 10 stores situated in different locations with each store having different products. The dataset set contains certain attributes like:

Variable	Type	Description
Item_Identifier	Numeric	Unique product ID.
Item_Weight	Numeric	Weight of the product.
Item_Fat_Content	Categorical	Indicates if the product is low fat or not.
Item_Visibility	Numeric	The percentage of a store's total display area dedicated to a specific product.
Item_Type	Categorical	It specifies the category to which the product belongs.
Item_MRP	Numeric	The product's Maximum Retail Price (list price).
Outlet_Identifier	Numeric	Unique Store ID.
Outlet_Establishment_Year	Numeric	The year in which the store was established.
Outlet_Size	Categorical	Specified the size of the store in terms of the amount of ground area it covers.
Outlet_Location_Type	Categorical	The sort of the city in which the shop is situated.
Outlet_Type	Categorical	Whether the shop is only a grocery store or a supermarket.
Item_Outlet_Sales	Numeric	Sales figures for the product at the shop. This is the outcome variable that has to be forecasted

HARDWARE & SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENT

Processor	:	Intel Core i5 CPU 1.00 GHz or more
RAM	:	8.00 GB
Hard Disk	:	256SSD or more
Monitor	:	15” CRT or LCD Monitor
Keyboard	:	Normal or Multimedia
Mouse	:	Compatible Mouse

SOFTWARE REQUIREMENT

Operating System	:	Windows 11 / Windows 10
Software	:	Anaconda Navigator
Software	:	Jupyter Notebook
Software	:	Microsoft Excel (.csv)

TECHNOLOGY USED

Programming Language	:	Python
Network	:	Internet Connectivity

SOFTWARE SPECIFICATION

JUPYTER NOTEBOOK

Jupyter Notebook is a popular open-source tool that provides an interactive computing environment for creating, running and sharing live code, equations, visualizations and narratives in a web-based notebook format. It is compatible with a number of programming languages, including Python, R, Julia and others.

Jupyter Notebook allows you to write and execute code in cells, which are organized in a linear or non-linear order. Each cell can contain code, markdown text or even multimedia elements, making it a flexible tool for data analysis, machine learning, scientific computing and other computational tasks.

Some key features of the Jupyter Notebook include:

1. Code Execution:

   One can write and run code in individual cells, which allows for interactive and iterative development. You can also store variables, functions and other objects in memory and access them across cells.

2. Markdown Support:

   You can add formatted text, equations and images using Markdown cells, which allows for documentation, explanations and visualizations alongside your code.

3. Visualization:

   Jupyter Notebook supports rich visualizations using popular data visualization libraries such as Matplotlib, Seaborn, Plotly and others. One can create static or interactive visualizations directly in the notebook.

4. Collaboration and Sharing:

   Jupyter Notebook allows you to share your notebooks with others, making it easy to collaborate on data analysis or machine learning projects. Notebooks can be exported in various formats, such as HTML, PDF or slides, for easy sharing or presentation.

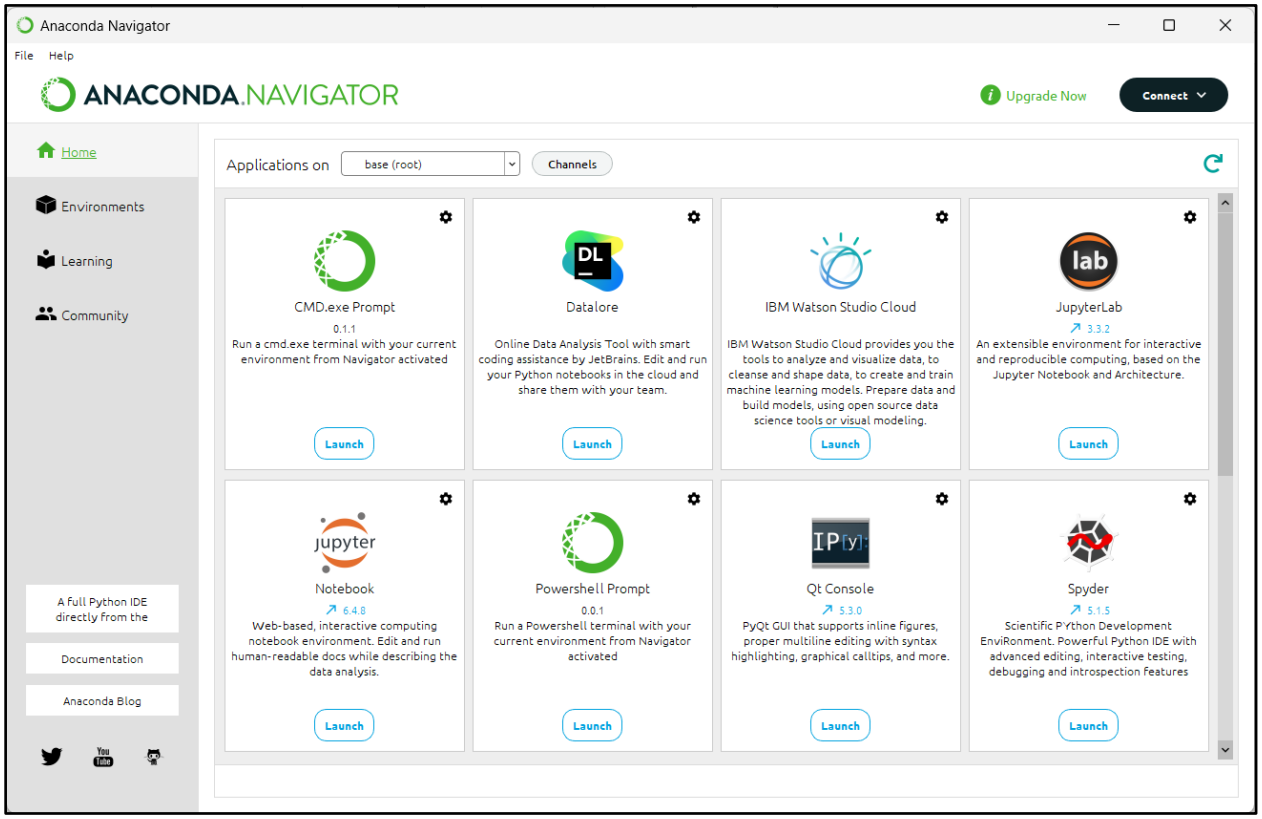
5. Extensibility:

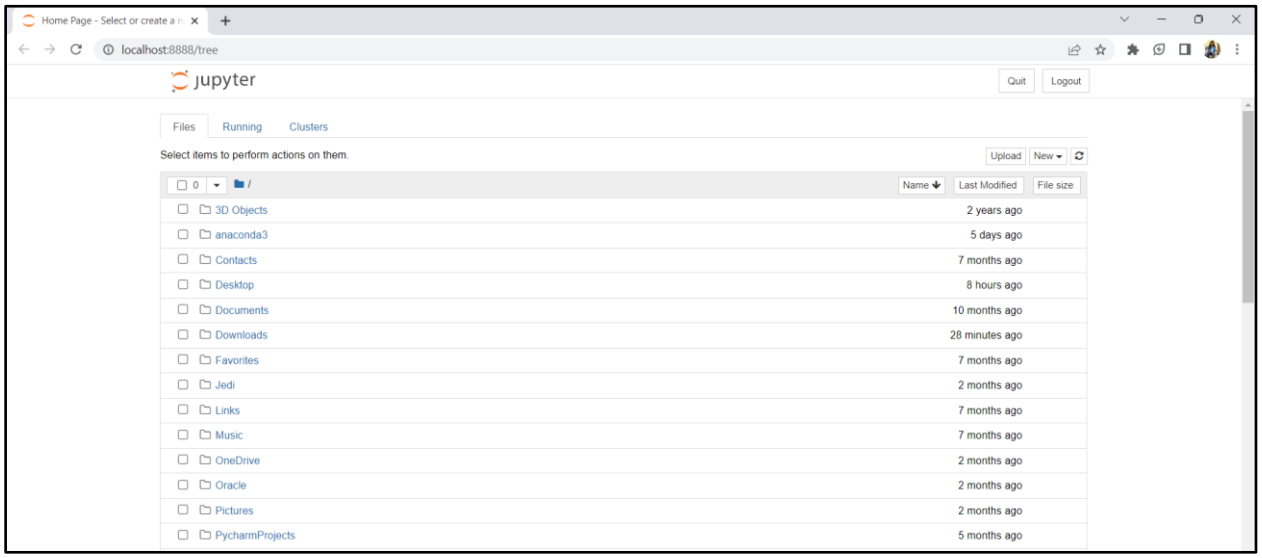
   Jupyter Notebook has a large ecosystem of extensions and plugins that enhance its functionality, including code snippets, debugging tools and more.

6. Integration with other tools:

   Jupyter Notebook can be integrated with other popular data science and machine learning tools, such as NumPy, Pandas, Scikit-Learn, TensorFlow and others, making it a powerful tool for end-to-end data analysis and model development workflows.

Jupyter Notebook can be launched using Anaconda or it can be directly accessed.





EXPERIMENTATION ENVIRONMENT

PYTHON

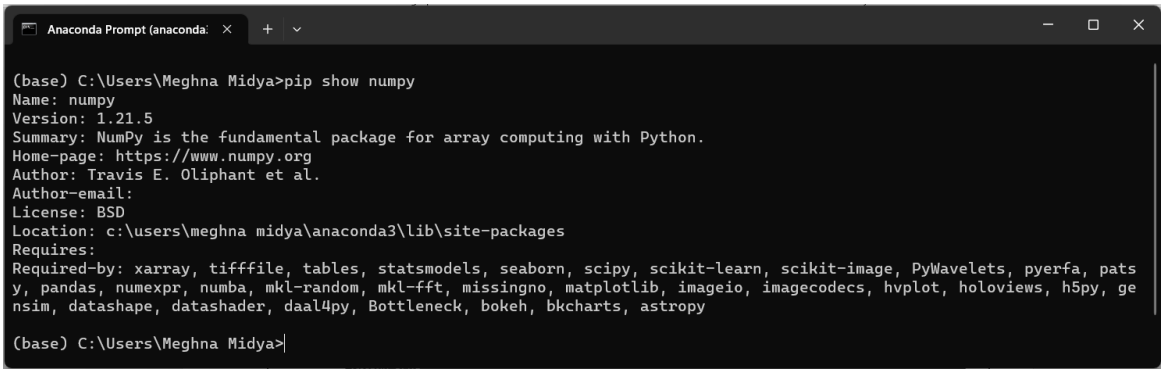
Python is a well-known high-level, interpreted programming language that has been used for a wide range of tasks, including data analysis, web development, scientific computing and artificial intelligence. It was initially made available in 1991 by Guido van Rossum, and since then it has grown to be one of the most widely used programming languages worldwide.

Python is known for its simplicity and readability, which makes it easy to learn and use, especially for beginners. Its syntax is simple and fast to comprehend. Python supports multiple programming paradigms, including procedural, object-oriented and functional programming, and it is cross-platform, meaning that it can run on a wide variety of operating systems, including Windows, Mac and Linux. In Data Science, Python is popularly used and as well as being a dynamic and open-source language it is a top choice for many users.

Programmers use Python's excellent data science tools on a regular basis to overcome difficulties. Python has a vast standard library that includes modules for tasks such as web development, file handling, networking, data processing and more. Python libraries are typically written in Python or a combination of Python and other programming languages and are distributed as open-source software, allowing developers to access, modify and contribute to the codebase. It also has a large and active community of developers who contribute to open-source libraries and frameworks, making it easy to find and use pre-existing code. The Python libraries used in this project are briefly described as follows:

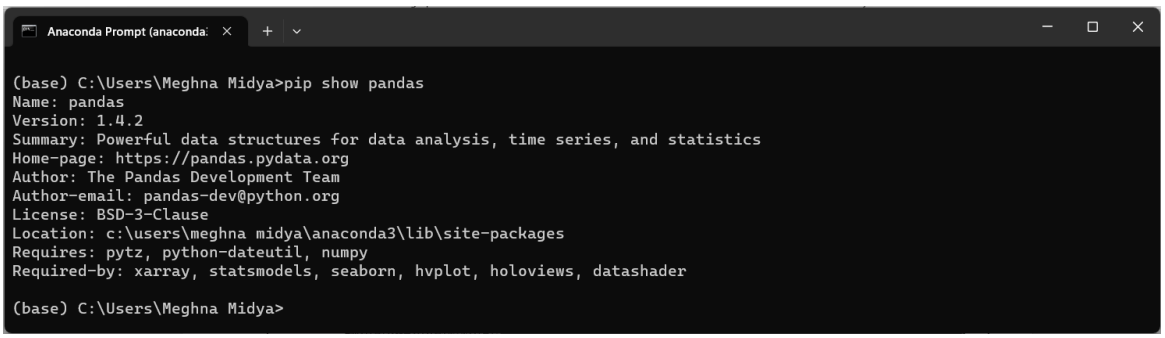
1. NumPy

NumPy (Numerical Python) is the core Python library for numerical calculation; it includes a strong N-dimensional array object. It is a general-purpose program for processing multidimensional arrays that provide the ability to deal with high-performance arrays. NumPy arrays provide vectorization of mathematical operations, which is more efficient than Python's looping structures.



2. Pandas

The data science life cycle is incomplete without Pandas (Python Data Analysis). Pandas Python is one of those data analysis packages that includes high-level data structures and simple data manipulation capabilities. Providing a simple yet efficient approach to analyzing data needs the ability to index, retrieve, divide, connect, restructure and perform numerous other analyses on both multi and single-dimensional data. Pandas offers quick, adaptable data structures, like data frame CDs, that make it simple and easy to work with structured data.



3. Matplotlib

Matplotlib package aids in visualizing and plotting data to make static, animated and interactive visualizations. Matplotlib is a 2D graphical Python library. It does, however, enable 3D graphics, but only to a limited extent. Because of the graphs and plots that it produces, it is frequently used for data visualization.


```
Anaconda Prompt (anaconda:

(base) C:\Users\Meghna Midya>pip show matplotlib
Name: matplotlib
Version: 3.5.1
Summary: Python plotting package
Home-page: https://matplotlib.org
Author: John D. Hunter, Michael Droettboom
Author-email: matplotlib-users@python.org
License: PSF
Location: c:\users\meghna midya\anaconda3\lib\site-packages
Requires: python-dateutil, pyparsing, cycler, fonttools, pillow, numpy, packaging, kiwisolver
Required-by: seaborn, missingno

(base) C:\Users\Meghna Midya>
```

4. Missingno

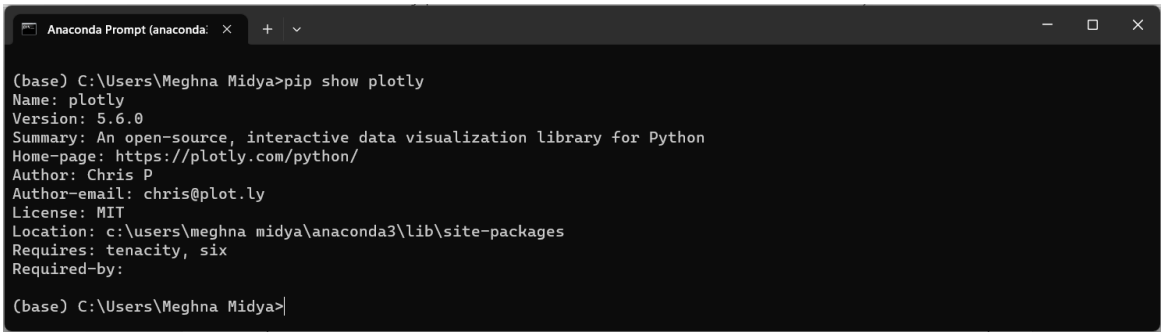
A highly attractive approach to see the distribution of NaN values is provided by the Missingno library. Missingno is a Python library that is compatible with Pandas.

```
Anaconda Prompt (anaconda:

(base) C:\Users\Meghna Midya>pip install missingno
Collecting missingno
  Using cached missingno-0.5.2-py3-none-any.whl (8.7 kB)
Requirement already satisfied: seaborn in c:\users\meghna midya\anaconda3\lib\site-packages (from missingno) (0.11.2)
Requirement already satisfied: numpy in c:\users\meghna midya\anaconda3\lib\site-packages (from missingno) (1.21.5)
Requirement already satisfied: scipy in c:\users\meghna midya\anaconda3\lib\site-packages (from missingno) (1.7.3)
Requirement already satisfied: matplotlib in c:\users\meghna midya\anaconda3\lib\site-packages (from missingno) (3.5.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\meghna midya\anaconda3\lib\site-packages (from matplotlib->missingno) (4.25.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\meghna midya\anaconda3\lib\site-packages (from matplotlib->missingno) (3.0.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\meghna midya\anaconda3\lib\site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\meghna midya\anaconda3\lib\site-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\meghna midya\anaconda3\lib\site-packages (from matplotlib->missingno) (1.3.2)
Requirement already satisfied: packaging>=20.0 in c:\users\meghna midya\anaconda3\lib\site-packages (from matplotlib->missingno) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\meghna midya\anaconda3\lib\site-packages (from matplotlib->missingno) (9.0.1)
Requirement already satisfied: six>=1.5 in c:\users\meghna midya\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
Requirement already satisfied: pandas>=0.23 in c:\users\meghna midya\anaconda3\lib\site-packages (from seaborn->missingno) (1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\meghna midya\anaconda3\lib\site-packages (from pandas>=0.23->seaborn->missingno) (2021.3)
Installing collected packages: missingno
Successfully installed missingno-0.5.2

(base) C:\Users\Meghna Midya>
```

```
Anaconda Prompt (anaconda:

(base) C:\Users\Meghna Midya>pip show missingno
Name: missingno
Version: 0.5.2
Summary: Missing data visualization module for Python.
Home-page: https://github.com/ResidentMario/missingno
Author: Aleksey Bilogur
Author-email: aleksey.bilogur@gmail.com
License: MIT License
Location: c:\users\meghna midya\anaconda3\lib\site-packages
Requires: scipy, seaborn, matplotlib, numpy
Required-by:

(base) C:\Users\Meghna Midya>
```

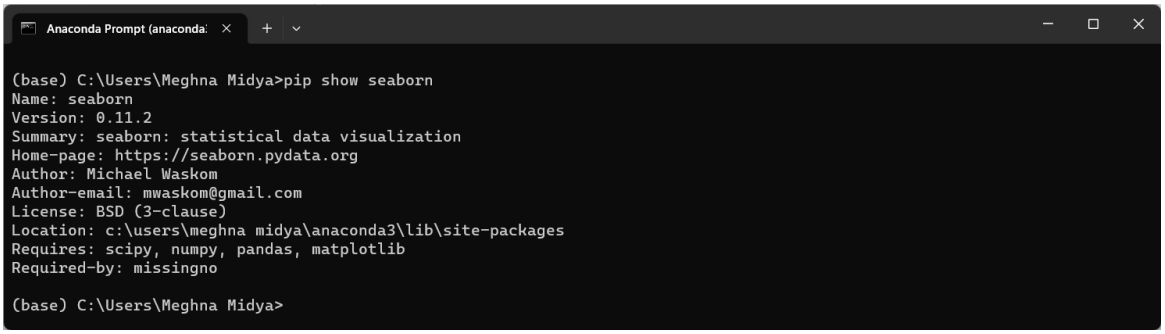
5. Plotly

Plotly is a free and open-source Python framework for 3D data visualization. Plotly supports a variety of chart forms, including scatter plots, histograms, line charts, bar charts, box plots, multiple axes, sparklines, dendrograms, three-dimensional graphs and more. Plotly also includes contour plots, distinguishing them from other data visualization frameworks.



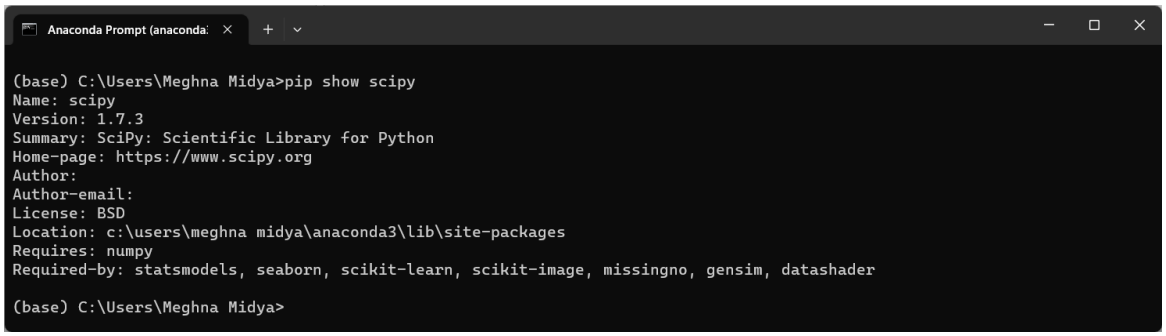
6. Seaborn

Seaborn is a well-known Python data visualization package that is based on Matplotlib and provides a high-level interface for making visually appealing and helpful statistical visualizations. Seaborn is particularly well-suited for visualizing complex datasets and is often used in data science, machine learning and statistical analysis applications.



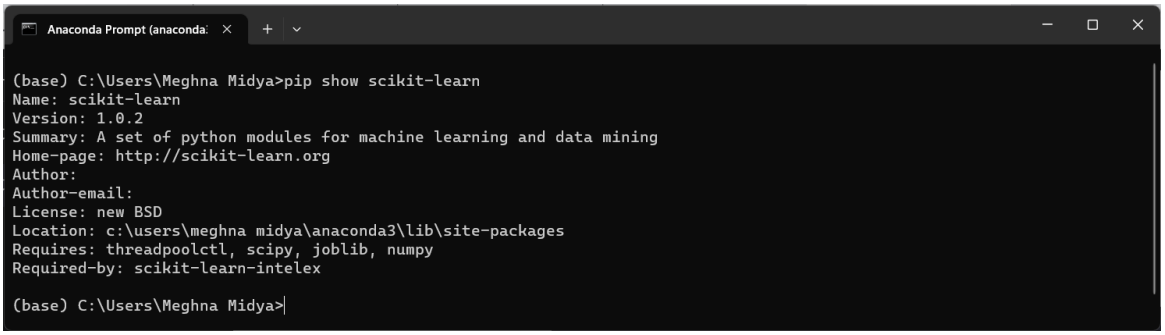
7. SciPy

Another free and open-source Python data science library that is extensively used for high-level mathematical calculation is SciPy (Scientific Python). Since it extends NumPy and includes a number of user-friendly and efficient scientific computation techniques, it is widely used for scientific and technical calculations. SciPy includes a number of high-level data manipulation and visualization commands and classes. SciPy is skilled in data processing and system prototyping.

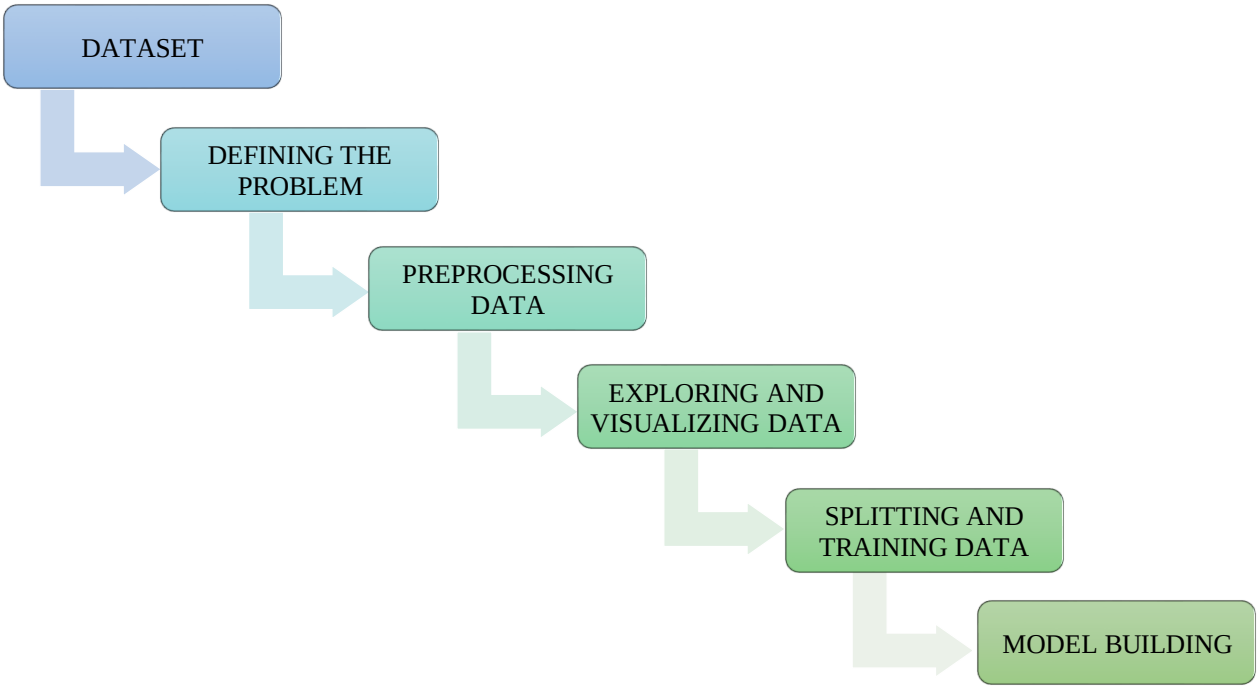


8. Sci-Kit Learn (sklearn)

Scikit-learn, commonly abbreviated as sklearn, is a popular Python library for machine learning that provides a wide range of tools for tasks such as classification, regression, clustering, dimensionality reduction, model evaluation and more. Scikit-learn is built on top of NumPy, SciPy and Matplotlib, and is widely used in machine learning and data science applications for developing predictive models and performing data analysis tasks.

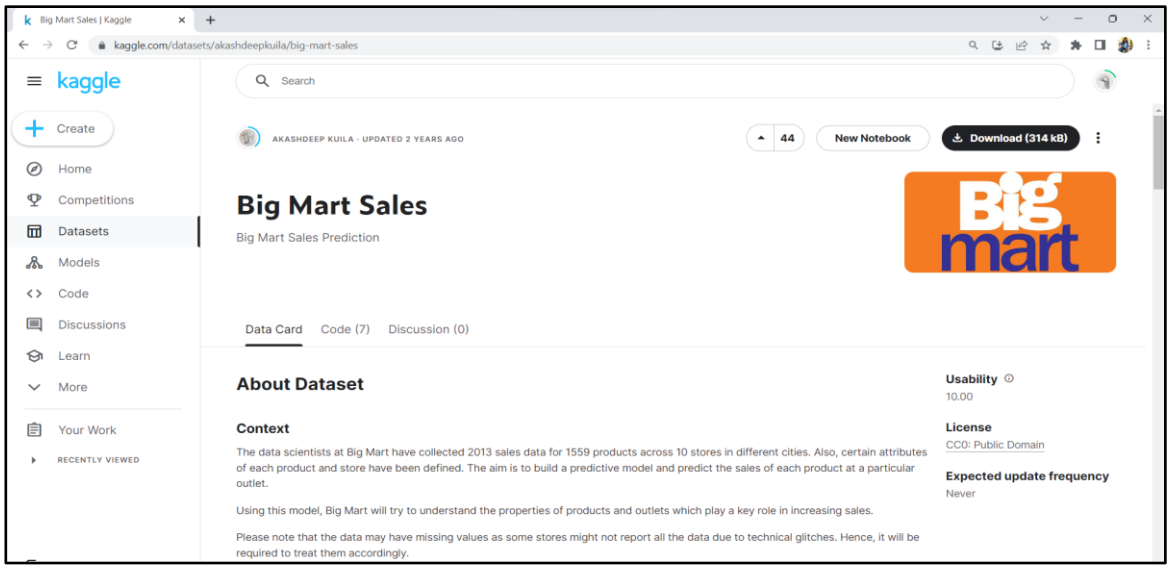


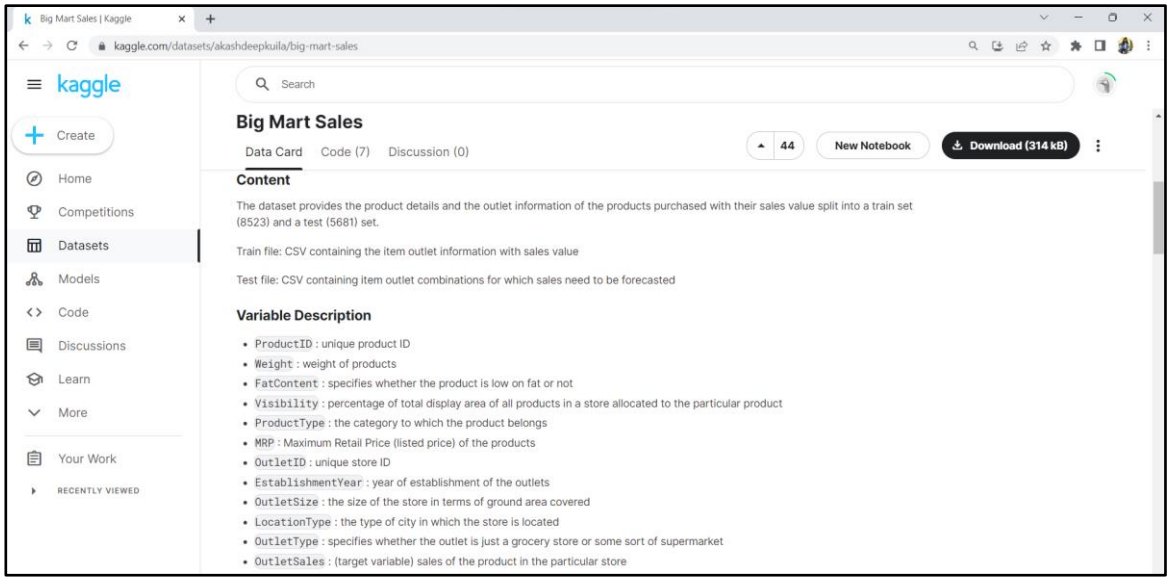
METHODOLOGY

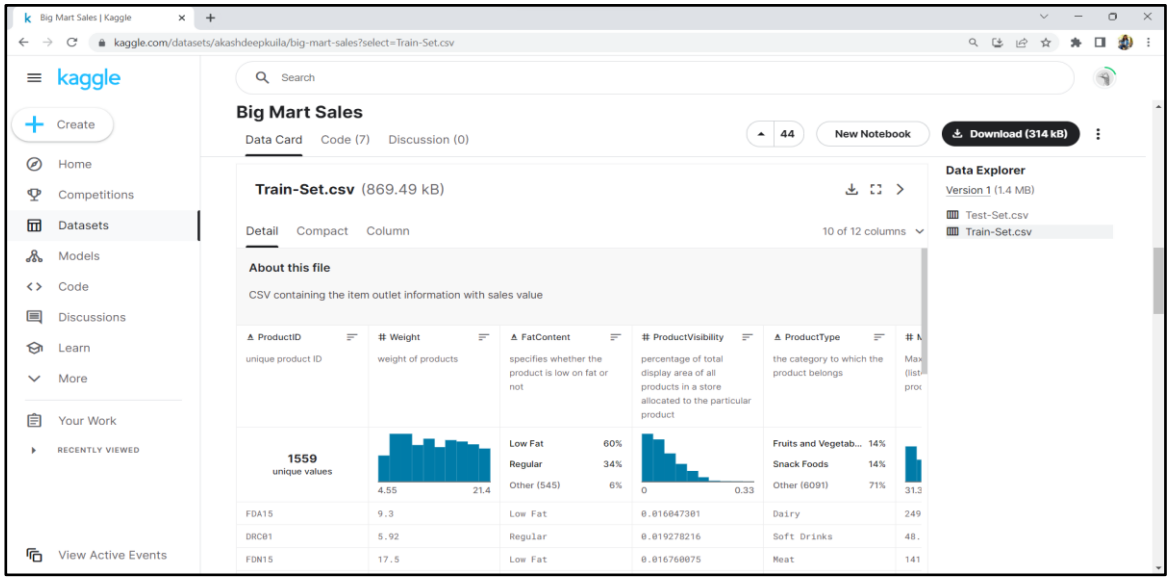


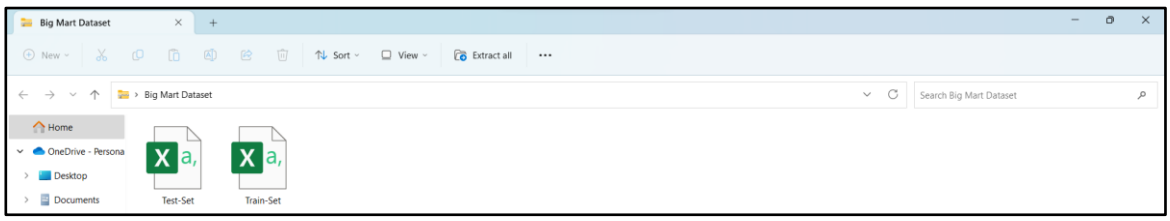
1. Dataset

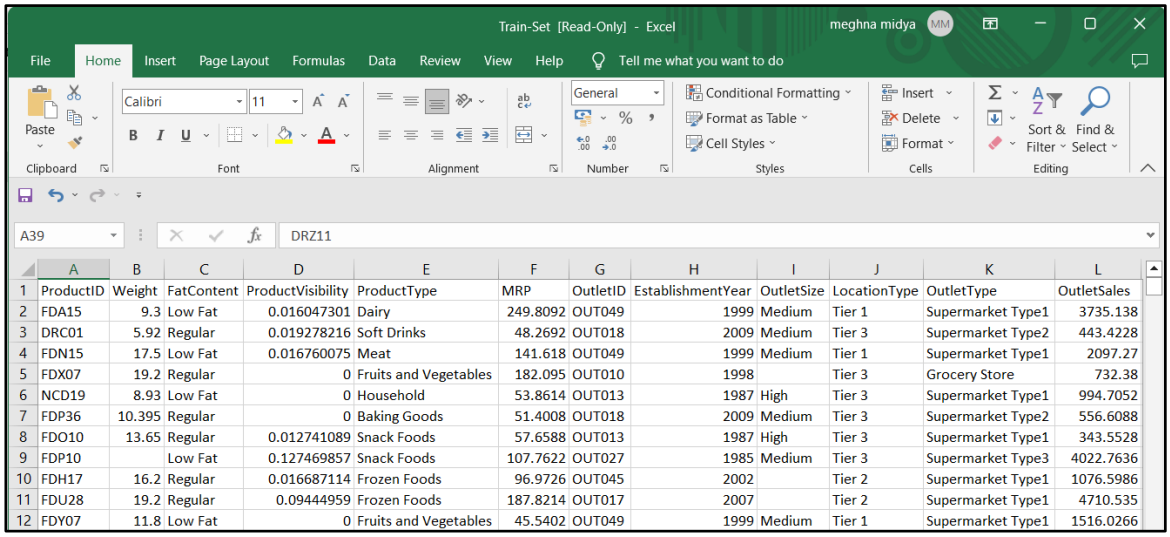
Through the use of a Kaggle Dataset, the project's necessary data was gathered. Data collection is the process of gathering, acquiring and compiling data from various sources or methods. Datasets are used in a wide range of fields, including but not limited to, data science, machine learning, statistics, social sciences, healthcare, finance, marketing and more. They serve as the foundation for conducting research, analysis and modeling and are crucial for generating insights, making informed decisions and developing predictive or prescriptive models.











	A	B	C	D	E	F	G	H	I	J	K	L
1	ProductID	Weight	FatContent	ProductVisibility	ProductType	MRP	OutletID	EstablishmentYear	OutletSize	LocationType	OutletType	OutletSales
2	FDA15	9.3	Low Fat	0.016047301	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.138
3	DRC01	5.92	Regular	0.019278216	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
4	FDN15	17.5	Low Fat	0.016760075	Meat	141.618	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.27
5	FDX07	19.2	Regular	0	Fruits and Vegetables	182.095	OUT010	1998		Tier 3	Grocery Store	732.38
6	NCD19	8.93	Low Fat	0	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052
7	FDP36	10.395	Regular	0	Baking Goods	51.4008	OUT018	2009	Medium	Tier 3	Supermarket Type2	556.6088
8	FDO10	13.65	Regular	0.012741089	Snack Foods	57.6588	OUT013	1987	High	Tier 3	Supermarket Type1	343.5528
9	FDP10		Low Fat	0.127469857	Snack Foods	107.7622	OUT027	1985	Medium	Tier 3	Supermarket Type3	4022.7636
10	FDH17	16.2	Regular	0.016687114	Frozen Foods	96.9726	OUT045	2002		Tier 2	Supermarket Type1	1076.5986
11	FDU28	19.2	Regular	0.09444959	Frozen Foods	187.8214	OUT017	2007		Tier 2	Supermarket Type1	4710.535
12	FDY07	11.8	Low Fat	0	Fruits and Vegetables	45.5402	OUT049	1999	Medium	Tier 1	Supermarket Type1	1516.0266

2. Defining the Problem

It is necessary to accurately define the data problem that is to be solved. The issue or goal of the data analysis and data prediction should be clearly stated. To establish your complete prediction method, it is essential that you comprehend the issue or goal.

Each business will encounter sales projection issues at some point throughout its existence. If done correctly, it may significantly affect the performance and profitability of that firm. Big Mart's data scientists have gathered sales information from 2013 for numerous items across stores in several cities. The goal is to create a prediction model that can forecast the sales of each item at certain stores.

3. Preprocessing Data

Preprocessing includes cleaning, converting, and preparing raw data so that it is appropriate for additional analysis or modeling. It is a critical stage in the data analysis and machine learning pipeline. Preprocessing is often necessary to address issues such as missing data, inconsistent formatting, noise, outliers or irrelevant information in the data, and to standardize or normalize the data to a common format or scale. Properly preprocessing data can significantly impact the accuracy, reliability and interpretability of the results obtained from the data analysis or modeling process.

4. Exploring and Visualizing Data

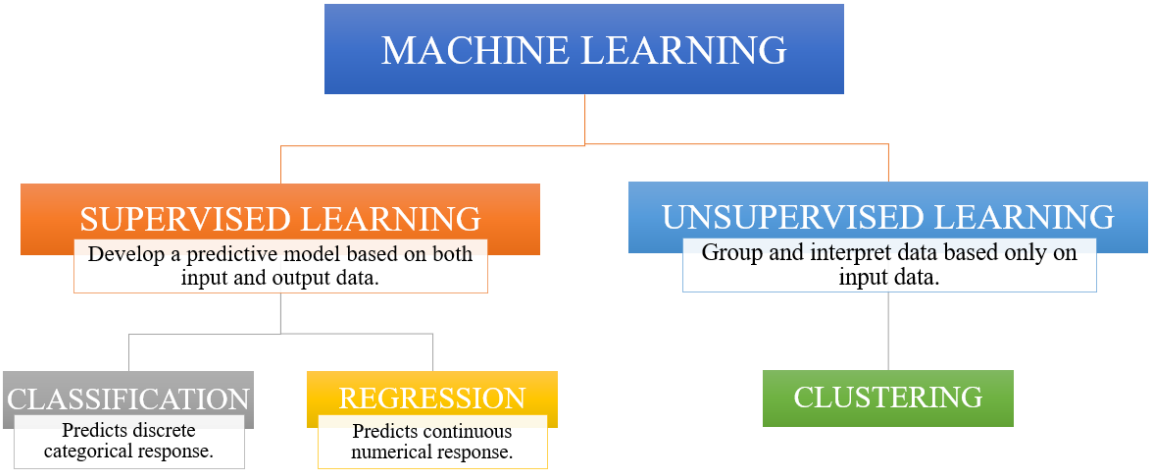
Exploring and visualizing data is an important step in the data analysis process that involves examining, summarizing and visualizing the main characteristics, patterns and relationships present in the data. Data exploration and visualization techniques are used to gain insights, identify trends, detect patterns and uncover hidden information in the data, which can help inform decision-making, generate hypotheses, or guide further analysis.

5. Splitting and Training Data

Splitting the data into training and testing datasets is necessary for any data analysis. The training dataset is used to train the prediction model, while the testing dataset is used to evaluate the performance of the trained model. Preprocessing the training and testing datasets separately or together ensures that any data transformations or imputations are applied consistently.

6. Model Building

Machine Learning prediction models are trained in this process and then later on evaluated using the data. Choose an appropriate prediction model based on the characteristics of your data and the problem you are trying to solve. Store the generated output in the required format.



EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis is a critical step in the data analysis process that involves examining and analyzing data to understand its structure, patterns, relationships and characteristics. EDA helps in gaining insights, identifying trends, detecting anomalies and making data-driven decisions. Some key objectives of EDA include:

1. Data Understanding:

   EDA involves thoroughly examining the data to understand its content, structure and quality. This includes checking data types, variable distributions, missing values, outliers and other data characteristics.

2. Data Visualization:

   EDA often involves creating visualizations such as plots, charts and graphs to explore data visually. Visualizations can help in identifying patterns, trends and relationships in the data, making it easier to interpret and understand.

3. Data Summarization and Aggregation:

   EDA involves aggregating and summarizing data to obtain descriptive statistics such as mean, median, mode, standard deviation and others. This can help in gaining insights into the central tendency, variability and distribution of the data.

4. Data Profiling:

   EDA may involve creating data profiles, which provide an overview of the data characteristics, such as unique values, data ranges, data distributions and data quality metrics. Data profiling helps in identifying data issues and understanding the data's overall quality.

5. Data Cleansing and Preprocessing:

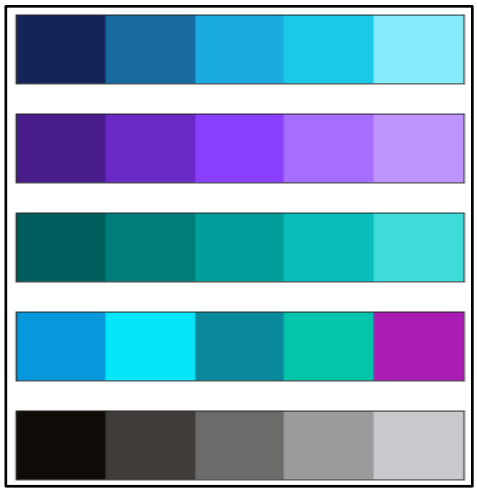
   EDA may also involve identifying and addressing data quality issues, such as missing values, outliers, inconsistencies and errors. This may include data imputation, data cleaning and data transformation techniques to prepare the data for further analysis.

PROJECT

Importing Libraries And Extracting Dataset

In [1]:
```
# --- Importing Libraries ---
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import missingno as mso
import plotly.express as px
import seaborn as sns
import matplotlib.gridspec as gridspec
import warnings
from scipy.stats import mode

# --- Libraries Settings ---
warnings.filterwarnings('ignore')
sns.set_style('white')
```

In [2]:
```
# --- Creating List of Color Palletes ---
cyan_grad = ['#142459', '#176BA0', '#19AADE', '#1AC9E6', '#87EAFA']
purple_grad = ['#491D8B', '#6929C4', '#8A3FFC', '#A56EFF', '#BE95FF']
teal_grad = ['#005D5D', '#007D79', '#009D9A', '#08BDBA', '#3DDBD9']
color_mix = ['#0698DC', '#05E6FA', '#09899B', '#04C4AC', '#AB1CB4']
black_grad = ['#100C07', '#3E3B39', '#6D6A6A', '#9B9A9C', '#CAC9CD']

# --- Ploting Color Palletes ---
sns.palplot(cyan_grad)
sns.palplot(purple_grad)
sns.palplot(teal_grad)
sns.palplot(color_mix)
sns.palplot(black_grad)
```



In [3]:
```
# Reading the Train and Test CSV files
train = pd.read_csv("C:/Users/Meghna Midya/Downloads/Dataset/Train.csv")
test = pd.read_csv("C:/Users/Meghna Midya/Downloads/Dataset/Test.csv")
```

In [4]:
```
# --- Reading Train Dataset ---
train.head(10).style.background_gradient(cmap='YlOrRd').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	FDA15	9.300000	Low Fat	0.016047	Dairy	249.809200	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.138000
1	DRC01	5.920000	Regular	0.019278	Soft Drinks	48.269200	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.422800
2	FDN15	17.500000	Low Fat	0.016760	Meat	141.618000	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.270000
3	FDX07	19.200000	Regular	0.000000	Fruits and Vegetables	182.095000	OUT010	1998	nan	Tier 3	Grocery Store	732.380000
4	NCD19	8.930000	Low Fat	0.000000	Household	53.861400	OUT013	1987	High	Tier 3	Supermarket Type1	994.705200
5	FDP36	10.395000	Regular	0.000000	Baking Goods	51.400800	OUT018	2009	Medium	Tier 3	Supermarket Type2	556.608800
6	FDO10	13.650000	Regular	0.012741	Snack Foods	57.658800	OUT013	1987	High	Tier 3	Supermarket Type1	343.552800
7	FDP10	nan	Low Fat	0.127470	Snack Foods	107.762200	OUT027	1985	Medium	Tier 3	Supermarket Type3	4022.763600
8	FDH17	16.200000	Regular	0.016687	Frozen Foods	96.972600	OUT045	2002	nan	Tier 2	Supermarket Type1	1076.598600
9	FDU28	19.200000	Regular	0.094450	Frozen Foods	187.821400	OUT017	2007	nan	Tier 2	Supermarket Type1	4710.535000

```
In [5]: # --- Describing Train Dataset ---
        print('.: Train Dataset Description :.')
        print('*' * 31)
        train.describe(include='all').style.set_properties(**{'border': '1px solid black'})
```

```
.: Train Dataset Description :.
*******************************
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
count	8523	7060.000000	8523	8523.000000	8523	8523.000000	8523	8523.000000	6113	8523	8523	8523.000000
unique	1559	nan	5	nan	16	nan	10	nan	3	3	4	nan
top	FDW13	nan	Low Fat	nan	Fruits and Vegetables	nan	OUT027	nan	Medium	Tier 3	Supermarket Type1	nan
freq	10	nan	5089	nan	1232	nan	935	nan	2793	3350	5577	nan
mean	nan	12.857645	nan	0.066132	nan	140.992782	nan	1997.831867	nan	nan	nan	2181.288914
std	nan	4.643456	nan	0.051598	nan	62.275067	nan	8.371760	nan	nan	nan	1706.499616
min	nan	4.555000	nan	0.000000	nan	31.290000	nan	1985.000000	nan	nan	nan	33.290000
25%	nan	8.773750	nan	0.026989	nan	93.826500	nan	1987.000000	nan	nan	nan	834.247400
50%	nan	12.600000	nan	0.053931	nan	143.012800	nan	1999.000000	nan	nan	nan	1794.331000
75%	nan	16.850000	nan	0.094585	nan	185.643700	nan	2004.000000	nan	nan	nan	3101.296400
max	nan	21.350000	nan	0.328391	nan	266.888400	nan	2009.000000	nan	nan	nan	13086.964800

From the train dataset description, we get a clear idea about the mean value, standard deviation, minimum and maximum value of the numerical values and for categorical values we idea about their uniqueness, mostly occurred values and their frequency.

```
In [6]: # --- Printing Train Dataset Info ---
        print('.: Train Dataset Info :.')
        print('*' * 23)
        print('Total Rows:', train.shape[0])
        print('Total Columns:', train.shape[1])
```

```
.: Train Dataset Info :.
***********************
Total Rows: 8523
Total Columns: 12
```

```
In [7]: # --- Printing Train Dataset Detail ---
        print('.: Train Dataset Details :.')
        print('*' * 27)
        train.info()
```

```
.: Train Dataset Details :.
***************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

It is obvious from running train.info() that the Item_Weight and Outlet_Size variables in the train dataset have NaN values (Null values). Additionally, we can observe that the dataset has 7 categorical features (Item_Identifier, Item_Fat_Content, Item_Type, Outlet_Identifier, Outlet_Size, Outlet_Location_Type and Outlet_Type) in addition to 5 numerical features (Item_Weight, Item_Visibility, Item_MRP, Outlet_Establishment_Year and Item_Outlet_Sales).

```
In [8]: # --- Reading Test Dataset ---
        test.head(10).style.background_gradient(cmap='summer').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
0	FDW58	20.750000	Low Fat	0.007565	Snack Foods	107.862200	OUT049	1999	Medium	Tier 1	Supermarket Type1
1	FDW14	8.300000	reg	0.038428	Dairy	87.319800	OUT017	2007	nan	Tier 2	Supermarket Type1
2	NCN55	14.600000	Low Fat	0.099575	Others	241.753800	OUT010	1998	nan	Tier 3	Grocery Store
3	FDQ58	7.315000	Low Fat	0.015388	Snack Foods	155.034000	OUT017	2007	nan	Tier 2	Supermarket Type1
4	FDY38	nan	Regular	0.118599	Dairy	234.230000	OUT027	1985	Medium	Tier 3	Supermarket Type3
5	FDH56	9.800000	Regular	0.063817	Fruits and Vegetables	117.149200	OUT046	1997	Small	Tier 1	Supermarket Type1
6	FDL48	19.350000	Regular	0.082602	Baking Goods	50.103400	OUT018	2009	Medium	Tier 3	Supermarket Type2
7	FDC48	nan	Low Fat	0.015782	Baking Goods	81.059200	OUT027	1985	Medium	Tier 3	Supermarket Type3
8	FDN33	6.305000	Regular	0.123365	Snack Foods	95.743600	OUT045	2002	nan	Tier 2	Supermarket Type1
9	FDA36	5.985000	Low Fat	0.005698	Baking Goods	186.892400	OUT017	2007	nan	Tier 2	Supermarket Type1

```
In [9]: # --- Describing Test Dataset ---
        print('.: Test Dataset Description :.')
        print('*' * 30)
        test.describe(include='all').style.set_properties(**{'border': '1px solid black'})
```

```
.: Test Dataset Description :.
******************************
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
count	5681	4705.000000	5681	5681.000000	5681	5681.000000	5681	5681.000000	4075	5681	5681
unique	1543	nan	5	nan	16	nan	10	nan	3	3	4
top	DRF48	nan	Low Fat	nan	Snack Foods	nan	OUT027	nan	Medium	Tier 3	Supermarket Type1
freq	8	nan	3396	nan	789	nan	624	nan	1862	2233	3717
mean	nan	12.695633	nan	0.065684	nan	141.023273	nan	1997.828903	nan	nan	nan
std	nan	4.664849	nan	0.051252	nan	61.809091	nan	8.372256	nan	nan	nan
min	nan	4.555000	nan	0.000000	nan	31.990000	nan	1985.000000	nan	nan	nan
25%	nan	8.645000	nan	0.027047	nan	94.412000	nan	1987.000000	nan	nan	nan
50%	nan	12.500000	nan	0.054154	nan	141.415400	nan	1999.000000	nan	nan	nan
75%	nan	16.700000	nan	0.093463	nan	186.026600	nan	2004.000000	nan	nan	nan
max	nan	21.350000	nan	0.323637	nan	266.588400	nan	2009.000000	nan	nan	nan

From the test dataset description, we get a clear idea about the mean value, standard deviation, minimum and maximum value of the numerical values and for categorical values we idea about their uniqueness, mostly occurred values and their frequency.

```
In [10]: # --- Printing Test Dataset Info ---
         print('.: Test Dataset Info :.')
         print('*' * 23)
         print('Total Rows:', test.shape[0])
         print('Total Columns:', test.shape[1])

         .: Test Dataset Info :.
         ***********************
         Total Rows: 5681
         Total Columns: 11
```

```
In [11]: # --- Printing Test Dataset Detail ---
         print('.: Test Dataset Details :.')
         print('*' * 27)
         test.info()

         .: Test Dataset Details :.
         ***************************
         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 5681 entries, 0 to 5680
         Data columns (total 11 columns):
          #   Column                     Non-Null Count  Dtype
         ---  ------                     --------------  -----
          0   Item_Identifier            5681 non-null   object
          1   Item_Weight                4705 non-null   float64
          2   Item_Fat_Content           5681 non-null   object
          3   Item_Visibility            5681 non-null   float64
          4   Item_Type                  5681 non-null   object
          5   Item_MRP                   5681 non-null   float64
          6   Outlet_Identifier          5681 non-null   object
          7   Outlet_Establishment_Year  5681 non-null   int64
          8   Outlet_Size                4075 non-null   object
          9   Outlet_Location_Type       5681 non-null   object
          10  Outlet_Type                5681 non-null   object
         dtypes: float64(3), int64(1), object(7)
         memory usage: 488.3+ KB
```

It is obvious from running test.info() that the Item_Weight and Outlet_Size variables in the test dataset have NaN values. Also, we can see that the dataset has 4 Numerical features(Item_Weight, Item_Visibility, Item_MRP and Outlet_Establishment_Year) and 7 Categorical features(Item_Identifier, Item_Fat_Content, Item_Type, Outlet_Identifier, Outlet_Size, Outlet_Location_Type and Outlet_Type).

Exploratory Data Analysis And Data Preprocessing

```
In [12]: # --- Join both the train and test dataset ---
         train['source']='Train'
         test['source']='Test'

         dataset = pd.concat([train,test], ignore_index = True)

         print('.: Train Dataset Shape :.')
         print('*' * 25)
         print('Total Rows:', train.shape[0])
         print('Total Columns:', train.shape[1])

         print('\n.: Test Dataset Shape :.')
         print('*' * 24)
         print('Total Rows:', test.shape[0])
         print('Total Columns:', test.shape[1])


         print('\n.: Concatenated Dataset Shape :.')
         print('*' * 32)
         print('Total Rows:', dataset.shape[0])
         print('Total Columns:', dataset.shape[1])

         .: Train Dataset Shape :.
         *************************
         Total Rows: 8523
         Total Columns: 13

         .: Test Dataset Shape :.
         ************************
         Total Rows: 5681
         Total Columns: 12

         .: Concatenated Dataset Shape :.
         ********************************
         Total Rows: 14204
         Total Columns: 13
```

Combining the training and testing datasets together for pre-processing both datasets in a single go. This will help to easily pre-process the data without repeating the same steps for both datasets.

In [13]:
```
# --- Reading Combined Dataset (From Top) ---
dataset.head(10).style.background_gradient(cmap='BuPu').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source
0	FDA15	9.300000	Low Fat	0.016047	Dairy	249.809200	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.138000	Train
1	DRC01	5.920000	Regular	0.019278	Soft Drinks	48.269200	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.422800	Train
2	FDN15	17.500000	Low Fat	0.016760	Meat	141.618000	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.270000	Train
3	FDX07	19.200000	Regular	0.000000	Fruits and Vegetables	182.095000	OUT010	1998	nan	Tier 3	Grocery Store	732.380000	Train
4	NCD19	8.930000	Low Fat	0.000000	Household	53.861400	OUT013	1987	High	Tier 3	Supermarket Type1	994.705200	Train
5	FDP36	10.395000	Regular	0.000000	Baking Goods	51.400800	OUT018	2009	Medium	Tier 3	Supermarket Type2	556.608800	Train
6	FDO10	13.650000	Regular	0.012741	Snack Foods	57.658800	OUT013	1987	High	Tier 3	Supermarket Type1	343.552800	Train
7	FDP10	nan	Low Fat	0.127470	Snack Foods	107.762200	OUT027	1985	Medium	Tier 3	Supermarket Type3	4022.763600	Train
8	FDH17	16.200000	Regular	0.016687	Frozen Foods	96.972600	OUT045	2002	nan	Tier 2	Supermarket Type1	1076.598600	Train
9	FDU28	19.200000	Regular	0.094450	Frozen Foods	187.821400	OUT017	2007	nan	Tier 2	Supermarket Type1	4710.535000	Train

In [14]:
```
# --- Reading Combined Dataset (From Bottom) ---
dataset.tail(10).style.background_gradient(cmap='BuPu').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source
14194	FDA01	15.000000	reg	0.054463	Canned	59.590400	OUT049	1999	Medium	Tier 1	Supermarket Type1	nan	Test
14195	NCH42	6.860000	Low Fat	0.036594	Household	231.101000	OUT049	1999	Medium	Tier 1	Supermarket Type1	nan	Test
14196	FDF46	7.070000	Low Fat	0.094053	Snack Foods	116.083400	OUT018	2009	Medium	Tier 3	Supermarket Type2	nan	Test
14197	DRL35	15.700000	Low Fat	0.030704	Hard Drinks	43.277000	OUT046	1997	Small	Tier 1	Supermarket Type1	nan	Test
14198	FDW46	13.000000	Regular	0.070411	Snack Foods	63.448400	OUT049	1999	Medium	Tier 1	Supermarket Type1	nan	Test
14199	FDB58	10.500000	Regular	0.013496	Snack Foods	141.315400	OUT046	1997	Small	Tier 1	Supermarket Type1	nan	Test
14200	FDD47	7.600000	Regular	0.142991	Starchy Foods	169.144800	OUT018	2009	Medium	Tier 3	Supermarket Type2	nan	Test
14201	NCO17	10.000000	Low Fat	0.073529	Health and Hygiene	118.744000	OUT045	2002	nan	Tier 2	Supermarket Type1	nan	Test
14202	FDJ26	15.300000	Regular	0.000000	Canned	214.621800	OUT017	2007	nan	Tier 2	Supermarket Type1	nan	Test
14203	FDU37	9.500000	Regular	0.104720	Canned	79.796000	OUT045	2002	nan	Tier 2	Supermarket Type1	nan	Test

In [15]:
```
# --- Printing Combined Dataset Detail ---
print('.: Combined Dataset Details :.')
print('*' * 30)
dataset.info()
```

```
.: Combined Dataset Details :.
******************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14204 entries, 0 to 14203
Data columns (total 13 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            14204 non-null  object
 1   Item_Weight                11765 non-null  float64
 2   Item_Fat_Content           14204 non-null  object
 3   Item_Visibility            14204 non-null  float64
 4   Item_Type                  14204 non-null  object
 5   Item_MRP                   14204 non-null  float64
 6   Outlet_Identifier          14204 non-null  object
 7   Outlet_Establishment_Year  14204 non-null  int64
 8   Outlet_Size                10188 non-null  object
 9   Outlet_Location_Type       14204 non-null  object
 10  Outlet_Type                14204 non-null  object
 11  Item_Outlet_Sales          8523 non-null   float64
 12  source                     14204 non-null  object
dtypes: float64(4), int64(1), object(8)
memory usage: 1.4+ MB
```

In [16]:
```
# --- Percentage of test set in the dataset ---
print('.: Percentage of Test Dataset in Combined Dataset :.')
print('*' * 52)
train.Item_Type.value_counts(dropna=False)
print(dataset["Item_Outlet_Sales"].isnull().sum()/dataset.shape[0]*100,"%")
```

```
.: Percentage of Test Dataset in Combined Dataset :.
****************************************************
39.995775837792166 %
```

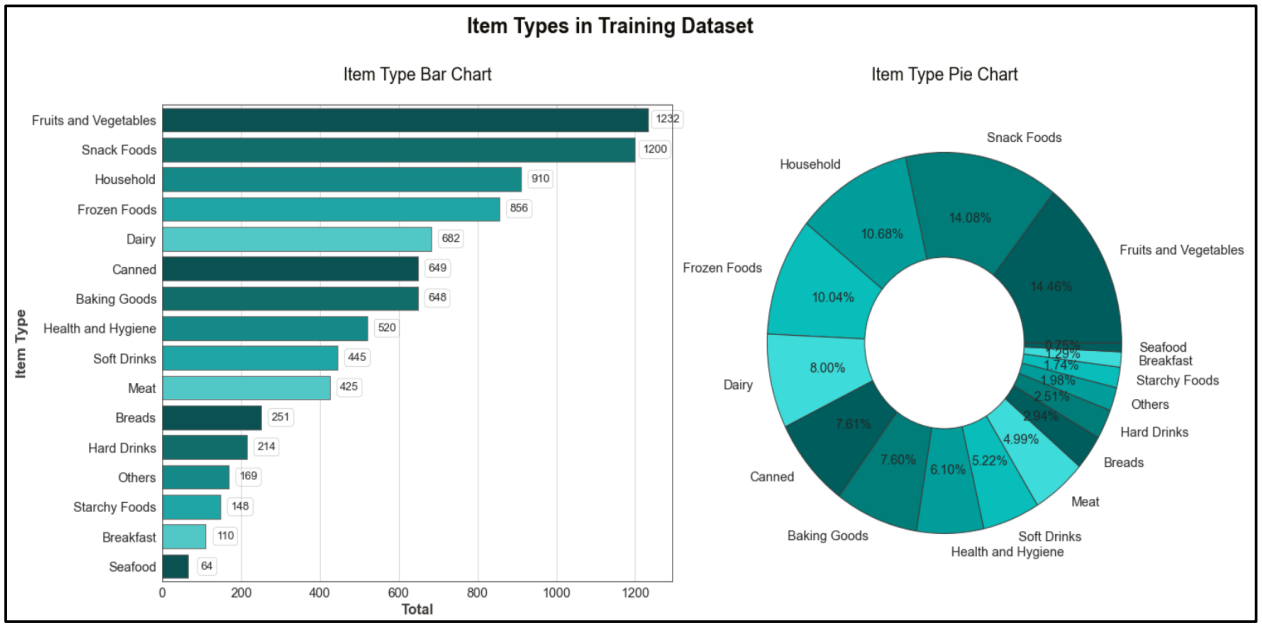
```
In [17]: # --- Data Visualization for Item_Type in train dataset ---
         # --- Setting Colors, Labels, Order ---
         colors=teal_grad
         label1=train['Item_Type'].dropna().unique()
         order1=train['Item_Type'].value_counts().index

         # --- Counting Categorical Labels without Dropping Null Values ---
         print('*' * 36)
         print('.: Item Type List (Train Dataset) :.')
         print('*' * 36)
         train.Item_Type.value_counts(dropna=False)
```

```
************************************
.: Item Type List (Train Dataset) :.
************************************
```

```
Out[17]: Fruits and Vegetables    1232
         Snack Foods              1200
         Household                 910
         Frozen Foods              856
         Dairy                     682
         Canned                    649
         Baking Goods              648
         Health and Hygiene        520
         Soft Drinks               445
         Meat                      425
         Breads                    251
         Hard Drinks               214
         Others                    169
         Starchy Foods             148
         Breakfast                 110
         Seafood                    64
         Name: Item_Type, dtype: int64
```

In Item_Type, there are 16 various types of items that are listed in the dataset. Also, the number of times the items are repeated in the dataset is mentioned.

```
In [18]: # --- Size for Both Figures ---
         plt.figure(figsize=(20, 10))
         plt.suptitle('Item Types in Training Dataset', fontweight='bold', fontsize='24', fontfamily='sans-serif', color=black_grad[0])

         # --- Histogram ---
         countplt = plt.subplot(1, 2, 1)
         plt.title('Item Type Bar Chart\n', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
         ax = sns.countplot(y='Item_Type', data=train, palette=colors, order=order1, edgecolor=black_grad[2])
         for rect in ax.patches:
             width, height = rect.get_width(), rect.get_height()
             x, y = rect.get_xy()
             ax.text (x+width+50, y+height/2, '{:.0f}'.format(width), horizontalalignment='center', verticalalignment='center',
                      fontsize=13.5, bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.15, boxstyle='round'))
             ax.tick_params(axis='x', labelsize=15.5)
             ax.tick_params(axis='y', labelsize=15.5)
         plt.tight_layout(rect=[0, 0.04, 1, 0.965])
         plt.xlabel('Total', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
         plt.ylabel('Item Type', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
         plt.grid(axis='x');
         countplt

         # --- Pie Chart ---
         plt.subplot(1, 2, 2)
         plt.title('Item Type Pie Chart\n', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
         plt.pie(train['Item_Type'].value_counts(), labels=order1, colors=colors, pctdistance=0.67, autopct='%.2f%%',
                 wedgeprops=dict(edgecolor=black_grad[1]), textprops={'fontsize':15.5})
         centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
         plt.gcf().gca().add_artist(centre);
```



In [19]:
```
# --- Data Visualization for Item_Type in test dataset ---
# --- Setting Colors, Labels, Order ---
colors=teal_grad
label2=test['Item_Type'].dropna().unique()
order2=test['Item_Type'].value_counts().index

# --- Counting Categorical Labels without Dropping Null Values ---
print('*' * 35)
print('.: Item Type List (Test Dataset) :.')
print('*' * 35)
test.Item_Type.value_counts(dropna=False)
```

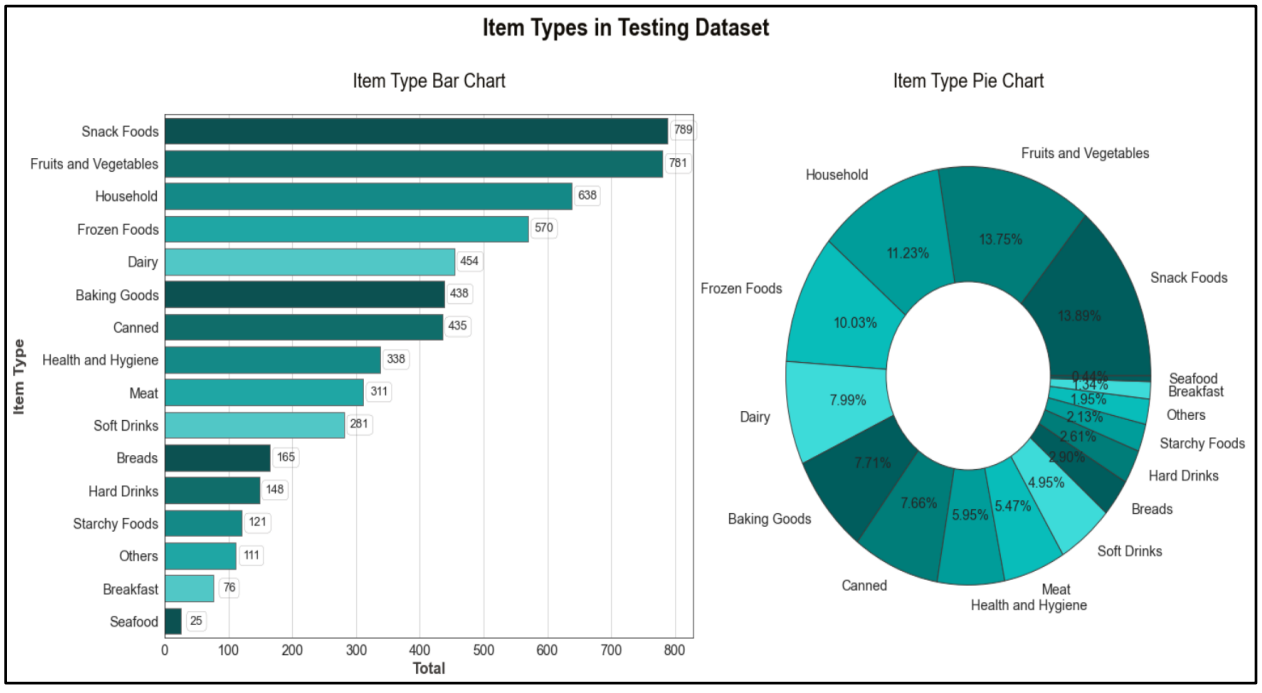
```
***********************************
.: Item Type List (Test Dataset) :.
***********************************
```

Out[19]:

Snack Foods	789
Fruits and Vegetables	781
Household	638
Frozen Foods	570
Dairy	454
Baking Goods	438
Canned	435
Health and Hygiene	338
Meat	311
Soft Drinks	281
Breads	165
Hard Drinks	148
Starchy Foods	121
Others	111
Breakfast	76
Seafood	25

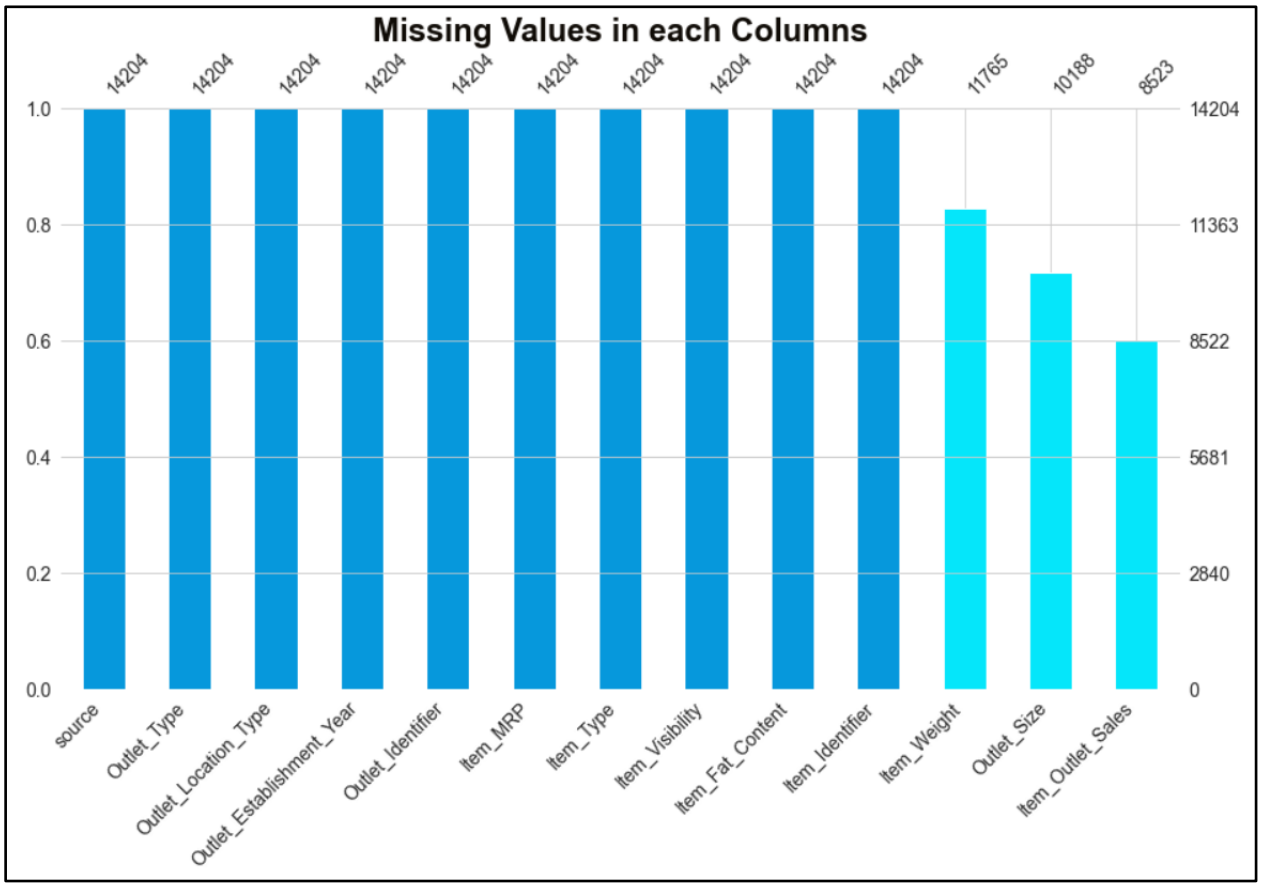
Name: Item_Type, dtype: int64

In [20]:
```
# --- Size for Both Figures ---
plt.figure(figsize=(20, 10))
plt.suptitle('Item Types in Testing Dataset', fontweight='bold', fontsize='24', fontfamily='sans-serif', color=black_grad[0])

# --- Histogram ---
countplt = plt.subplot(1, 2, 1)
plt.title('Item Type Bar Chart\n', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
ax = sns.countplot(y='Item_Type', data=test, palette=colors, order=order2, edgecolor=black_grad[2])
for rect in ax.patches:
    width, height = rect.get_width(), rect.get_height()
    x, y = rect.get_xy()
    ax.text (x+width+25, y+height/2, '{:.0f}'.format(width), horizontalalignment='center', verticalalignment='center',
             fontsize=13.5, bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.15, boxstyle='round'))
    ax.tick_params(axis='x', labelsize=15.5)
    ax.tick_params(axis='y', labelsize=15.5)
plt.tight_layout(rect=[0, 0.04, 1, 0.965])
plt.xlabel('Total', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Item Type', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.grid(axis='x');
countplt

# --- Pie Chart ---
plt.subplot(1, 2, 2)
plt.title('Item Type Pie Chart\n', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
plt.pie(test['Item_Type'].value_counts(), labels=order2, colors=colors, pctdistance=0.67, autopct='%.2f%%',
        wedgeprops=dict(edgecolor=black_grad[1]), textprops={'fontsize':15.5})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre);
```



Finding the missing values:

```
In [21]: sns.set_style('whitegrid')

         # --- Ploting Missing Values ---
         mso.bar(dataset, fontsize=14, color=[color_mix[0], color_mix[0], color_mix[0], color_mix[0], color_mix[0], color_mix[0],
                                              color_mix[0], color_mix[0], color_mix[0], color_mix[0], color_mix[1], color_mix[1],
                                              color_mix[1]], figsize=(15, 8), sort='descending', labels=True)

         # --- Title Settings ---
         plt.title('Missing Values in each Columns', fontweight='bold', fontsize='24', fontfamily='sans-serif', color=black_grad[0])
         plt.grid(axis='both', alpha=0);

         # ---Checking Null Values in Combined Dataset ---
         print('.: Combined Dataset Null Values:.')
         print('*' * 33)
         dataset.isnull().sum()

         .: Combined Dataset Null Values:.
         *********************************

Out[21]: Item_Identifier               0
         Item_Weight                2439
         Item_Fat_Content              0
         Item_Visibility               0
         Item_Type                     0
         Item_MRP                      0
         Outlet_Identifier             0
         Outlet_Establishment_Year     0
         Outlet_Size                4016
         Outlet_Location_Type          0
         Outlet_Type                   0
         Item_Outlet_Sales          5681
         source                        0
         dtype: int64
```



Almost all the columns have no missing values except the “Item_Weight”, “Outlet_Size” and “Item_Outlet_Sales” columns.

The total of missing values in each column is less than 25%, which means that imputation can still be done to fill in the missing values in the two columns(“Item_Weight” and “Outlet_Size”). There are 2439 missing values in the “Item_Weight” column and 4016 missing values in the “Outlet_Size” column.

Replacing missing values with substitute values:

The “Item_Weight” column missing values can be replaced by their mean and the “Outlet_Size” column missing values can be replaced by their mode.

In [22]:
```
# --- Calculating mean value of Item_Weight for every Unique item present in dataset ---
# --- pivot_table() allows us to create a table that contains the mean values of identifiers ---
avg = pd.pivot_table(dataset,values='Item_Weight', index='Item_Identifier',aggfunc='mean')
avg
```

Out[22]:

Item_Identifier	Item_Weight
DRA12	11.600
DRA24	19.350
DRA59	8.270
DRB01	7.390
DRB13	6.115
...	...
NCZ30	6.590
NCZ41	19.850
NCZ42	10.500
NCZ53	9.600
NCZ54	14.650

1559 rows × 1 columns

In [23]:
```
# --- Cross verifying the mean value by checking a particular item in dataset ---
df=dataset[dataset['Item_Identifier'].str.contains("DRA24")]
df.style.set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source
1148	DRA24	19.350000	Regular	0.040154	Soft Drinks	164.686800	OUT017	2007	nan	Tier 2	Supermarket Type1	1146.507600	Train
2879	DRA24	nan	Regular	0.069909	Soft Drinks	163.286800	OUT019	1985	Small	Tier 1	Grocery Store	491.360400	Train
4130	DRA24	19.350000	Regular	0.066832	Soft Drinks	163.886800	OUT010	1998	nan	Tier 3	Grocery Store	327.573600	Train
4416	DRA24	nan	Regular	0.039735	Soft Drinks	165.786800	OUT027	1985	Medium	Tier 3	Supermarket Type3	4913.604000	Train
4900	DRA24	19.350000	Regular	0.039921	Soft Drinks	163.386800	OUT035	2004	Small	Tier 2	Supermarket Type1	3439.522800	Train
6863	DRA24	19.350000	Regular	0.039990	Soft Drinks	165.086800	OUT049	1999	Medium	Tier 1	Supermarket Type1	982.720800	Train
8195	DRA24	19.350000	Regular	0.039895	Soft Drinks	162.486800	OUT013	1987	High	Tier 3	Supermarket Type1	4422.243600	Train
9734	DRA24	19.350000	Regular	0.040009	Soft Drinks	163.286800	OUT045	2002	nan	Tier 2	Supermarket Type1	nan	Test
11370	DRA24	19.350000	Regular	0.040091	Soft Drinks	163.686800	OUT018	2009	Medium	Tier 3	Supermarket Type2	nan	Test
13061	DRA24	19.350000	Regular	0.039928	Soft Drinks	164.586800	OUT046	1997	Small	Tier 1	Supermarket Type1	nan	Test

In [24]:
```
# --- Updating the null weight values of dataset with mean values ---
dataset[:][dataset['Item_Identifier'] == 'DRI11']
def impute(cols):
    Weight = cols[1]
    Identifier = cols[0]

    if pd.isnull(Weight):
        return avg['Item_Weight'][avg.index == Identifier]
    else:
        return Weight

print ('Orignal Number of missing values in Item_Weight:',sum(dataset['Item_Weight'].isnull()))

# --- Applying the impute() function to impute null values of Item_Weight ---
dataset['Item_Weight'] = dataset[['Item_Identifier','Item_Weight']].apply(impute,axis=1).astype(float)

print ('Number of missing values in Item_Weight after imputation: ',sum(dataset['Item_Weight'].isnull()))
```

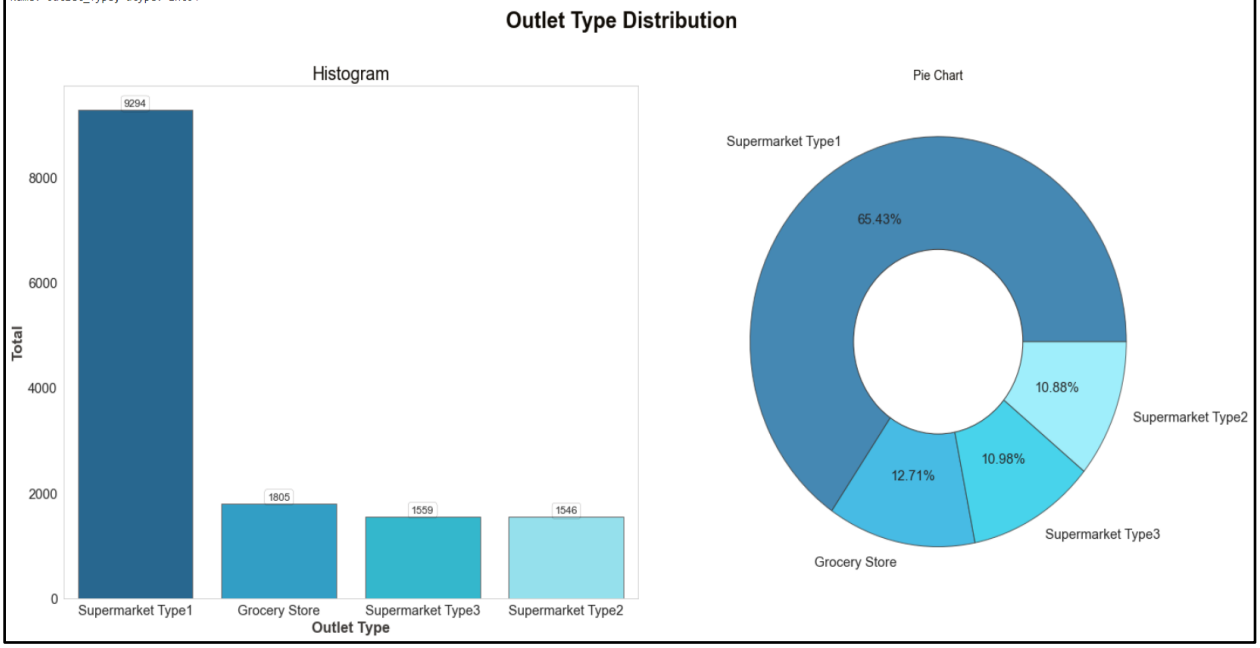
```
Orignal Number of missing values in Item_Weight: 2439
Number of missing values in Item_Weight after imputation:  0
```

In [25]:
```
# --- Cross verifying the mean value is updated or not by checking a particular item in dataset ---
df=dataset[dataset['Item_Identifier'].str.contains("DRA24")]
df.style.set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source
1148	DRA24	19.350000	Regular	0.040154	Soft Drinks	164.686800	OUT017	2007	nan	Tier 2	Supermarket Type1	1146.507600	Train
2879	DRA24	19.350000	Regular	0.069909	Soft Drinks	163.286800	OUT019	1985	Small	Tier 1	Grocery Store	491.360400	Train
4130	DRA24	19.350000	Regular	0.066832	Soft Drinks	163.886800	OUT010	1998	nan	Tier 3	Grocery Store	327.573600	Train
4416	DRA24	19.350000	Regular	0.039735	Soft Drinks	165.786800	OUT027	1985	Medium	Tier 3	Supermarket Type3	4913.604000	Train
4900	DRA24	19.350000	Regular	0.039921	Soft Drinks	163.386800	OUT035	2004	Small	Tier 2	Supermarket Type1	3439.522800	Train
6863	DRA24	19.350000	Regular	0.039990	Soft Drinks	165.086800	OUT049	1999	Medium	Tier 1	Supermarket Type1	982.720800	Train
8195	DRA24	19.350000	Regular	0.039895	Soft Drinks	162.486800	OUT013	1987	High	Tier 3	Supermarket Type1	4422.243600	Train
9734	DRA24	19.350000	Regular	0.040009	Soft Drinks	163.286800	OUT045	2002	nan	Tier 2	Supermarket Type1	nan	Test
11370	DRA24	19.350000	Regular	0.040091	Soft Drinks	163.686800	OUT018	2009	Medium	Tier 3	Supermarket Type2	nan	Test
13061	DRA24	19.350000	Regular	0.039928	Soft Drinks	164.586800	OUT046	1997	Small	Tier 1	Supermarket Type1	nan	Test

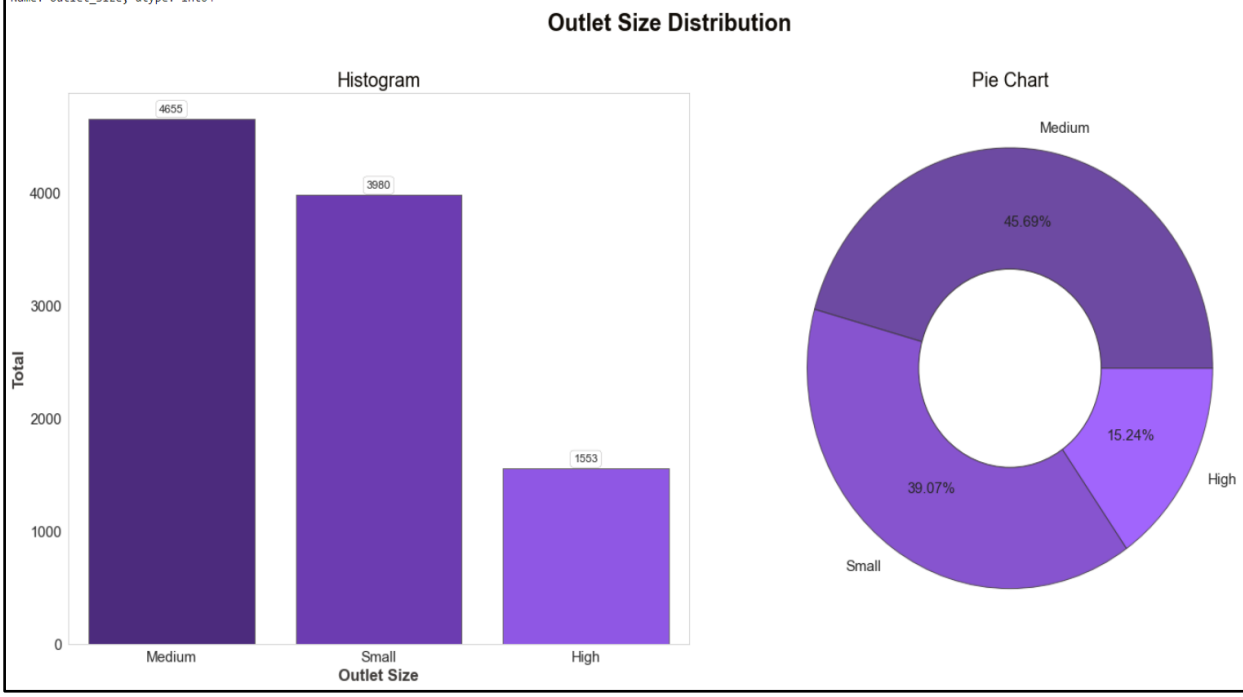
In [26]:
```
# --- Data Visualization for Outlet Type of Comined Dataset ---
# --- Setting Colors, Labels, Order ---
colors=cyan_grad[1:5]
labels=dataset['Outlet_Type'].dropna().unique()
order=dataset['Outlet_Type'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(20, 10))
plt.suptitle('Outlet Type Distribution', fontweight='bold', fontsize='24', fontfamily='sans-serif', color=black_grad[0])

# --- Histogram ---
countplt = plt.subplot(1, 2, 1)
plt.title('Histogram', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
ax = sns.countplot(x='Outlet_Type', data=dataset, palette=colors, order=order, edgecolor=black_grad[2])
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2, rect.get_height()+60,rect.get_height(), horizontalalignment='center',
             fontsize=12, bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.15, boxstyle='round'))
    ax.tick_params(axis='x', labelsize=15.5)
    ax.tick_params(axis='y', labelsize=15.5)
plt.tight_layout(rect=[0, 0.04, 1, 0.96])
plt.xlabel('Outlet Type', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.grid(axis='y')
countplt

# --- Pie Chart ---
plt.subplot(1, 2, 2)
plt.title('Pie Chart', fontsize=14, fontfamily='sans-serif', color=black_grad[0])
plt.pie(dataset['Outlet_Type'].value_counts(), colors=colors, labels=order, pctdistance=0.67, autopct='%.2f%%',
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]), textprops={'fontsize':15.5})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre);

# --- Count Categorical Labels without Dropping Null Values ---
print('.: Outlet Type Combined Data :.')
print('*' * 31)
dataset.Outlet_Type.value_counts(dropna=False)
```

```
.: Outlet Type Combined Data :.
*******************************
Supermarket Type1    9294
Grocery Store        1805
Supermarket Type3    1559
Supermarket Type2    1546
Name: Outlet_Type, dtype: int64
```



In [27]:
```
# --- Data Visualization for Outlet Size of Combined Dataset(including null values) ---
# --- Setting Colors, Labels, Order ---
colors=purple_grad
labels=dataset['Outlet_Size'].dropna().unique()
order=dataset['Outlet_Size'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(20,10))
plt.suptitle('Outlet Size Distribution', fontweight='bold', fontsize='24', fontfamily='sans-serif', color=black_grad[0])

# --- Histogram ---
countplt = plt.subplot(1, 2, 1)
plt.title('Histogram', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
ax = sns.countplot(x='Outlet_Size', data=dataset, palette=colors, order=order, edgecolor=black_grad[2])
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2, rect.get_height()+60,rect.get_height(), horizontalalignment='center',
             fontsize=12, bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.15, boxstyle='round'))
    ax.tick_params(axis='x', labelsize=15.5)
    ax.tick_params(axis='y', labelsize=15.5)
plt.tight_layout(rect=[0, 0.04, 1, 0.96])
plt.xlabel('Outlet Size', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.grid(axis='y')
countplt

# --- Pie Chart ---
plt.subplot(1, 2, 2)
plt.title('Pie Chart', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
plt.pie(dataset['Outlet_Size'].value_counts(), colors=colors, labels=order, pctdistance=0.67, autopct='%.2f%%',
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]), textprops={'fontsize':15.5})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre);

# --- Count Categorical Labels without Dropping Null Values ---
print('.: Outlet Size Total :.')
print('*' * 23)
dataset.Outlet_Size.value_counts(dropna=False)
```

```
.: Outlet Size Total :.
***********************
Medium    4655
NaN       4016
Small     3980
High      1553
Name: Outlet_Size, dtype: int64
```



In [28]:
```
# --- Calculating mode value of Outlet Size for every Unique Outlet Type present in dataset ---
# --- pivot_table() allows us to create a table that contains the mode of identifiers ---
mode = pd.pivot_table(dataset, values='Outlet_Size', columns='Outlet_Type',aggfunc=lambda x:x.mode())
mode
```

Out[28]:

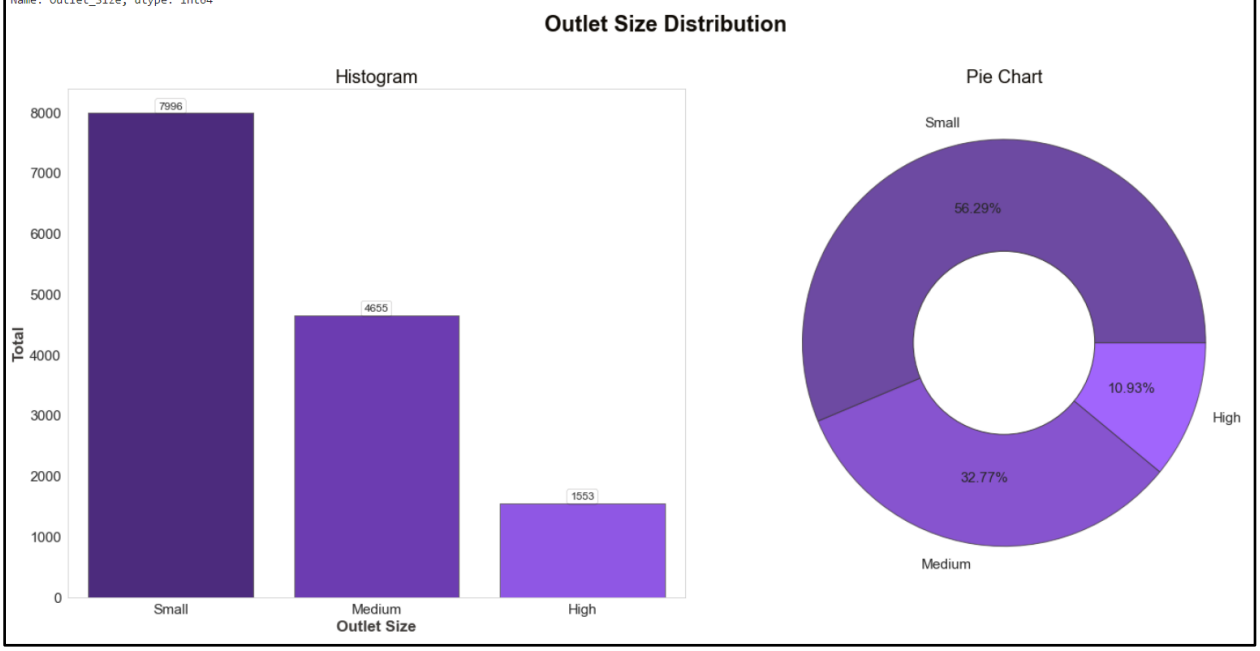
Outlet_Type	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3
Outlet_Size	Small	Small	Medium	Medium

In [29]:
```
# --- Updating the null outlet size values of dataset with mode values ---
# --- Imputing Outlet_Size missing values with their mode ---
def impute_mode(cols):
    size = cols[1]
    Type = cols[0]

    if pd.isnull(size):
        return mode.loc['Outlet_Size'][mode.columns == Type][0]
    else:
        return size
print ('Orignal Number of missing values in Outlet_Size:',sum(dataset['Outlet_Size'].isnull()))

# Applying the impute() function to impute null values of Item_Weight
dataset['Outlet_Size'] = dataset[['Outlet_Type','Outlet_Size']].apply(impute_mode,axis=1)

print ('Number of missing values in Outlet_Size after imputation: ',sum(dataset['Outlet_Size'].isnull()))
```

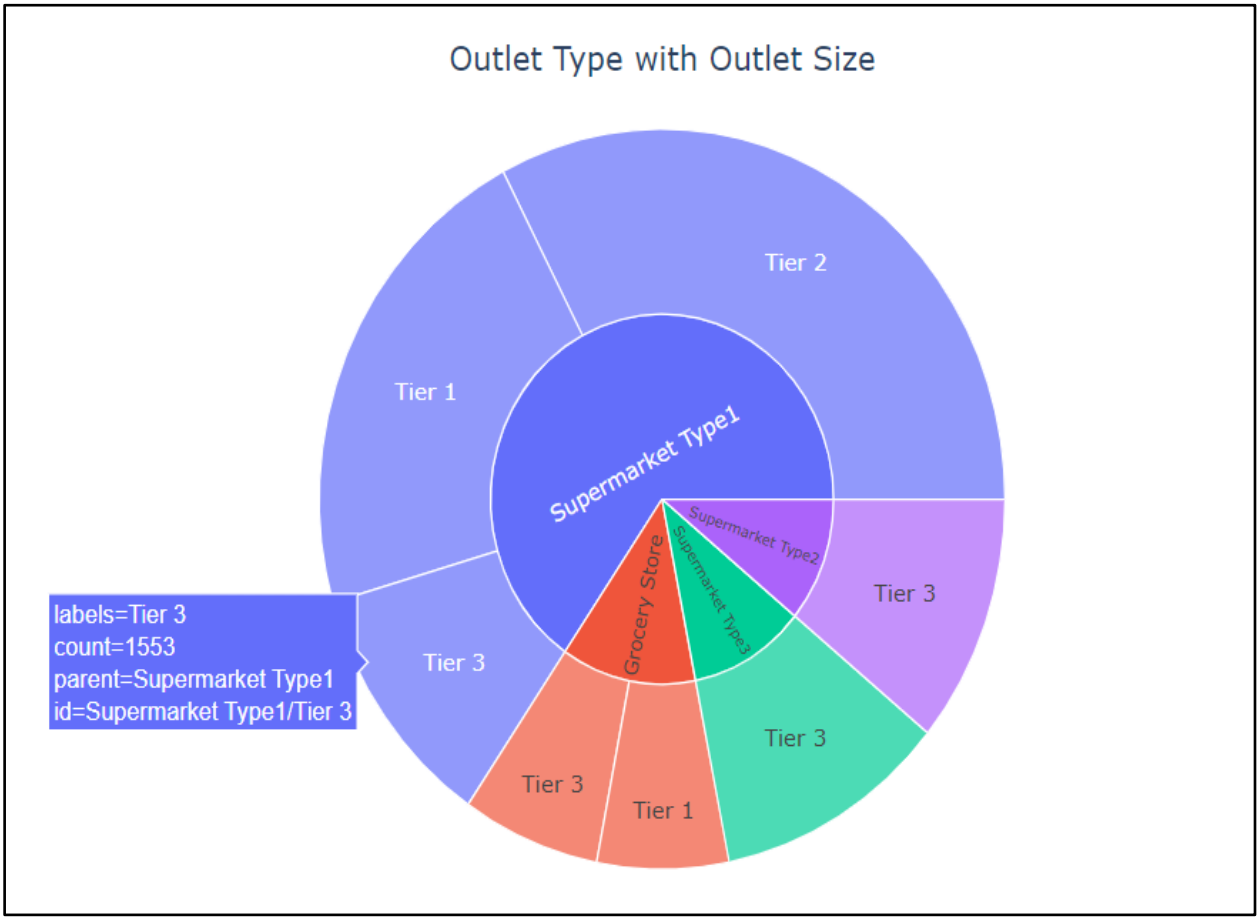
```
Orignal Number of missing values in Outlet_Size: 4016
Number of missing values in Outlet_Size after imputation:  0
```

In [30]:
```
# --- Data Visualization for Outlet Size of Combined Dataset ---
# --- Setting Colors, Labels, Order ---
colors=purple_grad
labels=dataset['Outlet_Size'].dropna().unique()
order=dataset['Outlet_Size'].value_counts().index

# --- Size for Both Figures ---
plt.figure(figsize=(20,10))
plt.suptitle('Outlet Size Distribution', fontweight='bold', fontsize='24', fontfamily='sans-serif', color=black_grad[0])

# --- Histogram ---
countplt = plt.subplot(1, 2, 1)
plt.title('Histogram', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
ax = sns.countplot(x='Outlet_Size', data=dataset, palette=colors, order=order, edgecolor=black_grad[2])
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2, rect.get_height()+60,rect.get_height(), horizontalalignment='center',
             fontsize=12, bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.15, boxstyle='round'))
    ax.tick_params(axis='x', labelsize=15.5)
    ax.tick_params(axis='y', labelsize=15.5)
plt.tight_layout(rect=[0, 0.04, 1, 0.96])
plt.xlabel('Outlet Size', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.grid(axis='y')
countplt

# --- Pie Chart ---
plt.subplot(1, 2, 2)
plt.title('Pie Chart', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
plt.pie(dataset['Outlet_Size'].value_counts(), colors=colors, labels=order, pctdistance=0.67, autopct='%.2f%%',
        wedgeprops=dict(alpha=0.8, edgecolor=black_grad[1]), textprops={'fontsize':15.5})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[1])
plt.gcf().gca().add_artist(centre);

# --- Count Categorical Labels w/out Dropping Null Values ---
print('.: Outlet Size Total :.')
print('*' * 25)
dataset.Outlet_Size.value_counts(dropna=False)
```

```
.: Outlet Size Total :.
*************************
Small     7996
Medium    4655
High      1553
Name: Outlet_Size, dtype: int64
```



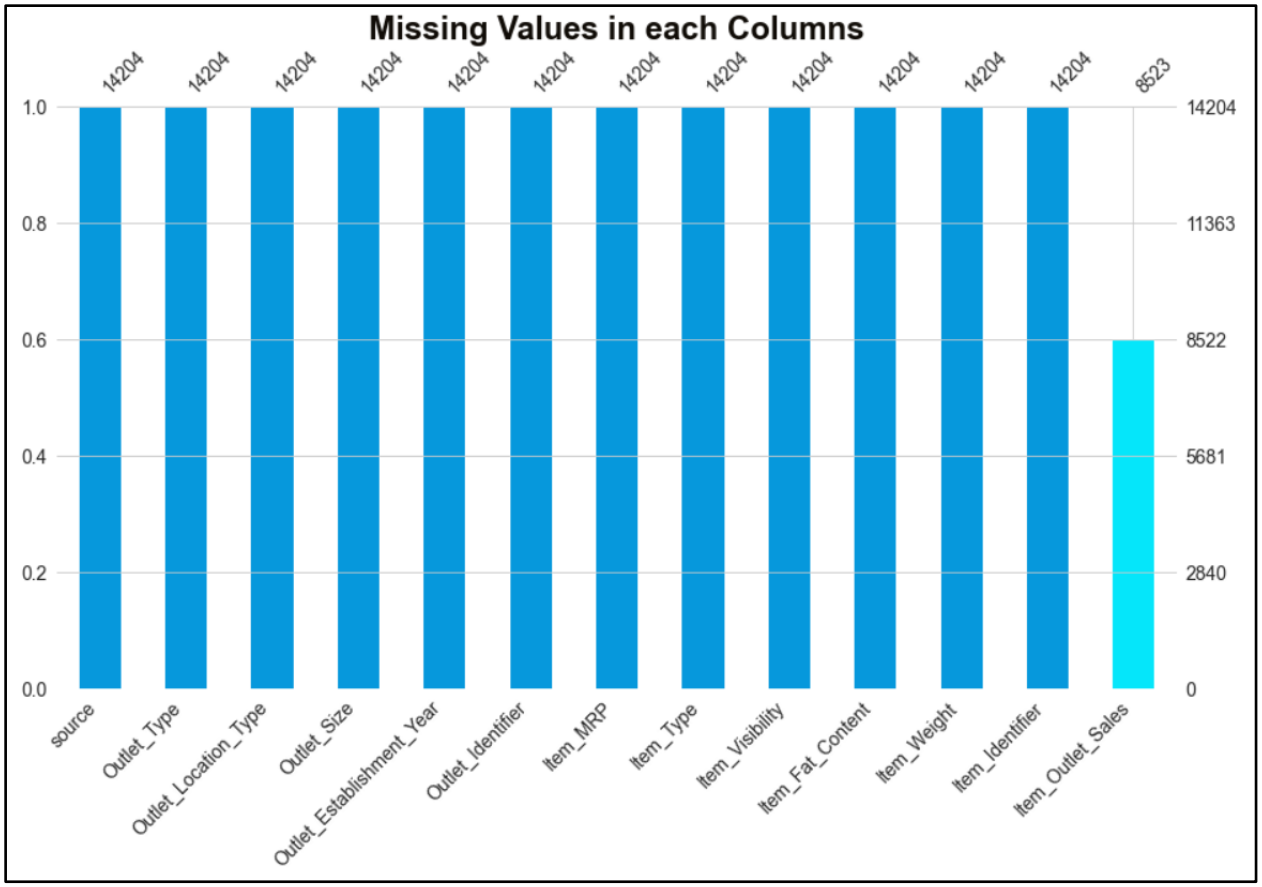
In [31]:
```
# --- Data Visualization for Outlet Type and Outlet Size of Combined Dataset ---
fig1=px.sunburst(dataset,path=['Outlet_Type','Outlet_Location_Type'],color_continuous_scale='RdBu')
fig1.update_layout(title='Outlet Type with Outlet Size',title_x=0.5)
fig1.show()
```



In [32]:
```
# --- Ploting / Checking Missing Values ---
mso.bar(dataset, fontsize=14, color=[color_mix[0], color_mix[0], color_mix[0], color_mix[0], color_mix[0], color_mix[0],
                                     color_mix[0], color_mix[0], color_mix[0], color_mix[0], color_mix[0], color_mix[0],
                                     color_mix[1]], figsize=(15, 8), sort='descending', labels=True)

# --- Title & Subtitle Settings ---
plt.title('Missing Values in each Columns', fontweight='bold', fontsize='24', fontfamily='sans-serif', color=black_grad[0])
plt.grid(axis='both', alpha=0);

# --- Checking Null Values in Combined Dataset ---
print('.: Combined Dataset Null Values:.')
print('*' * 33)
dataset.isnull().sum()
```

```
.: Combined Dataset Null Values:.
*********************************
```

Out[32]:
```
Item_Identifier                0
Item_Weight                    0
Item_Fat_Content               0
Item_Visibility                0
Item_Type                      0
Item_MRP                       0
Outlet_Identifier              0
Outlet_Establishment_Year      0
Outlet_Size                    0
Outlet_Location_Type           0
Outlet_Type                    0
Item_Outlet_Sales           5681
source                         0
dtype: int64
```



In [33]:
```
# --- Reading Combined Dataset after data processing (Adding missing values) ---
dataset.head(10).style.background_gradient(cmap='BuPu').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source
0	FDA15	9.300000	Low Fat	0.016047	Dairy	249.809200	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.138000	Train
1	DRC01	5.920000	Regular	0.019278	Soft Drinks	48.269200	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.422800	Train
2	FDN15	17.500000	Low Fat	0.016760	Meat	141.618000	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.270000	Train
3	FDX07	19.200000	Regular	0.000000	Fruits and Vegetables	182.095000	OUT010	1998	Small	Tier 3	Grocery Store	732.380000	Train
4	NCD19	8.930000	Low Fat	0.000000	Household	53.861400	OUT013	1987	High	Tier 3	Supermarket Type1	994.705200	Train
5	FDP36	10.395000	Regular	0.000000	Baking Goods	51.400800	OUT018	2009	Medium	Tier 3	Supermarket Type2	556.608800	Train
6	FDO10	13.650000	Regular	0.012741	Snack Foods	57.658800	OUT013	1987	High	Tier 3	Supermarket Type1	343.552800	Train
7	FDP10	19.000000	Low Fat	0.127470	Snack Foods	107.762200	OUT027	1985	Medium	Tier 3	Supermarket Type3	4022.763600	Train
8	FDH17	16.200000	Regular	0.016687	Frozen Foods	96.972600	OUT045	2002	Small	Tier 2	Supermarket Type1	1076.598600	Train
9	FDU28	19.200000	Regular	0.094450	Frozen Foods	187.821400	OUT017	2007	Small	Tier 2	Supermarket Type1	4710.535000	Train

In [34]:
```
# --- Reading Combined Dataset after data processing (Adding missing values) ---
dataset.tail(10).style.background_gradient(cmap='BuPu').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source
14194	FDA01	15.000000	reg	0.054463	Canned	59.590400	OUT049	1999	Medium	Tier 1	Supermarket Type1	nan	Test
14195	NCH42	6.860000	Low Fat	0.036594	Household	231.101000	OUT049	1999	Medium	Tier 1	Supermarket Type1	nan	Test
14196	FDF46	7.070000	Low Fat	0.094053	Snack Foods	116.083400	OUT018	2009	Medium	Tier 3	Supermarket Type2	nan	Test
14197	DRL35	15.700000	Low Fat	0.030704	Hard Drinks	43.277000	OUT046	1997	Small	Tier 1	Supermarket Type1	nan	Test
14198	FDW46	13.000000	Regular	0.070411	Snack Foods	63.448400	OUT049	1999	Medium	Tier 1	Supermarket Type1	nan	Test
14199	FDB58	10.500000	Regular	0.013496	Snack Foods	141.315400	OUT046	1997	Small	Tier 1	Supermarket Type1	nan	Test
14200	FDD47	7.600000	Regular	0.142991	Starchy Foods	169.144800	OUT018	2009	Medium	Tier 3	Supermarket Type2	nan	Test
14201	NCO17	10.000000	Low Fat	0.073529	Health and Hygiene	118.744000	OUT045	2002	Small	Tier 2	Supermarket Type1	nan	Test
14202	FDJ26	15.300000	Regular	0.000000	Canned	214.621800	OUT017	2007	Small	Tier 2	Supermarket Type1	nan	Test
14203	FDU37	9.500000	Regular	0.104720	Canned	79.796000	OUT045	2002	Small	Tier 2	Supermarket Type1	nan	Test

Modifying the inconsistent values:

In [35]:
```
# --- Counting Categorical Labels (Outlet Identifier) without Dropping Null Values ---
print('.: Outlet_Identifier Total :.')
print('*' * 29)
dataset.Outlet_Identifier.value_counts(dropna=False)
```

```
.: Outlet_Identifier Total :.
*****************************
```

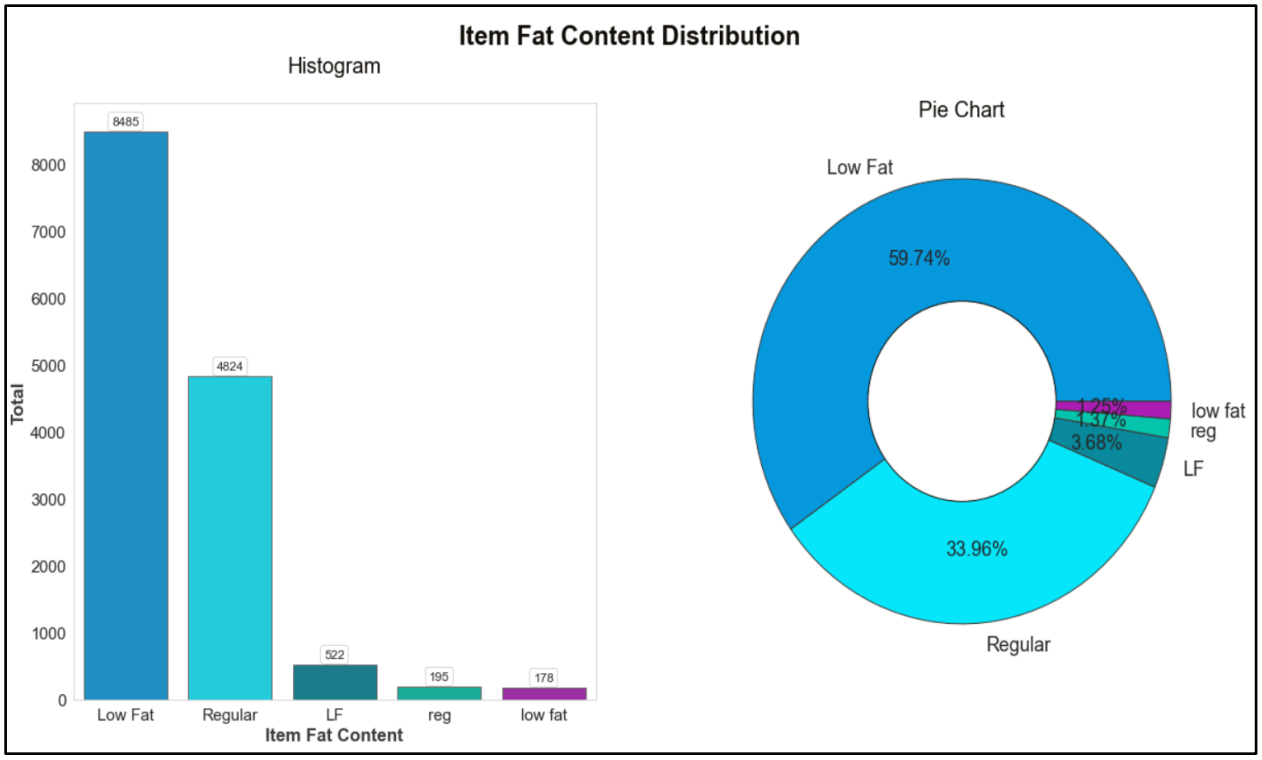
Out[35]:
```
OUT027    1559
OUT013    1553
OUT049    1550
OUT046    1550
OUT035    1550
OUT045    1548
OUT018    1546
OUT017    1543
OUT010     925
OUT019     880
Name: Outlet_Identifier, dtype: int64
```

In [36]:
```
# --- Data Visualization for Item Fat Content of Combined Dataset(including similar values) ---
# --- Setting Colors, Labels, Order ---
colors=color_mix
labels=dataset['Item_Fat_Content'].dropna().unique()
order=dataset['Item_Fat_Content'].value_counts().index

# --- Counting Categorical Labels (Item Fat Content) without Dropping Null Values ---
print('.: Item Fat Content Total :.')
print('*' * 29)
dataset.Item_Fat_Content.value_counts(dropna=False)
```

```
.: Item Fat Content Total :.
*****************************
```

Out[36]:
```
Low Fat    8485
Regular    4824
LF          522
reg         195
low fat     178
Name: Item_Fat_Content, dtype: int64
```

In [37]:
```
# --- Size for Both Figures ---
plt.figure(figsize=(20, 10))
plt.suptitle('Item Fat Content Distribution', fontweight='bold', fontsize='24', fontfamily='sans-serif', color=black_grad[0])

# --- Histogram ---
countplt = plt.subplot(1, 2, 1)
plt.title('Histogram\n', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
ax = sns.countplot(x='Item_Fat_Content', data=dataset, palette=colors, order=order, edgecolor=black_grad[2])
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2, rect.get_height()+100,rect.get_height(), horizontalalignment='center',
             fontsize=12, bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.15, boxstyle='round'))
    ax.tick_params(axis='x', labelsize=15.5)
    ax.tick_params(axis='y', labelsize=15.5)
plt.xlabel('Item Fat Content', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.grid(axis='y')
countplt

# --- Pie Chart ---
plt.subplot(1, 2, 2)
plt.title('Pie Chart', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
plt.pie(dataset['Item_Fat_Content'].value_counts(), colors=colors, labels=order, pctdistance=0.67, autopct='%.2f%%',
        wedgeprops=dict(edgecolor=black_grad[1]), textprops={'fontsize':18})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[0])
plt.gcf().gca().add_artist(centre)
```



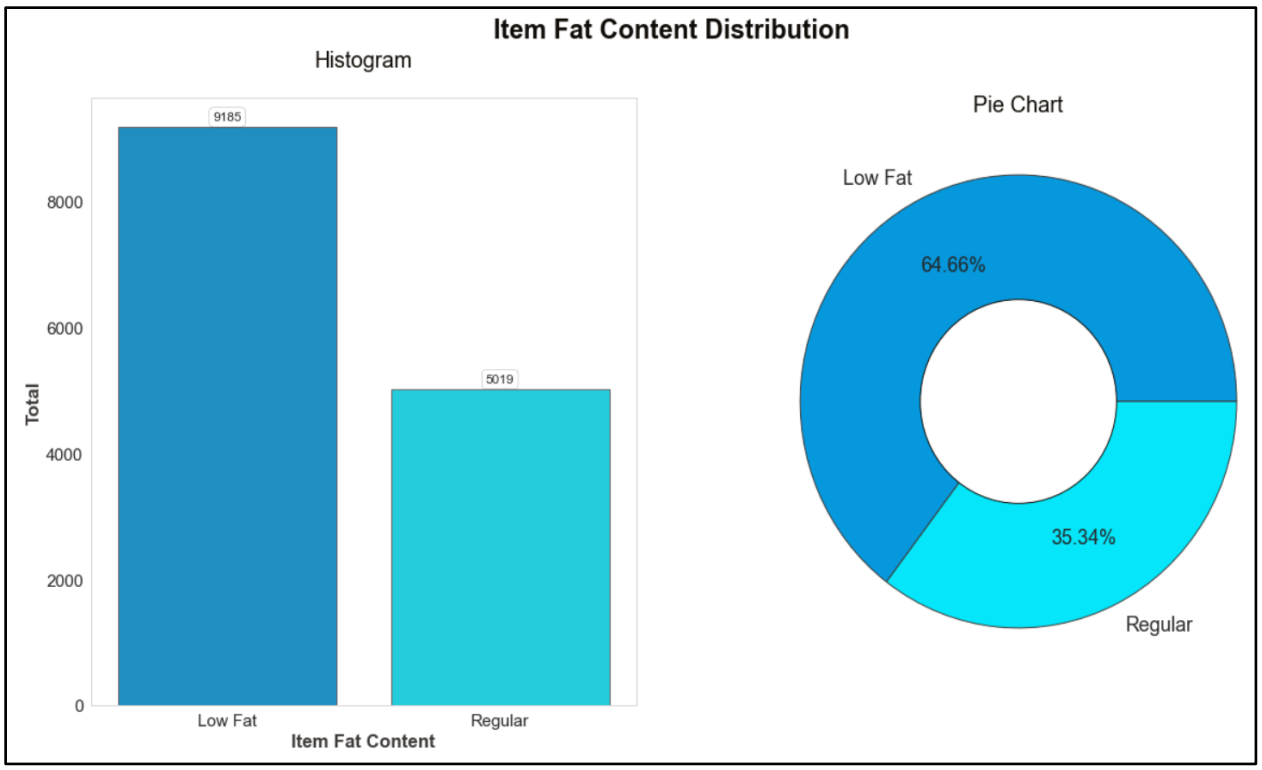
Here, there are typos line case sensitive words and shortcuts. 'Low Fat' and 'Regular' must be the real unique values. Before implementing the model, this adjustment is required.

In [38]:
```
# --- Data Visualization for Item Fat Content of Combined Dataset ---
# --- Replacing similar values with a specific value ---
dataset['Item_Fat_Content'] = dataset['Item_Fat_Content'].replace({'low fat':'Low Fat','reg':'Regular','LF':'Low Fat'})

colors=color_mix
labels=dataset['Item_Fat_Content'].dropna().unique()
order=dataset['Item_Fat_Content'].value_counts().index

print('.: Item Fat Content Total :.')
print('*' * 29)
dataset.Item_Fat_Content.value_counts()
```

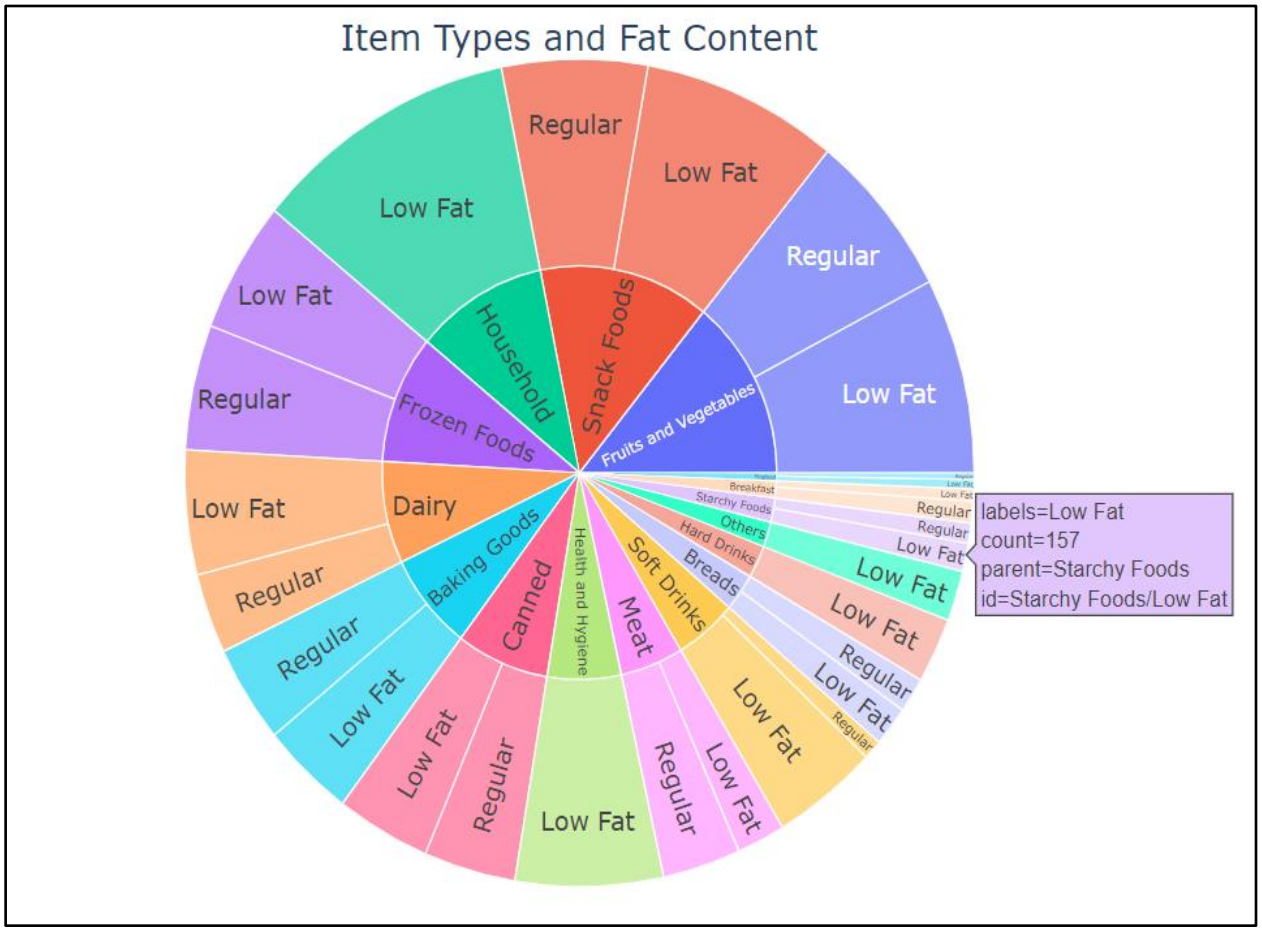
```
.: Item Fat Content Total :.
*****************************
```

Out[38]:
```
Low Fat    9185
Regular    5019
Name: Item_Fat_Content, dtype: int64
```

In [39]:
```
# --- Size for Both Figures ---
plt.figure(figsize=(20, 10))
plt.suptitle('Item Fat Content Distribution', fontweight='bold', fontsize='24', fontfamily='sans-serif', color=black_grad[0])

# --- Histogram ---
countplt = plt.subplot(1, 2, 1)
plt.title('Histogram\n', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
ax = sns.countplot(x='Item_Fat_Content', data=dataset, palette=colors, order=order, edgecolor=black_grad[2])
for rect in ax.patches:
    ax.text (rect.get_x()+rect.get_width()/2, rect.get_height()+100,rect.get_height(), horizontalalignment='center',
             fontsize=12, bbox=dict(facecolor='none', edgecolor=black_grad[0], linewidth=0.15, boxstyle='round'))
    ax.tick_params(axis='x', labelsize=15.5)
    ax.tick_params(axis='y', labelsize=15.5)
plt.xlabel('Item Fat Content', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.ylabel('Total', fontweight='bold', fontsize=17, fontfamily='sans-serif', color=black_grad[1])
plt.grid(axis='y')
countplt

# --- Pie Chart ---
plt.subplot(1, 2, 2)
plt.title('Pie Chart', fontsize=20, fontfamily='sans-serif', color=black_grad[0])
plt.pie(dataset['Item_Fat_Content'].value_counts(), colors=colors, labels=order, pctdistance=0.67, autopct='%.2f%%',
        wedgeprops=dict(edgecolor=black_grad[1]), textprops={'fontsize':18})
centre=plt.Circle((0, 0), 0.45, fc='white', edgecolor=black_grad[0])
plt.gcf().gca().add_artist(centre)
```




```
In [40]: # --- Data Visualization for Item Types and Fat Content of Combined Dataset ---
df_type=dataset.groupby('Item_Type').sum().reset_index()
print('\n')
fig2=px.sunburst(dataset,path=['Item_Type','Item_Fat_Content'],color_continuous_scale='RdBu')
fig2.update_layout(title='Item Types and Fat Content', title_x=0.5,title_y=0.999,font_size=15,
                   annotations=[dict(showarrow=True,height=1010,width=900)],margin=dict(l=20, r=20, t=20, b=20))
fig2.show()
```



```
In [41]: # --- Calculating Current Total number of Years from establishment ---
         dataset['Outlet_Year'] = 2023 - dataset['Outlet_Establishment_Year']
         dataset.drop(['Outlet_Establishment_Year'],axis=1,inplace=True)
         dataset.Outlet_Year.unique()

Out[41]: array([24, 14, 25, 36, 38, 21, 16, 26, 19], dtype=int64)
```

The Outlet_Establishment_Year contains the year in which the outlet was established. If we can get information on how many years the outlet has been working, then that data would be more useful.

```
In [42]: # --- Reading Combined Dataset after data processing (Adding missing values and replacling similar values) ---
         dataset.head(10).style.background_gradient(cmap='BuPu').set_properties(**{'border': '1px solid black'})
```

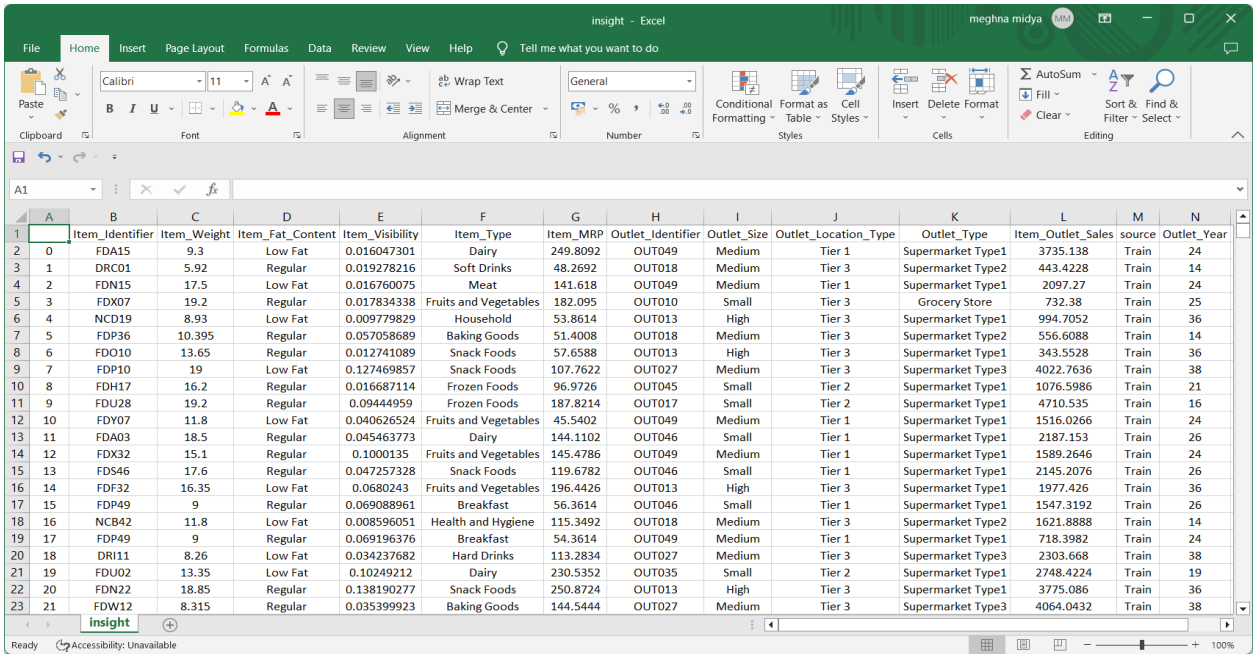
	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source	Outlet_Year
0	FDA15	9.300000	Low Fat	0.016047	Dairy	249.809200	OUT049	Medium	Tier 1	Supermarket Type1	3735.138000	Train	24
1	DRC01	5.920000	Regular	0.019278	Soft Drinks	48.269200	OUT018	Medium	Tier 3	Supermarket Type2	443.422800	Train	14
2	FDN15	17.500000	Low Fat	0.016760	Meat	141.618000	OUT049	Medium	Tier 1	Supermarket Type1	2097.270000	Train	24
3	FDX07	19.200000	Regular	0.000000	Fruits and Vegetables	182.095000	OUT010	Small	Tier 3	Grocery Store	732.380000	Train	25
4	NCD19	8.930000	Low Fat	0.000000	Household	53.861400	OUT013	High	Tier 3	Supermarket Type1	994.705200	Train	36
5	FDP36	10.395000	Regular	0.000000	Baking Goods	51.400800	OUT018	Medium	Tier 3	Supermarket Type2	556.608800	Train	14
6	FDO10	13.650000	Regular	0.012741	Snack Foods	57.658800	OUT013	High	Tier 3	Supermarket Type1	343.552800	Train	36
7	FDP10	19.000000	Low Fat	0.127470	Snack Foods	107.762200	OUT027	Medium	Tier 3	Supermarket Type3	4022.763600	Train	38
8	FDH17	16.200000	Regular	0.016687	Frozen Foods	96.972600	OUT045	Small	Tier 2	Supermarket Type1	1076.598600	Train	21
9	FDU28	19.200000	Regular	0.094450	Frozen Foods	187.821400	OUT017	Small	Tier 2	Supermarket Type1	4710.535000	Train	16

```
In [43]: # --- Calculating mean value of Item_Weight for every Unique item present in dataset ---
         vmean = dataset.pivot_table(index = "Item_Identifier",  values = "Item_Visibility")
         print(vmean)

                        Item_Visibility
         Item_Identifier
         DRA12                 0.034938
         DRA24                 0.045646
         DRA59                 0.133384
         DRB01                 0.079736
         DRB13                 0.006799
         ...                        ...
         NCZ30                 0.027302
         NCZ41                 0.056396
         NCZ42                 0.011015
         NCZ53                 0.026330
         NCZ54                 0.081345

         [1559 rows x 1 columns]
```

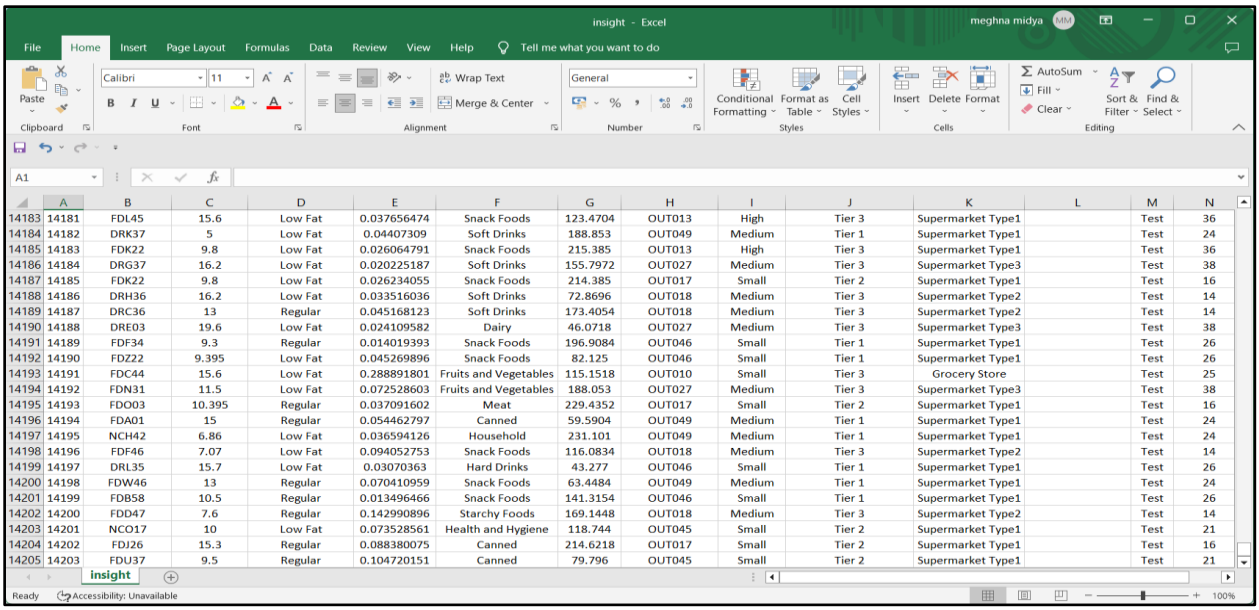
```
In [44]: # --- Updating the zero visibility item values of dataset with mean values ---
         dataset.loc[(dataset["Item_Visibility"] == 0.0),
                     "Item_Visibility"] = dataset.loc[(dataset["Item_Visibility"] == 0.0),
                                                      "Item_Identifier"].apply(lambda x : vmean.at[x, "Item_Visibility"])
```

```
In [45]: # --- Saving the updated combined dataset to new file ---
         dataset.to_csv('insight.csv')
```

```
,Item_Identifier,Item_Weight,Item_Fat_Content,Item_Visibility,Item_Type,Item_MRP,Outlet_Identifier,Outlet_Size,Outlet_Location_Type,Outlet_Type,Item_Outlet_Sales,source,Outlet_Year
0,FDA15,9.3,Low Fat,0.016047301,Dairy,249.8092,OUT049,Medium,Tier 1,Supermarket Type1,3735.138,Train,24
1,DRC01,5.92,Regular,0.019278216,Soft Drinks,48.2692,OUT018,Medium,Tier 3,Supermarket Type2,443.4228,Train,14
2,FDN15,17.5,Low Fat,0.016760075,Meat,141.618,OUT049,Medium,Tier 1,Supermarket Type1,2097.27,Train,24
3,FDX07,19.2,Regular,0.01783433788888889,Fruits and Vegetables,182.095,OUT010,Small,Tier 3,Grocery Store,732.38,Train,25
4,NCD19,8.93,Low Fat,0.009779829,Household,53.8614,OUT013,High,Tier 3,Supermarket Type1,994.7052,Train,36
5,FDP36,10.395,Regular,0.057058689,Baking Goods,51.4008,OUT018,Medium,Tier 3,Supermarket Type2,556.6088,Train,14
6,FDO10,13.65,Regular,0.012741089,Snack Foods,57.6588,OUT013,High,Tier 3,Supermarket Type1,343.5528,Train,36
7,FDP10,19.0,Low Fat,0.127469857,Snack Foods,107.7622,OUT027,Medium,Tier 3,Supermarket Type3,4022.7636,Train,38
8,FDH17,16.2,Regular,0.016687114,Frozen Foods,96.9726,OUT045,Small,Tier 2,Supermarket Type1,1076.5986,Train,21
9,FDU28,19.2,Regular,0.09444959,Frozen Foods,187.8214,OUT017,Small,Tier 2,Supermarket Type1,4710.535,Train,16
10,FDY07,11.8,Low Fat,0.040626524444444444,Fruits and Vegetables,45.5402,OUT049,Medium,Tier 1,Supermarket Type1,1516.0266,Train,24
11,FDA03,18.5,Regular,0.045463773,Dairy,144.1102,OUT046,Small,Tier 1,Supermarket Type1,2187.153,Train,26
12,FDX32,15.1,Regular,0.1000135,Fruits and Vegetables,145.4786,OUT049,Medium,Tier 1,Supermarket Type1,1589.2646,Train,24
13,FDS46,17.6,Regular,0.047257328,Snack Foods,119.6782,OUT046,Small,Tier 1,Supermarket Type1,2145.2076,Train,26
14,FDF32,16.35,Low Fat,0.0680243,Fruits and Vegetables,196.4426,OUT013,High,Tier 3,Supermarket Type1,1977.426,Train,36
15,FDP49,9.0,Regular,0.069088961,Breakfast,56.3614,OUT046,Small,Tier 1,Supermarket Type1,1547.3192,Train,26
16,NCB42,11.8,Low Fat,0.008596051,Health and Hygiene,115.3492,OUT018,Medium,Tier 3,Supermarket Type2,1621.8888,Train,14
17,FDP49,9.0,Regular,0.069196376,Breakfast,54.3614,OUT049,Medium,Tier 1,Supermarket Type1,718.3982,Train,24
18,DRI11,8.26,Low Fat,0.034237682,Hard Drinks,113.2834,OUT027,Medium,Tier 3,Supermarket Type3,2303.668,Train,38
19,FDU02,13.35,Low Fat,0.10249212,Dairy,230.5352,OUT035,Small,Tier 2,Supermarket Type1,2748.4224,Train,19
20,FDN22,18.85,Regular,0.138190277,Snack Foods,250.8724,OUT013,High,Tier 3,Supermarket Type1,3775.086,Train,36
21,FDW12,8.315,Regular,0.035399923,Baking Goods,144.5444,OUT027,Medium,Tier 3,Supermarket Type3,4064.0432,Train,38
22,NCB30,14.6,Low Fat,0.025698134,Household,196.5084,OUT035,Small,Tier 2,Supermarket Type1,1587.2672,Train,19
23,FDC37,15.5,Low Fat,0.057556998,Baking Goods,107.6938,OUT019,Small,Tier 1,Grocery Store,214.3876,Train,38
24,FDR28,13.85,Regular,0.025896485,Frozen Foods,165.021,OUT046,Small,Tier 1,Supermarket Type1,4078.025,Train,26
25,NCD06,13.0,Low Fat,0.099887103,Household,45.906,OUT017,Small,Tier 2,Supermarket Type1,838.908,Train,16
26,FDV10,7.645,Regular,0.066693437,Snack Foods,42.3112,OUT035,Small,Tier 2,Supermarket Type1,1065.28,Train,19
27,DRJ59,11.65,Low Fat,0.019356132,Hard Drinks,39.1164,OUT013,High,Tier 3,Supermarket Type1,308.9312,Train,36
28,FDE51,5.925,Regular,0.161466534,Dairy,45.5086,OUT010,Small,Tier 3,Grocery Store,178.4344,Train,25
29,FDC14,14.5,Regular,0.072221801,Canned,43.6454,OUT019,Small,Tier 1,Grocery Store,125.8362,Train,38
30,FDV38,19.25,Low Fat,0.170348551,Dairy,55.7956,OUT010,Small,Tier 3,Grocery Store,163.7868,Train,25
31,NCS17,18.6,Low Fat,0.080829372,Health and Hygiene,96.4436,OUT018,Medium,Tier 3,Supermarket Type2,2741.7644,Train,14
32,FDP33,18.7,Low Fat,0.0930934742,Snack Foods,256.6672,OUT018,Medium,Tier 3,Supermarket Type2,3068.0064,Train,14
33,FDO23,17.85,Low Fat,0.14246229066666669,Breads,93.1436,OUT045,Small,Tier 2,Supermarket Type1,2174.5028,Train,21
34,DRH01,17.5,Low Fat,0.097904029,Soft Drinks,174.8738,OUT046,Small,Tier 1,Supermarket Type1,2085.2856,Train,26
```

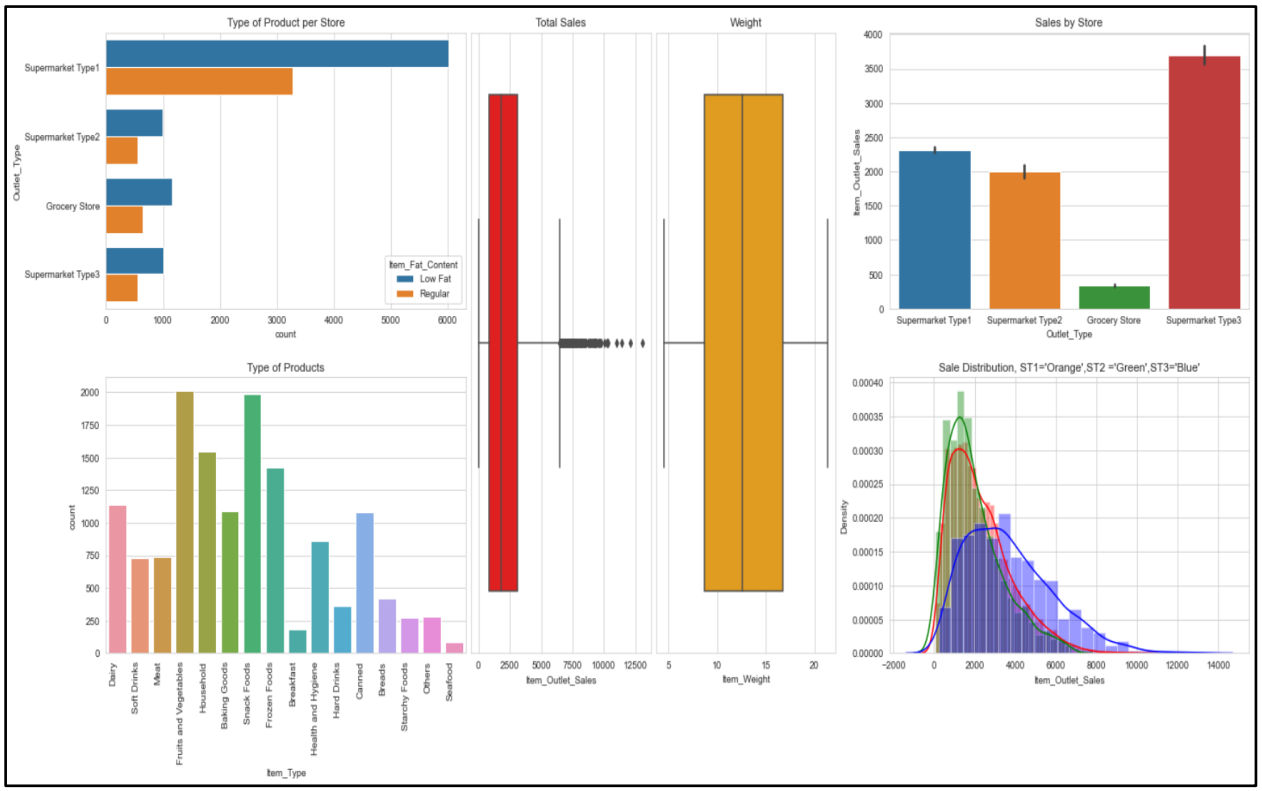


	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source	Outlet_Year
2	0	FDA15	9.3	Low Fat	0.016047301	Dairy	249.8092	OUT049	Medium	Tier 1	Supermarket Type1	3735.138	Train	24
3	1	DRC01	5.92	Regular	0.019278216	Soft Drinks	48.2692	OUT018	Medium	Tier 3	Supermarket Type2	443.4228	Train	14
4	2	FDN15	17.5	Low Fat	0.016760075	Meat	141.618	OUT049	Medium	Tier 1	Supermarket Type1	2097.27	Train	24
5	3	FDX07	19.2	Regular	0.017834338	Fruits and Vegetables	182.095	OUT010	Small	Tier 3	Grocery Store	732.38	Train	25
6	4	NCD19	8.93	Low Fat	0.009779829	Household	53.8614	OUT013	High	Tier 3	Supermarket Type1	994.7052	Train	36
7	5	FDP36	10.395	Regular	0.057058689	Baking Goods	51.4008	OUT018	Medium	Tier 3	Supermarket Type2	556.6088	Train	14
8	6	FDO10	13.65	Regular	0.012741089	Snack Foods	57.6588	OUT013	High	Tier 3	Supermarket Type1	343.5528	Train	36
9	7	FDP10	19	Low Fat	0.127469857	Snack Foods	107.7622	OUT027	Medium	Tier 3	Supermarket Type3	4022.7636	Train	38
10	8	FDH17	16.2	Regular	0.016687114	Frozen Foods	96.9726	OUT045	Small	Tier 2	Supermarket Type1	1076.5986	Train	21
11	9	FDU28	19.2	Regular	0.09444959	Frozen Foods	187.8214	OUT017	Small	Tier 2	Supermarket Type1	4710.535	Train	16
12	10	FDY07	11.8	Low Fat	0.040626524	Fruits and Vegetables	45.5402	OUT049	Medium	Tier 1	Supermarket Type1	1516.0266	Train	24
13	11	FDA03	18.5	Regular	0.045463773	Dairy	144.1102	OUT046	Small	Tier 1	Supermarket Type1	2187.153	Train	26
14	12	FDX32	15.1	Regular	0.1000135	Fruits and Vegetables	145.4786	OUT049	Medium	Tier 1	Supermarket Type1	1589.2646	Train	24
15	13	FDS46	17.6	Regular	0.047257328	Snack Foods	119.6782	OUT046	Small	Tier 1	Supermarket Type1	2145.2076	Train	26
16	14	FDF32	16.35	Low Fat	0.0680243	Fruits and Vegetables	196.4426	OUT013	High	Tier 3	Supermarket Type1	1977.426	Train	36
17	15	FDP49	9	Regular	0.069088961	Breakfast	56.3614	OUT046	Small	Tier 1	Supermarket Type1	1547.3192	Train	26
18	16	NCB42	11.8	Low Fat	0.008596051	Health and Hygiene	115.3492	OUT018	Medium	Tier 3	Supermarket Type2	1621.8888	Train	14
19	17	FDP49	9	Regular	0.069196376	Breakfast	54.3614	OUT049	Medium	Tier 1	Supermarket Type1	718.3982	Train	24
20	18	DRI11	8.26	Low Fat	0.034237682	Hard Drinks	113.2834	OUT027	Medium	Tier 3	Supermarket Type3	2303.668	Train	38
21	19	FDU02	13.35	Low Fat	0.10249212	Dairy	230.5352	OUT035	Small	Tier 2	Supermarket Type1	2748.4224	Train	19
22	20	FDN22	18.85	Regular	0.138190277	Snack Foods	250.8724	OUT013	High	Tier 3	Supermarket Type1	3775.086	Train	36
23	21	FDW12	8.315	Regular	0.035399923	Baking Goods	144.5444	OUT027	Medium	Tier 3	Supermarket Type3	4064.0432	Train	38



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
14183	14181	FDL45	15.6	Low Fat	0.037656474	Snack Foods	123.4704	OUT013	High	Tier 3	Supermarket Type1		Test	36
14184	14182	DRK37	5	Low Fat	0.04407309	Soft Drinks	188.853	OUT049	Medium	Tier 1	Supermarket Type1		Test	24
14185	14183	FDK22	9.8	Low Fat	0.026064791	Snack Foods	215.385	OUT013	High	Tier 3	Supermarket Type1		Test	36
14186	14184	DRG37	16.2	Low Fat	0.020225187	Soft Drinks	155.7972	OUT027	Medium	Tier 3	Supermarket Type3		Test	38
14187	14185	FDK22	9.8	Low Fat	0.026234055	Snack Foods	214.385	OUT017	Small	Tier 2	Supermarket Type1		Test	16
14188	14186	DRH36	16.2	Low Fat	0.033516036	Soft Drinks	72.8696	OUT018	Medium	Tier 3	Supermarket Type2		Test	14
14189	14187	DRC36	13	Regular	0.045168123	Soft Drinks	173.4054	OUT018	Medium	Tier 3	Supermarket Type2		Test	14
14190	14188	DRE03	19.6	Low Fat	0.024109582	Dairy	46.0718	OUT027	Medium	Tier 3	Supermarket Type3		Test	38
14191	14189	FDF34	9.3	Regular	0.014019393	Snack Foods	196.9084	OUT046	Small	Tier 1	Supermarket Type1		Test	26
14192	14190	FDZ22	9.395	Low Fat	0.045269896	Snack Foods	82.125	OUT046	Small	Tier 1	Supermarket Type1		Test	26
14193	14191	FDC44	15.6	Low Fat	0.288891801	Fruits and Vegetables	115.1518	OUT010	Small	Tier 3	Grocery Store		Test	25
14194	14192	FDN31	11.5	Low Fat	0.072528603	Fruits and Vegetables	188.053	OUT027	Medium	Tier 3	Supermarket Type3		Test	38
14195	14193	FDO03	10.395	Regular	0.037091602	Meat	229.4352	OUT017	Small	Tier 2	Supermarket Type1		Test	16
14196	14194	FDA01	15	Regular	0.054462797	Canned	59.5904	OUT049	Medium	Tier 1	Supermarket Type1		Test	24
14197	14195	NCH42	6.86	Low Fat	0.036594126	Household	231.101	OUT049	Medium	Tier 1	Supermarket Type1		Test	24
14198	14196	FDF46	7.07	Low Fat	0.094052753	Snack Foods	116.0834	OUT018	Medium	Tier 3	Supermarket Type2		Test	14
14199	14197	DRL35	15.7	Low Fat	0.03070363	Hard Drinks	43.277	OUT046	Small	Tier 1	Supermarket Type1		Test	26
14200	14198	FDW46	13	Regular	0.070410959	Snack Foods	63.4484	OUT049	Medium	Tier 1	Supermarket Type1		Test	24
14201	14199	FDB58	10.5	Regular	0.013496466	Snack Foods	141.3154	OUT046	Small	Tier 1	Supermarket Type1		Test	26
14202	14200	FDD47	7.6	Regular	0.142990896	Starchy Foods	169.1448	OUT018	Medium	Tier 3	Supermarket Type2		Test	14
14203	14201	NCO17	10	Low Fat	0.073528561	Health and Hygiene	118.744	OUT045	Small	Tier 2	Supermarket Type1		Test	21
14204	14202	FDJ26	15.3	Regular	0.088380075	Canned	214.6218	OUT017	Small	Tier 2	Supermarket Type1		Test	16
14205	14203	FDU37	9.5	Regular	0.104720151	Canned	79.796	OUT045	Small	Tier 2	Supermarket Type1		Test	21

In [46]:
```
# --- Creating Dashboard of completly processed dataset ---
fig = plt.figure(constrained_layout=True, figsize=(20,10))
grid = gridspec.GridSpec(ncols=6, nrows=2, figure=fig)

# --- bar plot Horizontal ---
ax1 = fig.add_subplot(grid[0, :2])
ax1.set_title('Type of Product per Store')
sns.countplot(y='Outlet_Type',hue = 'Item_Fat_Content',data = dataset, ax=ax1) #Outlet Type

# --- bar plot Vertical ---
ax2 = fig.add_subplot(grid[1, :2])
ax2.set_title('Type of Products')
bar = sns.countplot(x='Item_Type', data = dataset, ax = ax2)
bar.set_xticklabels(bar.get_xticklabels(),  rotation=90, horizontalalignment='right') #Type of Products

# --- box plot Sales ---
ax3 = fig.add_subplot(grid[:, 2])
ax3.set_title('Total Sales')
sns.boxplot(train.loc[:,'Item_Outlet_Sales'], orient='v', ax = ax3, color = 'Red') #Total Sales

# --- box plot Monthly payment ---
ax4 = fig.add_subplot(grid[:,3])
ax4.set_title("Weight")
sns.boxplot(dataset['Item_Weight'], orient='v' ,ax=ax4, color = 'Orange')


# --- Displot Distribution ---
ax5 = fig.add_subplot(grid[0, 4:6])
ax5.set_title("Sales by Store")
sns.barplot(x='Outlet_Type', y='Item_Outlet_Sales',data=dataset,ax=ax5)


# --- Displot Distribution ---
ax6 = fig.add_subplot(grid[1, 4:6])
ax6.set_title("Sale Distribution, ST1='Orange',ST2 ='Green',ST3='Blue'")
SuperT1 = dataset[dataset['Outlet_Type']=='Supermarket Type1']
SuperT2 = dataset[dataset['Outlet_Type']=='Supermarket Type2']
SuperT3 = dataset[dataset['Outlet_Type']=='Supermarket Type3']
Grocery = dataset[dataset['Outlet_Type']=='Grocery Store']

sns.distplot(SuperT1['Item_Outlet_Sales'], color = 'Red', ax=ax6)
sns.distplot(SuperT2['Item_Outlet_Sales'], color = 'Green', ax=ax6)
sns.distplot(SuperT3['Item_Outlet_Sales'], color = 'Blue', ax=ax6)

plt.show()
```



Converting Categorical Variables into Numerical Values:

```
In [47]: # --- Turning all categorical variables into numerical values can be done by mapping each categorical value with respective FREQU
cat_var = ['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Outlet_Type','Item_Type',
           'Item_Identifier','Outlet_Identifier']
for i in cat_var:
    p  = dataset[i].value_counts().to_dict()
    dataset[i] = dataset[i].map(p)
```

```
In [48]: # --- Reading Combined Dataset after data processing ---
dataset.head(10).style.background_gradient(cmap='BuPu').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source	Outlet_Year
0	9	9.300000	9185	0.016047	1136	249.809200	1550	4655	3980	9294	3735.138000	Train	24
1	9	5.920000	5019	0.019278	726	48.269200	1546	4655	5583	1546	443.422800	Train	14
2	10	17.500000	9185	0.016760	736	141.618000	1550	4655	3980	9294	2097.270000	Train	24
3	9	19.200000	5019	0.017834	2013	182.095000	925	7996	5583	1805	732.380000	Train	25
4	9	8.930000	9185	0.009780	1548	53.861400	1553	1553	5583	9294	994.705200	Train	36
5	8	10.395000	5019	0.057059	1086	51.400800	1546	4655	5583	1546	556.608800	Train	14
6	10	13.650000	5019	0.012741	1989	57.658800	1553	1553	5583	9294	343.552800	Train	36
7	10	19.000000	9185	0.127470	1989	107.762200	1559	4655	5583	1559	4022.763600	Train	38
8	9	16.200000	5019	0.016687	1426	96.972600	1548	7996	4641	9294	1076.598600	Train	21
9	10	19.200000	5019	0.094450	1426	187.821400	1543	7996	4641	9294	4710.535000	Train	16

```
In [49]: # --- Reading Combined Dataset after data processing ---
dataset.tail(10).style.background_gradient(cmap='BuPu').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source	Outlet_Year
14194	9	15.000000	5019	0.054463	1084	59.590400	1550	4655	3980	9294	nan	Test	24
14195	10	6.860000	9185	0.036594	1548	231.101000	1550	4655	3980	9294	nan	Test	24
14196	9	7.070000	9185	0.094053	1989	116.083400	1546	4655	5583	1546	nan	Test	14
14197	9	15.700000	9185	0.030704	362	43.277000	1550	7996	3980	9294	nan	Test	26
14198	9	13.000000	5019	0.070411	1989	63.448400	1550	4655	3980	9294	nan	Test	24
14199	10	10.500000	5019	0.013496	1989	141.315400	1550	7996	3980	9294	nan	Test	26
14200	10	7.600000	5019	0.142991	269	169.144800	1546	4655	5583	1546	nan	Test	14
14201	9	10.000000	9185	0.073529	858	118.744000	1548	7996	4641	9294	nan	Test	21
14202	10	15.300000	5019	0.088380	1084	214.621800	1543	7996	4641	9294	nan	Test	16
14203	9	9.500000	5019	0.104720	1084	79.796000	1548	7996	4641	9294	nan	Test	21

Converting And Splitting The Dataset

```
In [50]: # --- Dividing dataset into testing and training dataset ---
train = dataset.loc[dataset['source']=="Train"]
test = dataset.loc[dataset['source']=="Test"]

# --- Drop unnecessary columns ---
test.drop(['source'],axis=1,inplace=True)
train.drop(['source'],axis=1,inplace=True)
```

```
In [51]: # --- Reading the divided training dataset ---
train.head(10).style.background_gradient(cmap='YlOrRd').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	Outlet_Year
0	9	9.300000	9185	0.016047	1136	249.809200	1550	4655	3980	9294	3735.138000	24
1	9	5.920000	5019	0.019278	726	48.269200	1546	4655	5583	1546	443.422800	14
2	10	17.500000	9185	0.016760	736	141.618000	1550	4655	3980	9294	2097.270000	24
3	9	19.200000	5019	0.017834	2013	182.095000	925	7996	5583	1805	732.380000	25
4	9	8.930000	9185	0.009780	1548	53.861400	1553	1553	5583	9294	994.705200	36
5	8	10.395000	5019	0.057059	1086	51.400800	1546	4655	5583	1546	556.608800	14
6	10	13.650000	5019	0.012741	1989	57.658800	1553	1553	5583	9294	343.552800	36
7	10	19.000000	9185	0.127470	1989	107.762200	1559	4655	5583	1559	4022.763600	38
8	9	16.200000	5019	0.016687	1426	96.972600	1548	7996	4641	9294	1076.598600	21
9	10	19.200000	5019	0.094450	1426	187.821400	1543	7996	4641	9294	4710.535000	16

In [52]:
```
# --- Reading the divided testing dataset ---
test.head(10).style.background_gradient(cmap='summer').set_properties(**{'border': '1px solid black'})
```

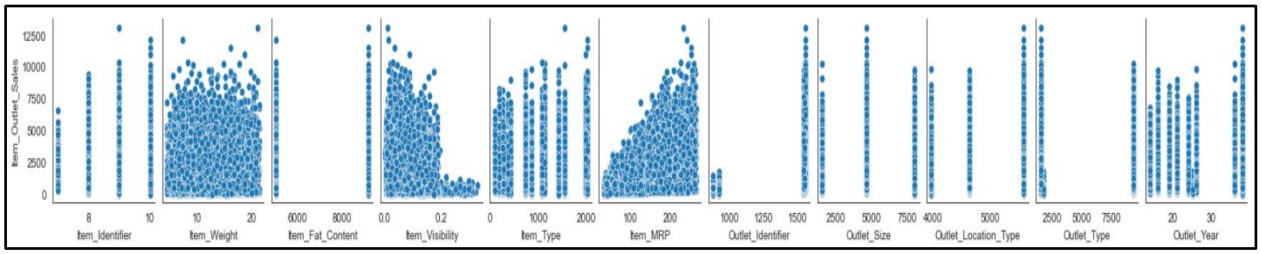
	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	Outlet_Year
8523	9	20.750000	9185	0.007565	1989	107.862200	1550	4655	3980	9294	nan	24
8524	9	8.300000	5019	0.038428	1136	87.319800	1543	7996	4641	9294	nan	16
8525	9	14.600000	9185	0.099575	280	241.753800	925	7996	5583	1805	nan	25
8526	9	7.315000	9185	0.015388	1989	155.034000	1543	7996	4641	9294	nan	16
8527	9	13.600000	5019	0.118599	1136	234.230000	1559	4655	5583	1559	nan	38
8528	10	9.800000	5019	0.063817	2013	117.149200	1550	7996	3980	9294	nan	26
8529	9	19.350000	5019	0.082602	1086	50.103400	1546	4655	5583	1546	nan	14
8530	9	9.195000	9185	0.015782	1086	81.059200	1559	4655	5583	1559	nan	38
8531	8	6.305000	5019	0.123365	1989	95.743600	1548	7996	4641	9294	nan	21
8532	9	5.985000	9185	0.005698	1086	186.892400	1543	7996	4641	9294	nan	16

Visualizing Correlation:

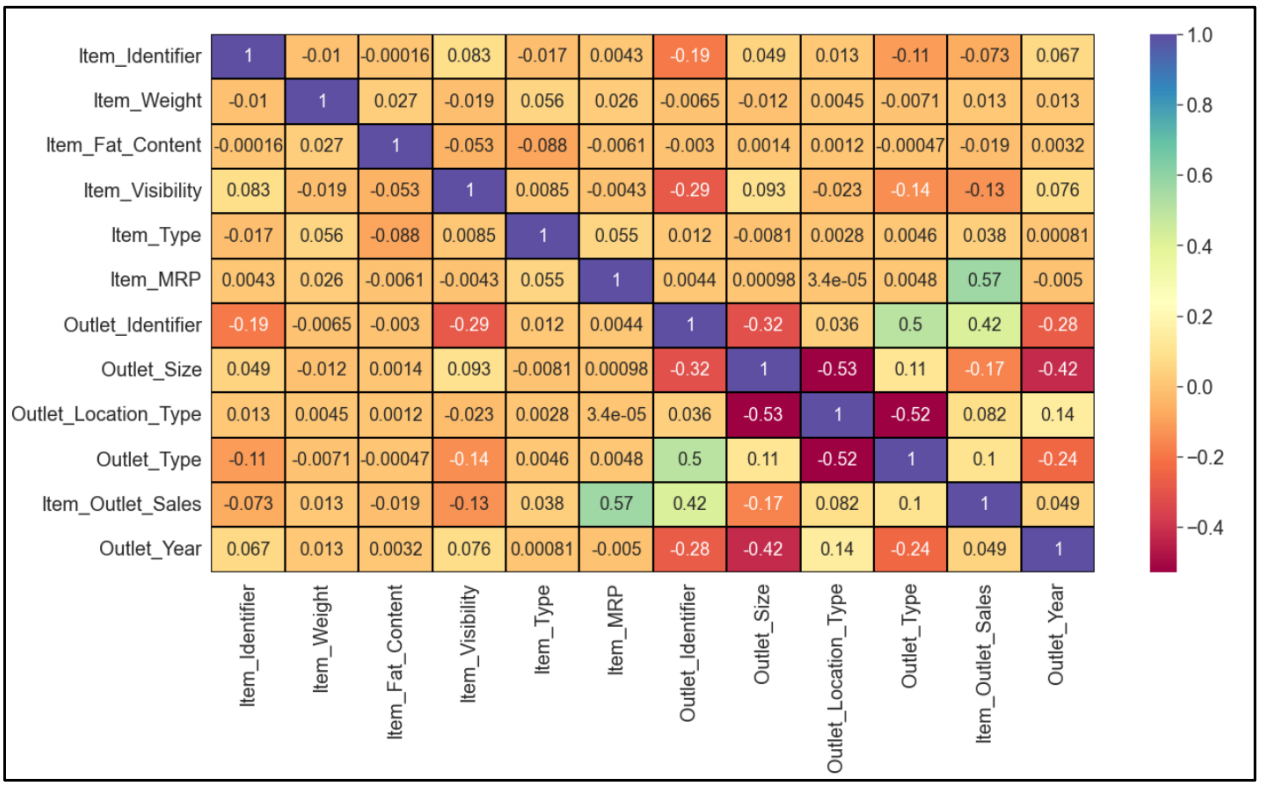
In [53]:
```
# --- Finding corelation between the columns of training dataset ---
corr_matrix=train.corr()
corr_matrix['Item_Outlet_Sales']
```

Out[53]:
```
Item_Identifier        -0.073375
Item_Weight             0.013261
Item_Fat_Content       -0.018719
Item_Visibility        -0.128453
Item_Type               0.038123
Item_MRP                0.567574
Outlet_Identifier       0.415482
Outlet_Size            -0.165968
Outlet_Location_Type    0.082256
Outlet_Type             0.100742
Item_Outlet_Sales       1.000000
Outlet_Year             0.049135
Name: Item_Outlet_Sales, dtype: float64
```

In [54]:
```
# --- Ploting multiple bivariate pairwise columns of training dataset ---
sns.set_style('white')
p=sns.pairplot(data=train,y_vars=['Item_Outlet_Sales'],x_vars=['Item_Identifier','Item_Weight', 'Item_Fat_Content',
                                                               'Item_Visibility', 'Item_Type', 'Item_MRP','Outlet_Identifier',
                                                               'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type',
                                                               'Outlet_Year'])

p.fig.set_figwidth(20)
```



```
In [55]: # --- Ploting heat map based on Correlation Values ---
         plt.figure(figsize = (20,10))
         sns.set(font_scale=1.8)
         sns.heatmap(corr_matrix, cmap = "Spectral", annot = True,annot_kws={"size": 18},linewidths=2, linecolor="black")
```



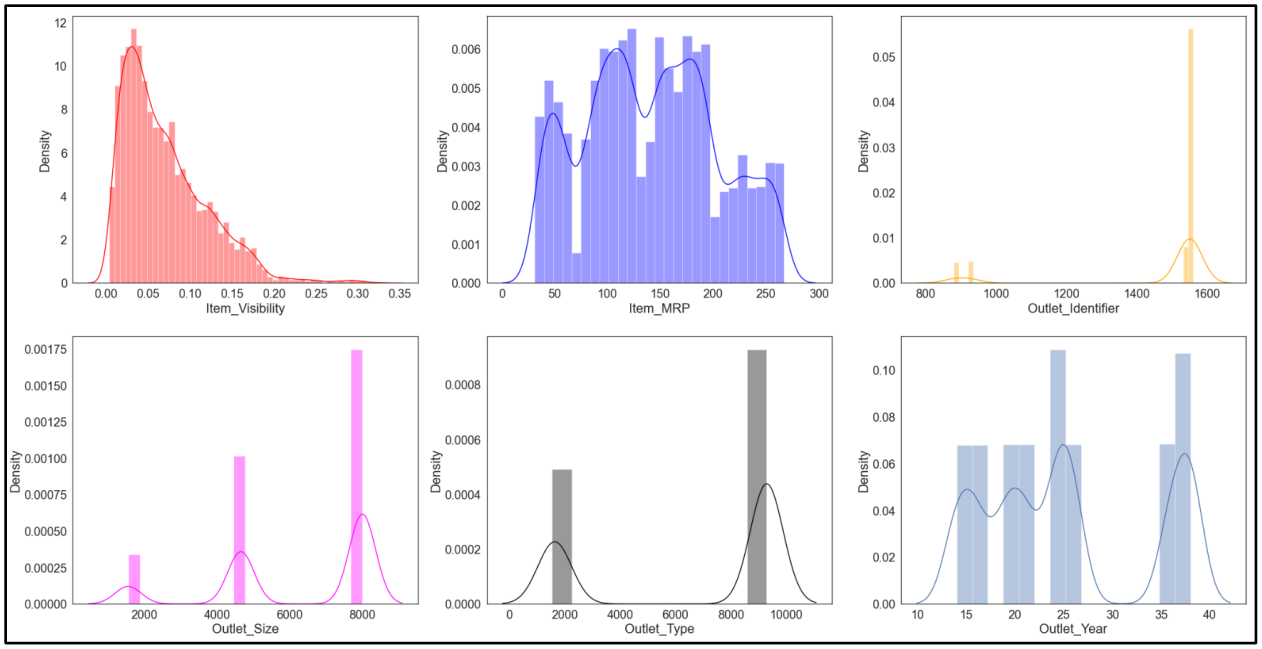
Model Building

Removing less correlated values:

```
In [56]: # --- Removing Unnecessary and Less Corelated data from the dataset ---
         train= train.drop(['Item_Identifier','Item_Weight', 'Item_Fat_Content', 'Item_Type', 'Outlet_Location_Type'],axis=1)
```

```
In [57]: # --- Finding Skewness ---
         train.skew()

Out[57]: Item_Visibility      1.224502
         Item_MRP             0.127202
         Outlet_Identifier   -2.244440
         Outlet_Size         -0.803252
         Outlet_Type         -0.650419
         Item_Outlet_Sales    1.177531
         Outlet_Year          0.396641
         dtype: float64
```

```
In [58]: sns.set_style('white')
         # --- Before Scaling ---
         fig, ax = plt.subplots(2,3,figsize = (35,18))
         sns.distplot(train["Item_Visibility"], kde =True, ax=ax[0,0], color = "red")
         sns.distplot(train["Item_MRP"], kde =True, ax=ax[0,1], color = "blue")
         sns.distplot(train["Outlet_Identifier"], kde =True, ax=ax[0,2], color = "orange")
         sns.distplot(train["Outlet_Size"], kde =True, ax=ax[1,0], color = "magenta")
         sns.distplot(train["Outlet_Type"], kde =True, ax=ax[1,1], color = "black")
         sns.distplot(train["Outlet_Year"], kde =True, ax=ax[1,2])
```



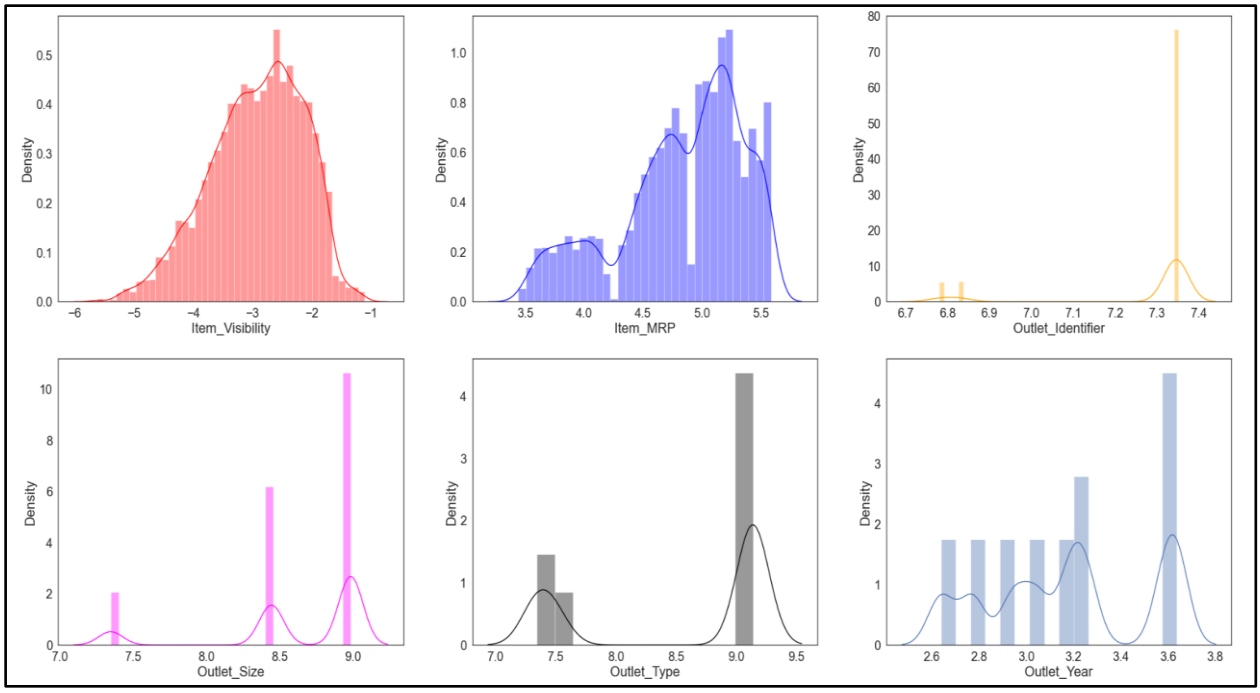
Normalization:

In [59]:
```
# --- Normalizing Dataset Values ---
for i in train.columns:
    train[i] =np.log(train[i])
```

In [60]:
```
# --- Reading Dataset ---
train.head(10).style.background_gradient(cmap='YlOrRd').set_properties(**{'border': '1px solid black'})
```

	Item_Visibility	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Type	Item_Outlet_Sales	Outlet_Year
0	-4.132215	5.520697	7.346010	8.445697	9.137124	8.225540	3.178054
1	-3.948780	3.876794	7.343426	8.445697	7.343426	6.094524	2.639057
2	-4.088756	4.953133	7.346010	8.445697	9.137124	7.648392	3.178054
3	-4.026630	5.204529	6.829794	8.986697	7.498316	6.596300	3.218876
4	-4.627433	3.986414	7.347944	7.347944	9.137124	6.902446	3.583519
5	-2.863675	3.939654	7.343426	8.445697	7.343426	6.321863	2.639057
6	-4.362923	4.054543	7.347944	7.347944	9.137124	5.839341	3.583519
7	-2.059875	4.679927	7.351800	8.445697	7.351800	8.299724	3.637586
8	-4.093118	4.574428	7.344719	8.986697	9.137124	6.981562	3.044522
9	-2.359689	5.235492	7.341484	8.986697	9.137124	8.457557	2.772589

In [61]:
```
# --- After Scaling the variables ---
fig, ax = plt.subplots(2,3,figsize = (35,18))
sns.distplot(train["Item_Visibility"], kde =True, ax=ax[0,0], color = "red")
sns.distplot(train["Item_MRP"], kde =True, ax=ax[0,1], color = "blue")
sns.distplot(train["Outlet_Identifier"], kde =True, ax=ax[0,2], color = "orange")
sns.distplot(train["Outlet_Size"], kde =True, ax=ax[1,0], color = "magenta")
sns.distplot(train["Outlet_Type"], kde =True, ax=ax[1,1], color = "black")
sns.distplot(train["Outlet_Year"], kde =True, ax=ax[1,2])
```



Identifying the best fitting model:

```
In [62]: # --- Defining Target and Predictor Variable ---
y=train["Item_Outlet_Sales"]
x= train.drop(["Item_Outlet_Sales"],axis=1)
x.head(10).style.background_gradient(cmap='YlOrRd').set_properties(**{'border': '1px solid black'})
```

	Item_Visibility	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Type	Outlet_Year
0	-4.132215	5.520697	7.346010	8.445697	9.137124	3.178054
1	-3.948780	3.876794	7.343426	8.445697	7.343426	2.639057
2	-4.088756	4.953133	7.346010	8.445697	9.137124	3.178054
3	-4.026630	5.204529	6.829794	8.986697	7.498316	3.218876
4	-4.627433	3.986414	7.347944	7.347944	9.137124	3.583519
5	-2.863675	3.939654	7.343426	8.445697	7.343426	2.639057
6	-4.362923	4.054543	7.347944	7.347944	9.137124	3.583519
7	-2.059875	4.679927	7.351800	8.445697	7.351800	3.637586
8	-4.093118	4.574428	7.344719	8.986697	9.137124	3.044522
9	-2.359689	5.235492	7.341484	8.986697	9.137124	2.772589

```
In [63]: from matplotlib import pyplot as plt
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

```
In [64]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=0)
```

Fitting linear regression model:

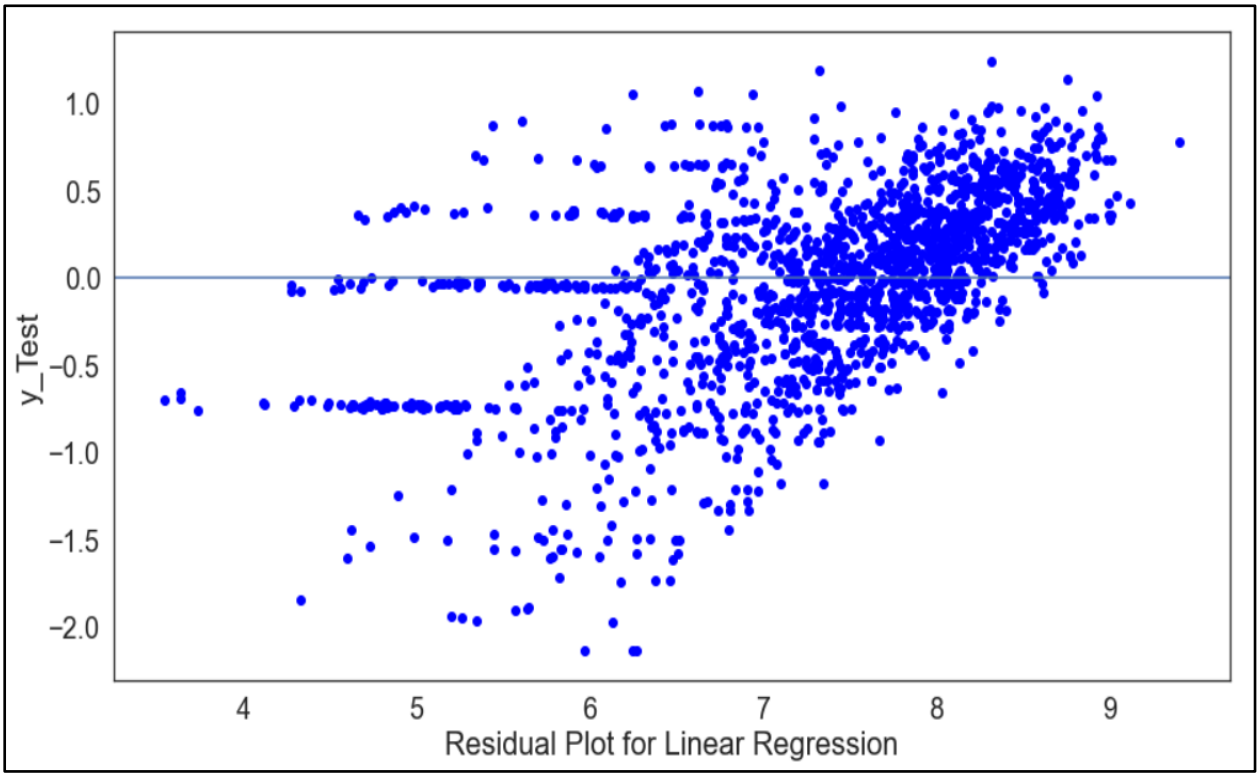
```
In [65]: # --- Fitting Linear Regression ---
         reg = LinearRegression()
         reg = reg.fit(x_train,y_train)
```

```
In [66]: reg.coef_

Out[66]: array([-1.35516130e-03,  1.01609492e+00,  4.34936806e+00,  2.06613056e-01,
                -1.10626996e-01,  4.94382445e-01])
```

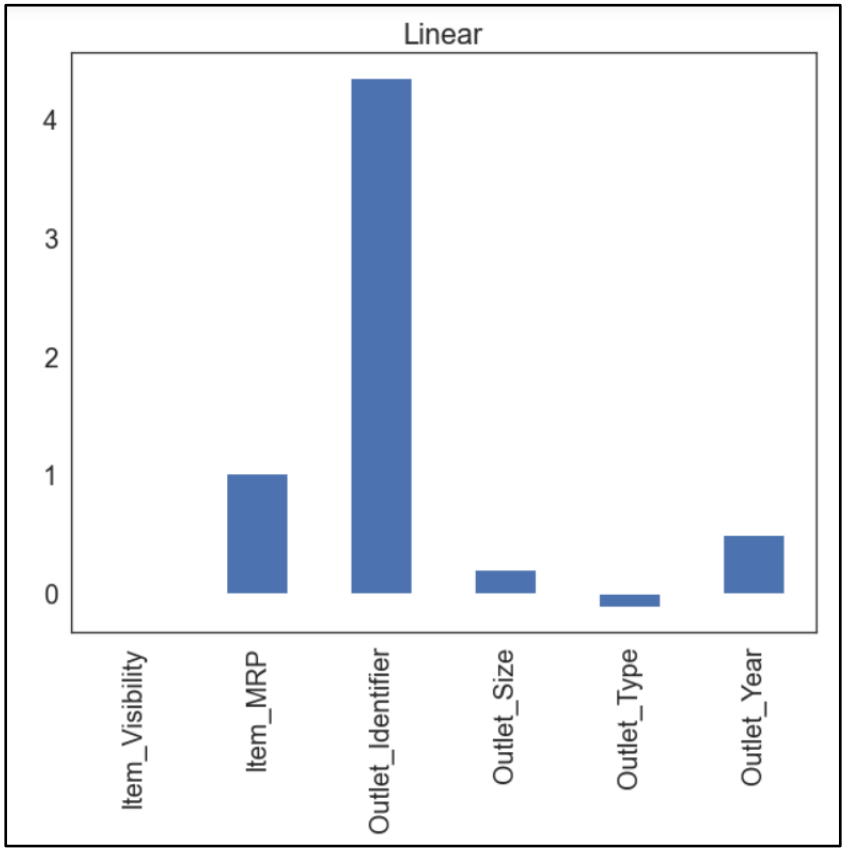
```
In [67]: reg.intercept_

Out[67]: -31.665850837981985
```

```
In [68]: # --- Prediction of Linear Regression ---
         y_pred = reg.predict(x_test)

         # --- RMSE ---
         rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
         print('Root Mean Squared Error:',rmse )

         # --- Accuracy Score Check ---
         r2_score = r2_score(y_test, y_pred)
         print('R2_Score:',r2_score*100,"%" )

         Root Mean Squared Error: 0.537468458180603
         R2_Score: 72.96248381320213 %
```

```
In [69]: # --- RESIDUE VALUE OF LINEAR REGRESSION ---
         residue_lreg = y_test - y_pred

         # --- Plotting Residual Plot ---
         plt.figure(figsize=(15, 8))
         plt.scatter(y_test,residue_lreg, c = "blue")
         plt.xlabel("Residual Plot for Linear Regression")
         plt.ylabel("y_Test")
         plt.axhline(y = 0)
```



```
In [70]: # --- Linear Regression ---
         plt.figure(figsize=(10, 8))
         Lreg_coef = pd.Series(reg.coef_,index =x.columns)
         Lreg_coef.plot(kind="bar", title= "Linear")
```



Fitting lasso regression model:

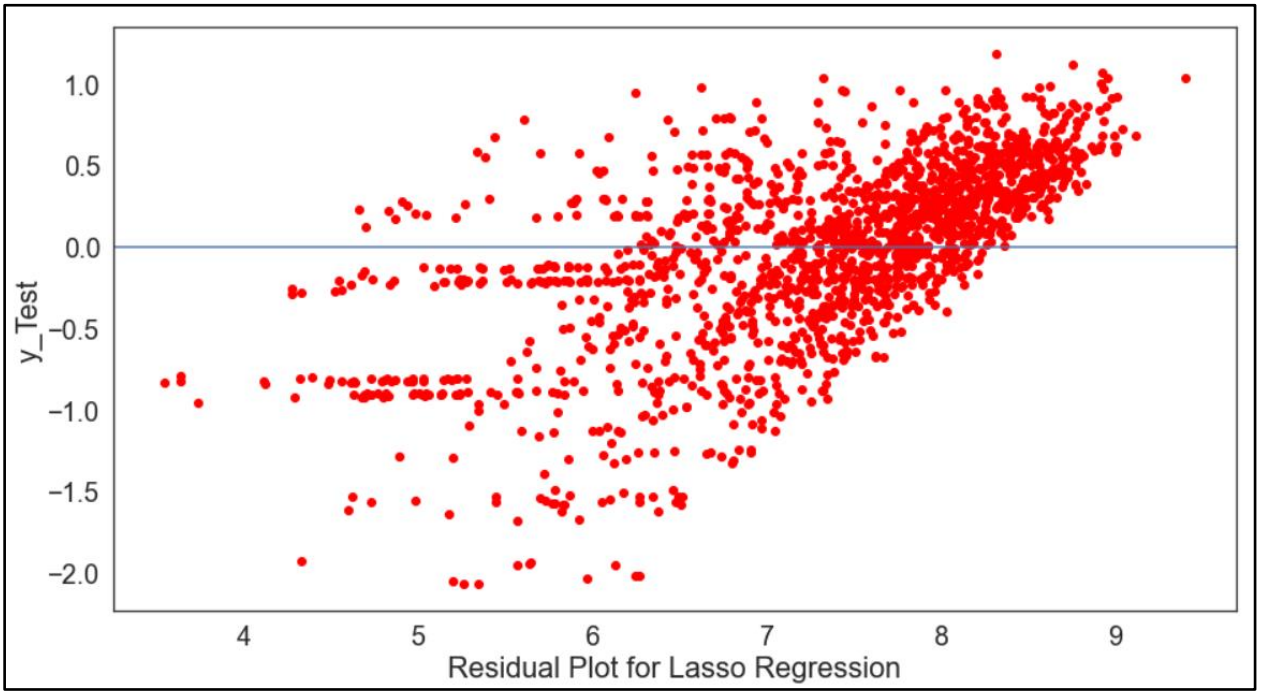
```
In [71]: from sklearn.linear_model import Lasso, Ridge

         # --- Fitting Lasso Regression ---
         ls = Lasso(alpha = 0.009)
         ls = ls.fit(x_train, y_train)
```

```
In [72]: ls.coef_

Out[72]: array([-0.        ,  0.98490925,  3.63249809,  0.02916615, -0.03949171,
                0.21741931])
```

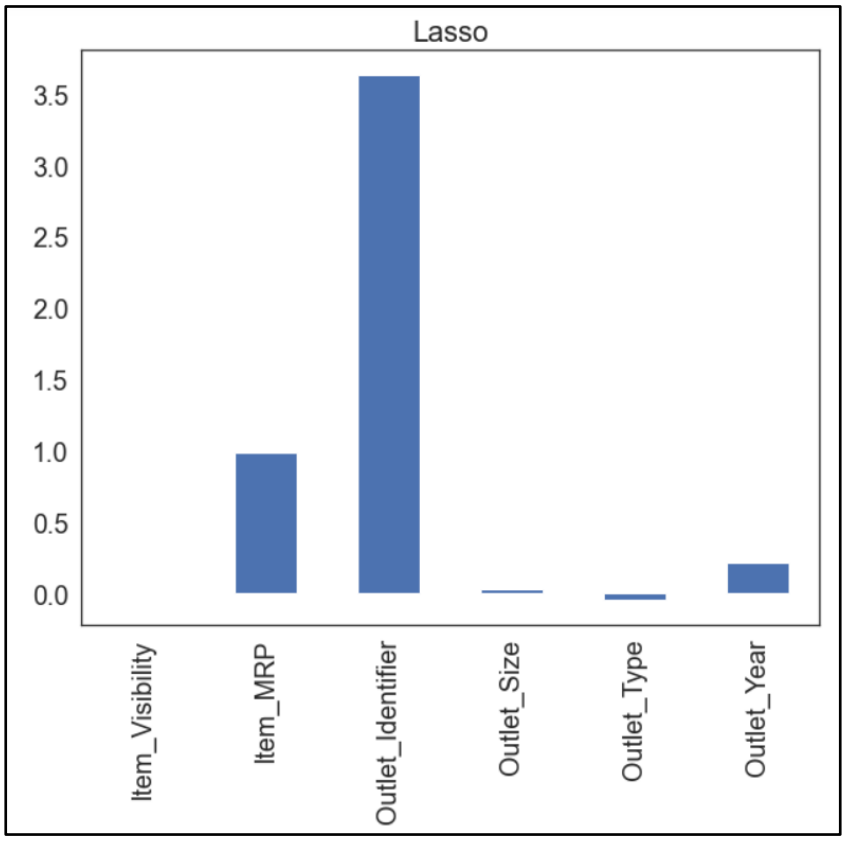
```
In [73]: ls.intercept_

Out[73]: -24.49194908310222
```

```
In [74]: # --- Prediction of Lasso Regression ---
         ls_pred = ls.predict(x_test)

         # --- RSME ---
         rmse_LS = np.sqrt(metrics.mean_squared_error(y_test, ls_pred))
         print('Root Mean Squared Error:',rmse_LS )

         # --- Accuracy Score Check ---
         from sklearn.metrics import r2_score
         r2_score_LS = r2_score(y_test, ls_pred)
         print('R2_Score:',r2_score_LS*100,"%" )

         Root Mean Squared Error: 0.5524229989252929
         R2_Score: 71.43696644494459 %
```

In [75]:
```
# --- RESIDUE VALUE OF LASSO REGRESSION ---
residue_lasso = y_test - ls_pred

# --- Plotting Residual Plot ---
plt.figure(figsize=(15, 8))
plt.scatter(y_test,residue_lasso, c = "red")
plt.xlabel("Residual Plot for Lasso Regression")
plt.ylabel("y_Test")
plt.axhline(y = 0)
```



In [76]:
```
# --- Lasso Regression ---
plt.figure(figsize=(10, 8))
lasso_coef = pd.Series(ls.coef_,index =x.columns)
lasso_coef.plot(kind="bar", title= "Lasso")
```



Fitting ridge regression model:

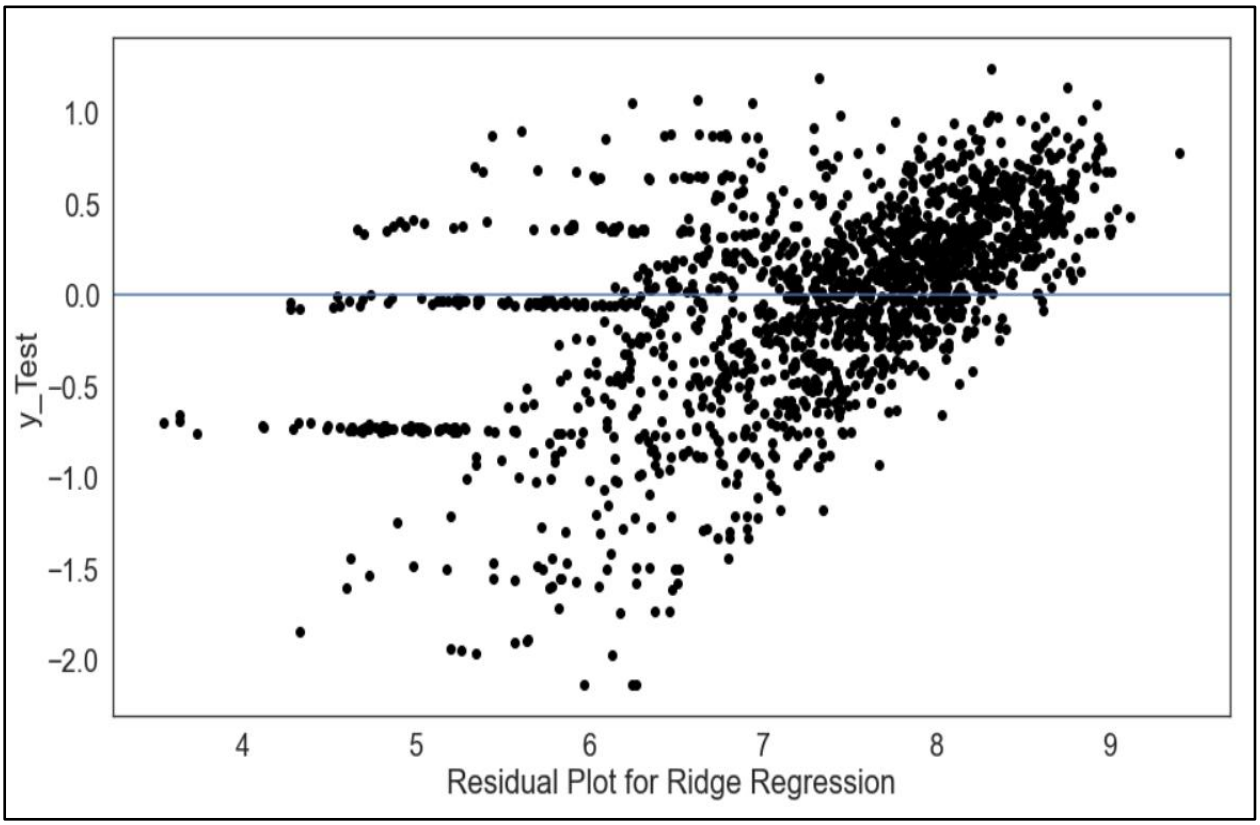
In [77]:
```
# --- Fitting Ridge Regression ---
rr = Ridge(alpha = 0.009)
rr.fit(x_train, y_train)
```

Out[77]:
```
Ridge(alpha=0.009)
```

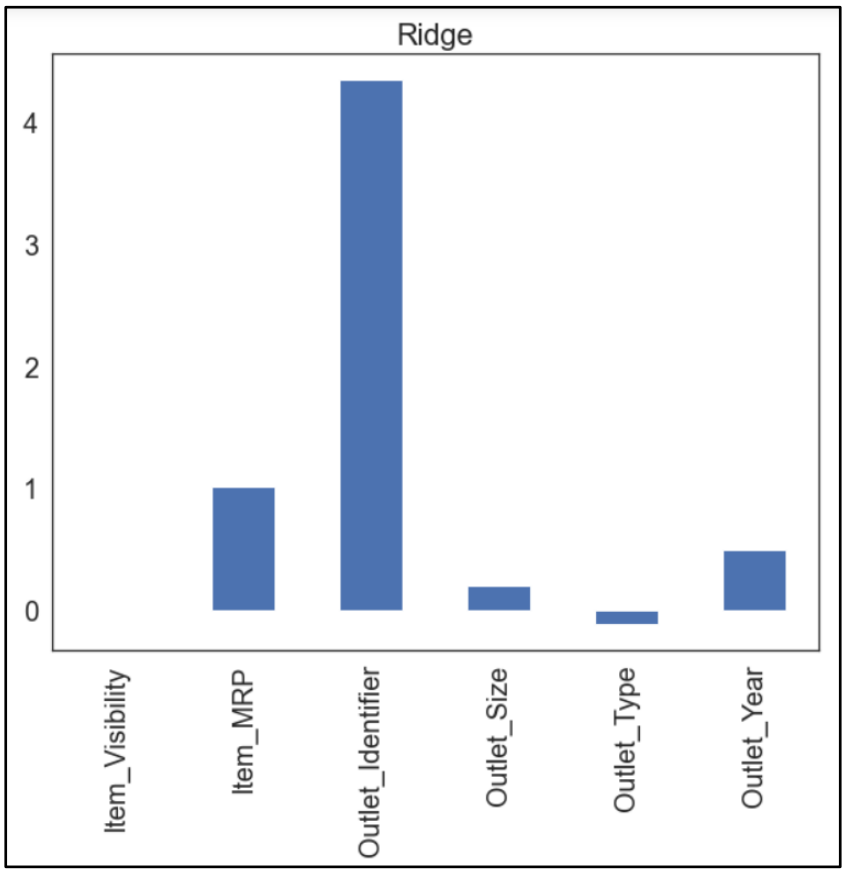
In [78]:
```
# --- Prediction of Ridge regression ---
rr_pred = rr.predict(x_test)

# --- RMSE ---
rmse_ridge = np.sqrt(metrics.mean_squared_error(y_test, rr_pred))
print('Root Mean Squared Error:',rmse_ridge)

# --- Accuracy score check ---
r2_score_RR = r2_score(y_test, y_pred)
print('R2_Score:',r2_score_RR*100,"%" )
```

```
Root Mean Squared Error: 0.5374699611092182
R2_Score: 72.96248381320213 %
```

In [79]:
```
# --- RESIDUE VALUE OF RIDGE REGRESSION ---
residue_rr = y_test-rr_pred

# --- Plotting Residual Plot ---
plt.figure(figsize=(15, 8))
plt.scatter(y_test,residue_rr, c = "black")
plt.xlabel("Residual Plot for Ridge Regression")
plt.ylabel("y_Test")
plt.axhline(y = 0)
```



In [80]:
```
# --- Ridge Regression ---
plt.figure(figsize=(10, 8))
ridge_coef = pd.Series(rr.coef_,index =x.columns)
ridge_coef.plot(kind="bar", title= "Ridge",position=0.5)
```



In [81]:
```
df1 = pd.DataFrame(columns=["Linear Regression", "Ridge Regression","Lasso Regression"])
for i in range(len(rr.coef_)):
    df1=df1.append({"Linear Regression":reg.coef_[i],"Ridge Regression":rr.coef_[i], "Lasso Regression":ls.coef_[i]},
                   ignore_index = True)
df1
```

	Linear Regression	Ridge Regression	Lasso Regression
0	-0.001355	-0.001363	-0.0
1	1.016095	1.016091	0.984909
2	4.349368	4.349037	3.632498
3	0.206613	0.20656	0.029166
4	-0.110627	-0.110598	-0.039492
5	0.494382	0.494304	0.217419

Deciding Best-Fitting Model

RMSE:

Linear Regression: 0.537468458180603

Lasso Regression: 0.5524229989252929

Ridge Regression: 0.5374699611092182

R2_score:

Linear Regression: 72.96248381320213 %

Lasso Regression: 71.43696644494459 %

Ridge Regression: 72.96248381320213 %

A model is said to be the best fit if the R2_score is closer to 100% and has a better RMSE score.

1. Ridge and Linear Regression are more accurate than Lasso Regression.
2. R2_Score values for Ridge Regression and Linear Regression are identical. However, the RMSE value of the Linear Regression model is higher.

As a result, the best-fit model for the provided dataset is: LINEAR REGRESSION

Predicting Data

```
In [82]: test.head(10).style.background_gradient(cmap='summer').set_properties(**{'border': '1px solid black'})
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	Outlet_Year
8523	9	20.750000	9185	0.007565	1989	107.862200	1550	4655	3980	9294	nan	24
8524	9	8.300000	5019	0.038428	1136	87.319800	1543	7996	4641	9294	nan	16
8525	9	14.600000	9185	0.099575	280	241.753800	925	7996	5583	1805	nan	25
8526	9	7.315000	9185	0.015388	1989	155.034000	1543	7996	4641	9294	nan	16
8527	9	13.600000	5019	0.118599	1136	234.230000	1559	4655	5583	1559	nan	38
8528	10	9.800000	5019	0.063817	2013	117.149200	1550	7996	3980	9294	nan	26
8529	9	19.350000	5019	0.082602	1086	50.103400	1546	4655	5583	1546	nan	14
8530	9	9.195000	9185	0.015782	1086	81.059200	1559	4655	5583	1559	nan	38
8531	8	6.305000	5019	0.123365	1989	95.743600	1548	7996	4641	9294	nan	21
8532	9	5.985000	9185	0.005698	1086	186.892400	1543	7996	4641	9294	nan	16

```
In [83]: test= test.drop(["Item_Outlet_Sales"],axis=1)
         test_ = test.drop(['Item_Identifier','Item_Weight', 'Item_Fat_Content', 'Item_Type', 'Outlet_Location_Type'],axis=1)
```

```
In [84]: test_.head(10).style.set_properties(**{'border': '1px solid black'})
```

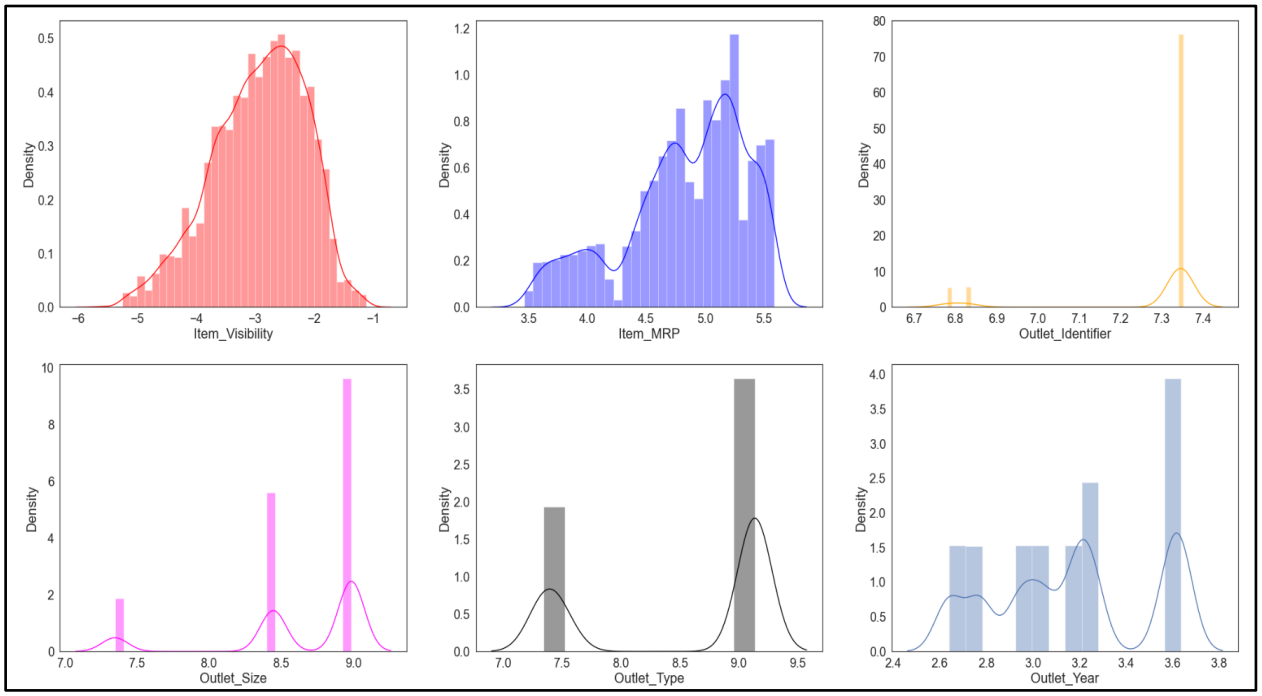
	Item_Visibility	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Type	Outlet_Year
8523	0.007565	107.862200	1550	4655	9294	24
8524	0.038428	87.319800	1543	7996	9294	16
8525	0.099575	241.753800	925	7996	1805	25
8526	0.015388	155.034000	1543	7996	9294	16
8527	0.118599	234.230000	1559	4655	1559	38
8528	0.063817	117.149200	1550	7996	9294	26
8529	0.082602	50.103400	1546	4655	1546	14
8530	0.015782	81.059200	1559	4655	1559	38
8531	0.123365	95.743600	1548	7996	9294	21
8532	0.005698	186.892400	1543	7996	9294	16

In [85]:
```
# --- Normalizing the Test dataset for prediction
test_.skew()
```

Out[85]:
```
Item_Visibility      1.307908
Item_MRP             0.136182
Outlet_Identifier   -2.244334
Outlet_Size         -0.803264
Outlet_Type         -0.650193
Outlet_Year          0.396306
dtype: float64
```

In [86]:
```
for i in test_.columns:
    test_[i] =np.log(test_[i])
```

In [87]:
```
fig, ax = plt.subplots(2,3,figsize = (35,18))
sns.distplot(test_["Item_Visibility"], kde =True, ax=ax[0,0], color = "red")
sns.distplot(test_["Item_MRP"], kde =True, ax=ax[0,1], color = "blue")
sns.distplot(test_["Outlet_Identifier"], kde =True, ax=ax[0,2], color = "orange")
sns.distplot(test_["Outlet_Size"], kde =True, ax=ax[1,0], color = "magenta")
sns.distplot(test_["Outlet_Type"], kde =True, ax=ax[1,1], color = "black")
sns.distplot(test_["Outlet_Year"], kde =True, ax=ax[1,2])
```



```
In [88]: test_.head(10).style.set_properties(**{'border': '1px solid black'})
```

	Item_Visibility	Item_MRP	Outlet_Identifier	Outlet_Size	Outlet_Type	Outlet_Year
8523	-4.884245	4.680854	7.346010	8.445697	9.137124	3.178054
8524	-3.258977	4.469577	7.341484	8.986697	9.137124	2.772589
8525	-2.306845	5.487920	6.829794	8.986697	7.498316	3.218876
8526	-4.174142	5.043644	7.341484	8.986697	9.137124	2.772589
8527	-2.132005	5.456304	7.351800	8.445697	7.351800	3.637586
8528	-2.751732	4.763448	7.346010	8.986697	9.137124	3.258097
8529	-2.493727	3.914089	7.343426	8.445697	7.343426	2.639057
8530	-4.148854	4.395180	7.351800	8.445697	7.351800	3.637586
8531	-2.092604	4.561674	7.344719	8.986697	9.137124	3.044522
8532	-5.167564	5.230533	7.341484	8.986697	9.137124	2.772589

```
In [89]: # --- Applying Linear Regression model to dataset ---
         item_outsale_pred = reg.predict(test_)
         item_outsale_pred

Out[89]: array([7.35281546, 7.02757116, 6.23741794, ..., 7.48754227, 7.94021682,
                7.08316885])
```

```
In [90]: # --- Performing inverse transformation ---
         actual_item_outsale = np.exp(item_outsale_pred+1)
         actual_item_outsale

Out[90]: array([4242.10742543, 3064.28995454, 1390.4989808 , ..., 4853.92178681,
                7632.85182535, 3239.48242958])
```

```
In [91]: # --- ADDING THE PREDICTED ITEM_OUTLET_SALE COLUMNS TO TEST DATA ---
         test = pd.read_csv("C:/Users/Meghna Midya/Downloads/Dataset/Test.csv")
         test["Item_Outlet_Sales"] = actual_item_outsale
```

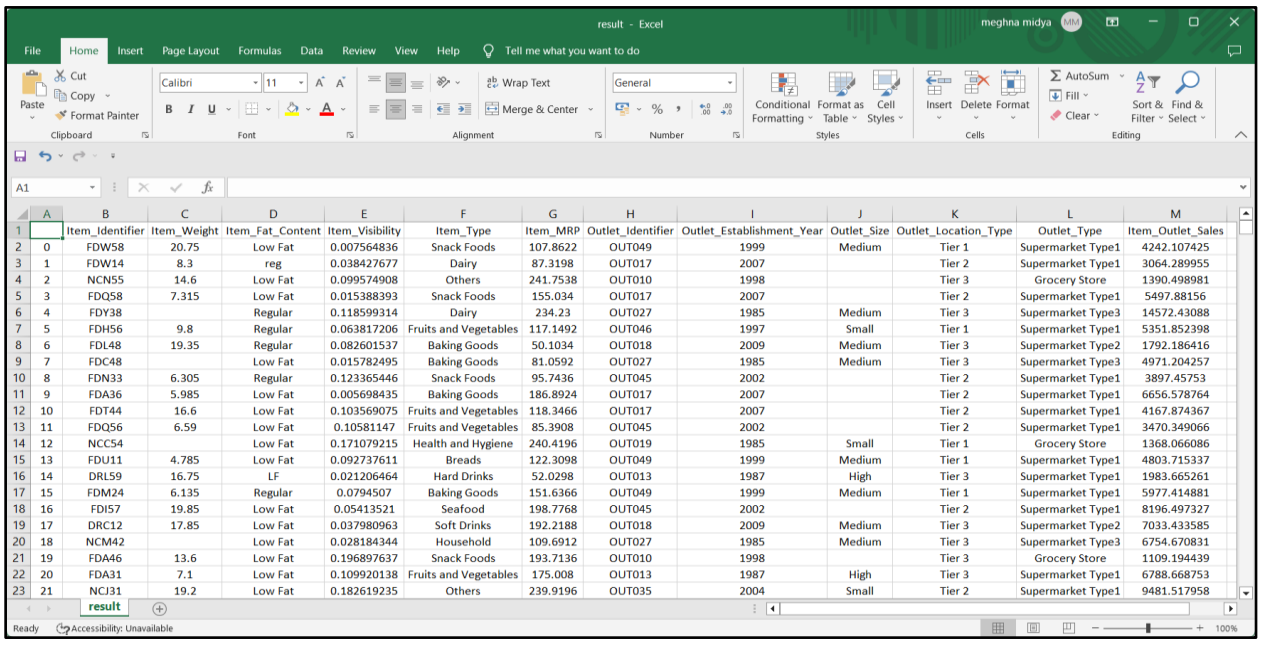
```
In [92]: test
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	Tier 1	Supermarket Type1	4242.107425
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	NaN	Tier 2	Supermarket Type1	3064.289955
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	Tier 3	Grocery Store	1390.498981
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	Tier 2	Supermarket Type1	5497.881560
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	Tier 3	Supermarket Type3	14572.430881
...	...	...	...	...	...	...	...	...	...	...	...	...
5676	FDB58	10.500	Regular	0.013496	Snack Foods	141.3154	OUT046	1997	Small	Tier 1	Supermarket Type1	6489.026943
5677	FDD47	7.600	Regular	0.142991	Starchy Foods	169.1448	OUT018	2009	Medium	Tier 3	Supermarket Type2	6165.326623
5678	NCO17	10.000	Low Fat	0.073529	Health and Hygiene	118.7440	OUT045	2002	NaN	Tier 2	Supermarket Type1	4853.921787
5679	FDJ26	15.300	Regular	0.000000	Canned	214.6218	OUT017	2007	NaN	Tier 2	Supermarket Type1	7632.851825
5680	FDU37	9.500	Regular	0.104720	Canned	79.7960	OUT045	2002	NaN	Tier 2	Supermarket Type1	3239.482430

5681 rows × 12 columns

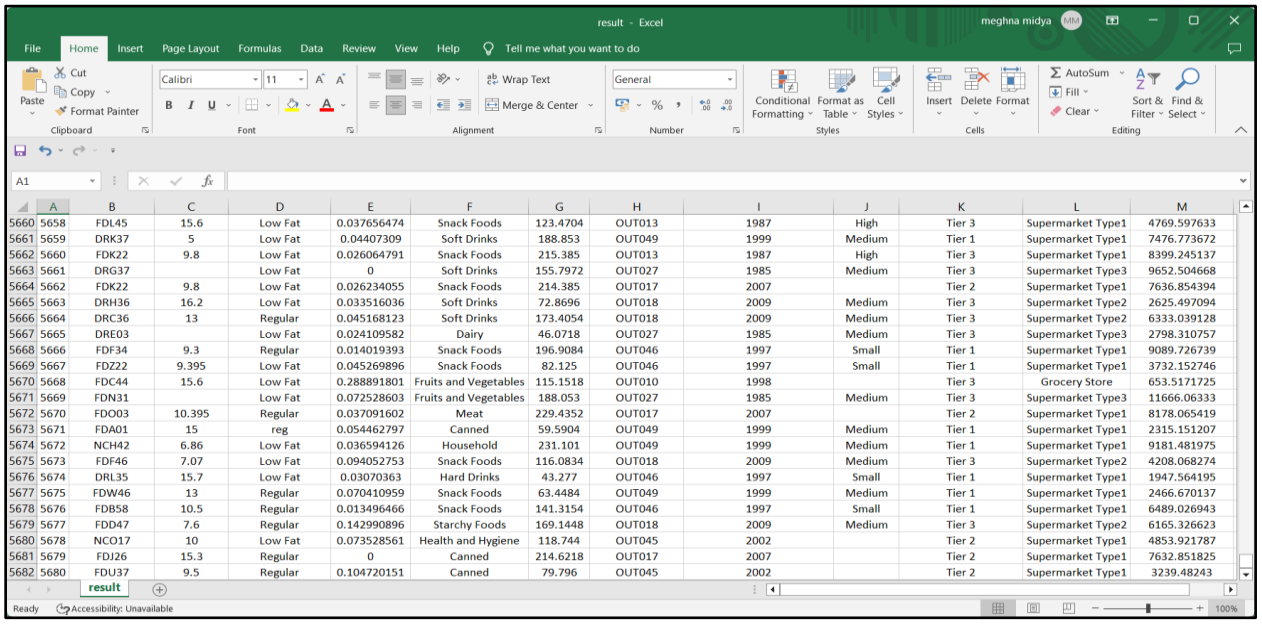
```
In [93]: test.to_csv('result.csv')
```

```
,Item_Identifier,Item_Weight,Item_Fat_Content,Item_Visibility,Item_Type,Item_MRP,Outlet_Identifier,Outlet_Establishment_Year,Outlet_Size,Outlet_Location_Type,Outlet_Type,Item_Outlet_Sales
0,FDW58,20.75,Low Fat,0.007564836,Snack Foods,107.8622,OUT049,1999,Medium,Tier 1,Supermarket Type1,4242.1074254254745
1,FDW14,8.3,reg,0.038427677,Dairy,87.3198,OUT017,2007,,Tier 2,Supermarket Type1,3064.2899545392092
2,NCN55,14.6,Low Fat,0.099574908,Others,241.7538,OUT010,1998,,Tier 3,Grocery Store,1390.498980804977
3,FDQ58,7.315,Low Fat,0.015388393,Snack Foods,155.034,OUT017,2007,,Tier 2,Supermarket Type1,5497.881559953636
4,FDY38,,Regular,0.118599314,Dairy,234.23,OUT027,1985,Medium,Tier 3,Supermarket Type3,14572.43088134675
5,FDH56,9.8,Regular,0.063817206,Fruits and Vegetables,117.1492,OUT046,1997,Small,Tier 1,Supermarket Type1,5351.852398043262
6,FDL48,19.35,Regular,0.082601537,Baking Goods,50.1034,OUT018,2009,Medium,Tier 3,Supermarket Type2,1792.186415860979
7,FDC48,,Low Fat,0.015782495,Baking Goods,81.0592,OUT027,1985,Medium,Tier 3,Supermarket Type3,4971.204256538935
8,FDN33,6.305,Regular,0.123365446,Snack Foods,95.7436,OUT045,2002,,Tier 2,Supermarket Type1,3897.4575295352424
9,FDA36,5.985,Low Fat,0.005698435,Baking Goods,186.8924,OUT017,2007,,Tier 2,Supermarket Type1,6656.578764172925
10,FDT44,16.6,Low Fat,0.103569075,Fruits and Vegetables,118.3466,OUT017,2007,,Tier 2,Supermarket Type1,4167.874367070882
11,FDQ56,6.59,Low Fat,0.10581147,Fruits and Vegetables,85.3908,OUT045,2002,,Tier 2,Supermarket Type1,3470.349066419368
12,NCC54,,Low Fat,0.171079215,Health and Hygiene,240.4196,OUT019,1985,Small,Tier 1,Grocery Store,1368.0660856457257
13,FDU11,4.785,Low Fat,0.092737611,Breads,122.3098,OUT049,1999,Medium,Tier 1,Supermarket Type1,4803.715337391876
14,DRL59,16.75,LF,0.021206464,Hard Drinks,52.0298,OUT013,1987,High,Tier 3,Supermarket Type1,1983.6652605364193
15,FDM24,6.135,Regular,0.0794507,Baking Goods,151.6366,OUT049,1999,Medium,Tier 1,Supermarket Type1,5977.4148814051805
16,FDI57,19.85,Low Fat,0.05413521,Seafood,198.7768,OUT045,2002,,Tier 2,Supermarket Type1,8196.497326914427
17,DRC12,17.85,Low Fat,0.037980963,Soft Drinks,192.2188,OUT018,2009,Medium,Tier 3,Supermarket Type2,7033.433585404057
18,NCM42,,Low Fat,0.028184344,Household,109.6912,OUT027,1985,Medium,Tier 3,Supermarket Type3,6754.670831483472
19,FDA46,13.6,Low Fat,0.196897637,Snack Foods,193.7136,OUT010,1998,,Tier 3,Grocery Store,1109.19443934556
20,FDA31,7.1,Low Fat,0.109920138,Fruits and Vegetables,175.008,OUT013,1987,High,Tier 3,Supermarket Type1,6788.668753387096
21,NCJ31,19.2,Low Fat,0.182619235,Others,239.9196,OUT035,2004,Small,Tier 2,Supermarket Type1,9481.517957536838
22,FDG52,13.65,LF,0.065630844,Frozen Foods,47.7402,OUT046,1997,Small,Tier 1,Supermarket Type1,2149.600999841996
23,NCL19,,Low Fat,0.027447057,Others,142.347,OUT019,1985,Small,Tier 1,Grocery Store,805.1910215793159
24,FDS10,19.2,Low Fat,0.035178935,Snack Foods,180.7318,OUT035,2004,Small,Tier 2,Supermarket Type1,7125.8362575144565
25,FDX22,6.785,Regular,0.038455125,Snack Foods,209.4928,OUT010,1998,,Tier 3,Grocery Store,1203.7193505804369
26,NCF19,13.0,Low Fat,0.035102094,Household,47.6034,OUT035,2004,Small,Tier 2,Supermarket Type1,1837.0254916569236
27,NCE06,5.825,Low Fat,0.091485232,Household,161.3894,OUT046,1997,Small,Tier 1,Supermarket Type1,7407.424257893474
28,DRC27,13.8,Low Fat,0.058102469,Dairy,244.6802,OUT046,1997,Small,Tier 1,Supermarket Type1,11312.717789407889
29,FDE21,12.8,LF,0.022940349,Fruits and Vegetables,116.5492,OUT035,2004,Small,Tier 2,Supermarket Type1,4565.577324236823
30,NCR42,,Low Fat,0.06737681,Household,32.09,OUT019,1985,Small,Tier 1,Grocery Store,177.00222309804056
31,FDX51,9.5,Regular,0.022148582,Meat,194.9452,OUT018,2009,Medium,Tier 3,Supermarket Type2,7140.028232637184
32,NCR06,12.5,Low Fat,0.006792707,Household,42.4112,OUT018,2009,Medium,Tier 3,Supermarket Type2,1518.1048325461882
33,FDU31,,Regular,0.024870035,Fruits and Vegetables,217.7508,OUT027,1985,Medium,Tier 3,Supermarket Type3,13559.96794081794
34,FDU59,5.78,Low Fat,0.096931426,Breads,164.2552,OUT017,2007,,Tier 2,Supermarket Type1,5815.784017941138
```



result - Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
2	0	FDW58	20.75	Low Fat	0.007564836	Snack Foods	107.8622	OUT049	1999	Medium	Tier 1	Supermarket Type1	4242.107425
3	1	FDW14	8.3	reg	0.038427677	Dairy	87.3198	OUT017	2007		Tier 2	Supermarket Type1	3064.289955
4	2	NCN55	14.6	Low Fat	0.099574908	Others	241.7538	OUT010	1998		Tier 3	Grocery Store	1390.498981
5	3	FDQ58	7.315	Low Fat	0.015388393	Snack Foods	155.034	OUT017	2007		Tier 2	Supermarket Type1	5497.88156
6	4	FDY38		Regular	0.118599314	Dairy	234.23	OUT027	1985	Medium	Tier 3	Supermarket Type3	14572.43088
7	5	FDH56	9.8	Regular	0.063817206	Fruits and Vegetables	117.1492	OUT046	1997	Small	Tier 1	Supermarket Type1	5351.852398
8	6	FDL48	19.35	Regular	0.082601537	Baking Goods	50.1034	OUT018	2009	Medium	Tier 3	Supermarket Type2	1792.186416
9	7	FDC48		Low Fat	0.015782495	Baking Goods	81.0592	OUT027	1985	Medium	Tier 3	Supermarket Type3	4971.204257
10	8	FDN33	6.305	Regular	0.123365446	Snack Foods	95.7436	OUT045	2002		Tier 2	Supermarket Type1	3897.45753
11	9	FDA36	5.985	Low Fat	0.005698435	Baking Goods	186.8924	OUT017	2007		Tier 2	Supermarket Type1	6656.578764
12	10	FDT44	16.6	Low Fat	0.103569075	Fruits and Vegetables	118.3466	OUT017	2007		Tier 2	Supermarket Type1	4167.874367
13	11	FDQ56	6.59	Low Fat	0.10581147	Fruits and Vegetables	85.3908	OUT045	2002		Tier 2	Supermarket Type1	3470.349066
14	12	NCC54		Low Fat	0.171079215	Health and Hygiene	240.4196	OUT019	1985	Small	Tier 1	Grocery Store	1368.066086
15	13	FDU11	4.785	Low Fat	0.092737611	Breads	122.3098	OUT049	1999	Medium	Tier 1	Supermarket Type1	4803.715337
16	14	DRL59	16.75	LF	0.021206464	Hard Drinks	52.0298	OUT013	1987	High	Tier 3	Supermarket Type1	1983.665261
17	15	FDM24	6.135	Regular	0.0794507	Baking Goods	151.6366	OUT049	1999	Medium	Tier 1	Supermarket Type1	5977.414881
18	16	FDI57	19.85	Low Fat	0.05413521	Seafood	198.7768	OUT045	2002		Tier 2	Supermarket Type1	8196.497327
19	17	DRC12	17.85	Low Fat	0.037980963	Soft Drinks	192.2188	OUT018	2009	Medium	Tier 3	Supermarket Type2	7033.433585
20	18	NCM42		Low Fat	0.028184344	Household	109.6912	OUT027	1985	Medium	Tier 3	Supermarket Type3	6754.670831
21	19	FDA46	13.6	Low Fat	0.196897637	Snack Foods	193.7136	OUT010	1998		Tier 3	Grocery Store	1109.194439
22	20	FDA31	7.1	Low Fat	0.109920138	Fruits and Vegetables	175.008	OUT013	1987	High	Tier 3	Supermarket Type1	6788.668753
23	21	NCJ31	19.2	Low Fat	0.182619235	Others	239.9196	OUT035	2004	Small	Tier 2	Supermarket Type1	9481.517958



result - Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M
5660	5658	FDL45	15.6	Low Fat	0.037656474	Snack Foods	123.4704	OUT013	1987	High	Tier 3	Supermarket Type1	4769.597633
5661	5659	DRK37	5	Low Fat	0.04407309	Soft Drinks	188.853	OUT049	1999	Medium	Tier 1	Supermarket Type1	7476.773672
5662	5660	FDK22	9.8	Low Fat	0.026064791	Snack Foods	215.385	OUT013	1987	High	Tier 3	Supermarket Type1	8399.245137
5663	5661	DRG37		Low Fat	0	Soft Drinks	155.7972	OUT027	1985	Medium	Tier 3	Supermarket Type3	9652.504668
5664	5662	FDK22	9.8	Low Fat	0.026234055	Snack Foods	214.385	OUT017	2007		Tier 2	Supermarket Type1	7636.854394
5665	5663	DRH36	16.2	Low Fat	0.033516036	Soft Drinks	72.8696	OUT018	2009	Medium	Tier 3	Supermarket Type2	2625.497094
5666	5664	DRC36	13	Regular	0.045168123	Soft Drinks	173.4054	OUT018	2009	Medium	Tier 3	Supermarket Type2	6333.039128
5667	5665	DRE03		Low Fat	0.024109582	Dairy	46.0718	OUT027	1985	Medium	Tier 3	Supermarket Type3	2798.310757
5668	5666	FDF34	9.3	Regular	0.014019393	Snack Foods	196.9084	OUT046	1997	Small	Tier 1	Supermarket Type1	9089.726739
5669	5667	FDZ22	9.395	Low Fat	0.045269896	Snack Foods	82.125	OUT046	1997	Small	Tier 1	Supermarket Type1	3732.152746
5670	5668	FDC44	15.6	Low Fat	0.288891801	Fruits and Vegetables	115.1518	OUT010	1998		Tier 3	Grocery Store	653.5171725
5671	5669	FDN31		Low Fat	0.072528603	Fruits and Vegetables	188.053	OUT027	1985	Medium	Tier 3	Supermarket Type3	11666.06333
5672	5670	FDO03	10.395	Regular	0.037091602	Meat	229.4352	OUT017	2007		Tier 2	Supermarket Type1	8178.065419
5673	5671	FDA01	15	reg	0.054462797	Canned	59.5904	OUT049	1999	Medium	Tier 1	Supermarket Type1	2315.151207
5674	5672	NCH42	6.86	Low Fat	0.036594126	Household	231.101	OUT049	1999	Medium	Tier 1	Supermarket Type1	9181.481975
5675	5673	FDF46	7.07	Low Fat	0.094052753	Snack Foods	116.0834	OUT018	2009	Medium	Tier 3	Supermarket Type2	4208.068274
5676	5674	DRL35	15.7	Low Fat	0.03070363	Hard Drinks	43.277	OUT046	1997	Small	Tier 1	Supermarket Type1	1947.564195
5677	5675	FDW46	13	Regular	0.070410959	Snack Foods	63.4484	OUT049	1999	Medium	Tier 1	Supermarket Type1	2466.670137
5678	5676	FDB58	10.5	Regular	0.013496466	Snack Foods	141.3154	OUT046	1997	Small	Tier 1	Supermarket Type1	6489.026943
5679	5677	FDD47	7.6	Regular	0.142990896	Starchy Foods	169.1448	OUT018	2009	Medium	Tier 3	Supermarket Type2	6165.326623
5680	5678	NCO17	10	Low Fat	0.073528561	Health and Hygiene	118.744	OUT045	2002		Tier 2	Supermarket Type1	4853.921787
5681	5679	FDJ26	15.3	Regular	0	Canned	214.6218	OUT017	2007		Tier 2	Supermarket Type1	7632.851825
5682	5680	FDU37	9.5	Regular	0.104720151	Canned	79.796	OUT045	2002		Tier 2	Supermarket Type1	3239.48243

CONCLUSION

Our forecasts assist Big Mart in refining its procedures and plans, allowing them to maximize its profit. The expected outcomes will be extremely beneficial for the company's leaders to understand their sales and profitability. This will also provide them insight into their future Big Mart sites or centers.

Sales forecasting is essential to the business sector in all industries. With the help of the sales forecasts, sales revenue analysis will help to get the details needed to estimate both the revenue and the income. Different types of Machine Learning techniques such as Linear Regression, Lasso Regression and Ridge Regression have been evaluated on sales data to find the critical factors that influence sales to provide a solution for forecasting sales.

Linear Regression has the highest accuracy when distinguished together. Hence, we can say that Linear Regression is the better algorithm for efficient sales analysis. This methodology is primarily used by shopping marts, groceries, Brand outlets, etc. The data analysis applied to the predictive machine learning models provides a very effective way to manage sales. This approach is very much encouraged in today's world since it aids many companies, enterprises, researchers and brands for outcomes that lead to the management of their profits, sales, inventory management, data research and customer demand.

In this project, the basics of machine learning and the associated data processing and modeling algorithms have been described, followed by their application for the task of sales prediction in Big Mart shopping centers at different locations. When put into practice, the predicted results demonstrate the relationship between the many factors taken into account and how a specific site of a medium size achieved the best sales, suggesting that other retail locations should adopt a similar strategy for increased sales.

REFERENCES

1. https://www.kaggle.com/dataset
2. https://www.google.com
3. https://matplotlib.org/stable/tutorials/colors/colormaps.html#classes-of-colormaps
4. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html
5. https://www.projectpro.io/article/top-5-libraries-for-data-science-in-python/196#mcetoc_1fu5v5t8dp
6. https://www.tutorialspoint.com/python/index.htm
7. https://www.tutorialspoint.com
8. https://www.geeksforgeeks.org

PLAGIARISM REPORT

Plagiarism Scan Report

Report Generated on: Apr 16,2023

Plagiarised

Unique

Total Words:	930
Total Characters:	6085
Plagiarized Sentences:	0
Unique Sentences:	42 (100%)

Content Checked for Plagiarism

ABSTRACT

In recent years, shopping malls and Big Marts manage and keep a record of their sales data for any and every unique item with the goal to estimate client demand in future periods. Every firm and organization is working on meeting customer needs while also improving their inventory management in the present era of achieving new heights of progress. In a data warehouse, these data stores hold a vast quantity of customer data and specific item information. More importantly, abnormalities and regular patterns are discovered through mining the data warehouse. The resulting data may be utilized to forecast future sales volume using various machine-learning algorithms. In this project, we provide a linear regression based predictive model to forecast sales for a firm like Big Mart. A detailed comparison of the model to others in terms of performance measures is also provided. With the use of numerous datasets gathered for Big Mart and the process used to create a predictive model, very accurate findings are produced, and these observations may be used to make decisions to increase sales.

INTRODUCTION

Day-by-day competition among different shopping malls as well as big marts is getting more serious only due to the rapid growth of global malls and online shopping. In order to draw in more consumers depending on the moment, every mall or market tries to give unique, limited-time deals. This way, the volume of sales for each item can be forecasted for inventory management of the organization, logistics, transport service, etc. The collecting of sales data for commodities or products together with their different dependent or independent aspects is a crucial step in today's contemporary world for helping to estimate future demand and inventory management. Large shopping centers like major malls and marts are examples of this. The dataset, which was created utilizing a range of dependent and independent variables, is made up of information on the attributes of the items and information acquired from customers, and inventory management information maintained in a data warehouse. The data is then changed in order to generate accurate forecasts as well as to collect new and exciting results that shed new light on our comprehension of the task's data. In order to predict forthcoming sales, this might then be utilized in conjunction with machine learning techniques. In this project, we address the problem of anticipating big-mart sales or projecting an item based on the consumers' expected future demand in various big-mart stores across various locations and goods. For the prediction or forecasting of sales volume, several machine learning methods, such as linear regression analysis, random forest, etc., are applied. In any retail center, the ability to predict sales is crucial since strong sales are the lifeblood of every business. Always a better prediction is helpful, to develop as well as to enhance the strategies of business about the marketplace which is also helpful to improve the knowledge of marketplaces. A conventional sales prediction research may assist in carefully analyzing prior scenarios or conditions, and the inference can then be used to client acquisition, money insufficiency, and strengths before making a budget and marketing strategies for the following year. In other words, sales forecasting is based on the resources that were accessible in the past. In-depth knowledge of the past is required for enhancing and improving the likelihood of marketplaces irrespective of any circumstances, especially the external circumstance, which allows for the preparation of the upcoming needs of the business.

Data Analysis:

Data visualization tools and technologies are crucial in the space of big data to analyze vast volumes of information. Data visualization is the depiction of data using conventional graphics like infographics, plots, and even animations. These informational visual representations make complicated data relationships and data-driven insights simple to comprehend.

Machine Learning:

The amount of data accessible is growing daily, and this massive volume of unprocessed data must be carefully analyzed in order to provide outcomes that meet the current standards for being highly useful and flawlessly pure. Data is incredibly important in current aspects, therefore as technology advances, so will the analysis and interpretation of data to provide effective outcomes. Both supervised and unsupervised types of tasks are dealt with in machine learning, and typically, a classification-type issue serves as a source for knowledge acquisition. Machine Learning appears in many guises. Various machine learning algorithm includes:

1. Linear Regression

Linear regression is a popular supervised machine learning algorithm used for predicting numerical values. It models the relationship between an independent variable (input feature) and a dependent variable (output or target variable) as a linear equation. The goal of linear regression is to find the best-fit line or hyperplane that minimizes the error between the predicted values and the actual values of the target variable.

2. Lasso Regression

The term Lasso regression is an acronym for Least Absolute Shrinkage and Selection Operator. Lasso regression is a type of regularization. It is preferred over regression approaches for more precise prediction. This model makes advantage of shrinkage. Shrinkage is the process of shrinking data values towards a central point known as the mean.

PROJECT DESCRIPTION

PROBLEM STATEMENT

Sales forecasting is a quite common real-world challenge that every organization experiences at some point in its entire existence. If done correctly, it may have a substantial influence on the company's success and performance. Big Mart's data scientists gathered 2013 sales data for numerous items across stores in several cities. Certain characteristics of each product and shop have also been determined. The goal is to create a predictive model that predicts the sales of each product at a certain location.

Plagiarism Scan Report

Report Generated on: Apr 16,2023

Plagiarised

Unique

Total Words:	748
Total Characters:	4948
Plagiarized Sentences:	0.8
Unique Sentences:	39.2 (98%)

Content Checked for Plagiarism

PROPOSED SYSTEM

The project is about building a model to predict accurate results for the dataset of Big Mart sales. This undergoes several sequences of steps as mentioned in the methodology and in this work, we propose a model using the Linear Regression technique. Every step plays a vital role in building the proposed model. In our model, we have used the 2013 Big Mart dataset. After preprocessing and filling in missing values, we used an ensemble classifier using Linear Regression, Ridge Regression and Lasso regression. Both the R2 score and RSME are used as accuracy metrics for predicting sales in Big Mart.

OBJECTIVES

Converting data into an appropriate form using various preprocessing techniques for the implementation of Machine Learning algorithms.

The objective of this project is to predict future sales from given data of the previous years using Machine Learning Techniques.

To determine and conclude the best model which is more efficient and gives fast and accurate results.

DATASET DESCRIPTION

The data which was required for the project is collected through a Kaggle Dataset. In the project, I have used the 2013 Sales data of Big Mart as the dataset. Big Mart's data scientists collected sales data from their 10 stores situated in different locations with each store having different products. The dataset set contains certain attributes like:

Variable Type Description

Item_Identifier Numeric Unique product ID.

Item_Weight Numeric Weight of the product.

Item_Fat_Content Categorical Indicates if the product is low fat or not.

Item_Visibility Numeric The percentage of a store's total display area dedicated to a specific product.

Item_Type Categorical It specifies the category to which the product belongs.

Item_MRP Numeric The product's Maximum Retail Price (list price).

Outlet_Identifier Numeric Unique Store ID.

Outlet_Establishment_Year Numeric The year in which the store was established.

Outlet_Size Categorical Specified the size of the store in terms of the amount of ground area it covers.

Outlet_Location_Type Categorical The sort of the city in which the shop is situated.

Outlet_Type Categorical Whether the shop is only a grocery store or a supermarket.

Item_Outlet_Sales Numeric Sales figures for the product at the shop. This is the outcome variable that has to be forecasted

HARDWARE & SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENT

Processor : Intel Core i5 CPU 1.00 GHz or more

RAM : 8.00 GB

Hard Disk : 256SSD or more

Monitor : 15" CRT or LCD Monitor

Keyboard : Normal or Multimedia

Mouse : Compatible Mouse

SOFTWARE REQUIREMENT

Operating System : Windows 11 / Windows 10

Software : Anaconda Navigator

Software : Jupyter Notebook

Software : Microsoft Excel (.csv)

TECHNOLOGY USED

Programming Language : Python

Network : Internet Connectivity

SOFTWARE SPECIFICATION

JUPYTER NOTEBOOK

Jupyter Notebook is a popular open-source tool that provides an interactive computing environment for creating, running and sharing live code, equations, visualizations and narratives in a web-based notebook format. It is compatible with a number of programming languages, including Python, R, Julia, and others.

Jupyter Notebook allows you to write and execute code in cells, which are organized in a linear or non-linear order. Each cell can contain code, markdown text or even multimedia elements, making it a flexible tool for data analysis, machine learning, scientific computing and other computational tasks.

Some key features of the Jupyter Notebook include:

1. Code Execution:

One can write and run code in individual cells, which allows for interactive and iterative development. You can also store variables, functions, and other objects in memory and access them across cells.

2. Markdown Support:

You can add formatted text, equations and images using Markdown cells, which allows for documentation, explanations, and visualizations alongside your code.

3. Visualization:

Jupyter Notebook supports rich visualizations using popular data visualization libraries such as Matplotlib, Seaborn, Plotly and others. One can create static or interactive visualizations directly in the notebook.

4. Collaboration and Sharing:

Jupyter Notebook allows you to share your notebooks with others, making it easy to collaborate on data analysis or machine learning projects. Notebooks can be exported in various formats, such as HTML, PDF or slides, for easy sharing or presentation.

5. Extensibility:

Jupyter Notebook has a large ecosystem of extensions and plugins that enhance its functionality, including code snippets, debugging tools and more.

6. Integration with other tools:

Jupyter Notebook can be integrated with other popular data science and machine learning tools, such as NumPy, Pandas, Scikit-Learn, TensorFlow and others, making it a powerful tool for end-to-end data analysis and model development workflows.

Jupyter Notebook can be launched using Anaconda or it can be directly accessed.

Group Policy Settings Reference Spreadsheet for Windows ...Release history for Office Deployment Tool (ODT)

Sep 21, 2022 — Supported Operating System Windows 11, Windows 10, Windows 7, Windows 8, Windows 8.1, Windows Server 2008 R2, Windows Server 2012, ...Supported Operating System: Windows 11, Windows 10, Windows 8.1, Windows Server 2022, Windows Server 2019, and Windows Server 2016.
https://www.microsoft.com/en-us/download/details.aspx?id=104594

25%

Plagiarism Scan Report

Report Generated on: Apr 16,2023

Plagiarised

Unique

Total Words:	988
Total Characters:	6711
Plagiarized Sentences:	0
Unique Sentences:	49 (100%)

Content Checked for Plagiarism

EXPERIMENTATION ENVIRONMENT
PYTHON

Python is a well-known high-level, interpreted programming language that has been used for a wide range of tasks, including data analysis, web development, scientific computing, and artificial intelligence. It was initially made available in 1991 by Guido van Rossum, and since then it has grown to be one of the most widely used programming languages worldwide.
Python is known for its simplicity and readability, which makes it easy to learn and use, especially for beginners. Its syntax is simple and fast to comprehend. Python supports multiple programming paradigms, including procedural, object-oriented and functional programming, and it is cross-platform, meaning that it can run on a wide variety of operating systems, including Windows, Mac and Linux. In Data Science, Python is popularly used and as well as being a dynamic and open-source language it is a top choice for many users.
Programmers use Python's excellent data science tools on a regular basis to overcome difficulties. Python has a vast standard library that includes modules for tasks such as web development, file handling, networking, data processing, and more. Python libraries are typically written in Python or a combination of Python and other programming languages and are distributed as open-source software, allowing developers to access, modify and contribute to the codebase. It also has a large and active community of developers who contribute to open-source libraries and frameworks, making it easy to find

1. NumPy
NumPy (Numerical Python) is the core Python library for numerical calculation; it includes a strong N-dimensional array object. It is a general-purpose programme for processing multidimensional arrays that provides the ability to deal with high-performance arrays. NumPy arrays provide vectorization of mathematical operations, which is more efficient than Python's looping structures.
2. Pandas
The data science life cycle is incomplete without Pandas (Python Data Analysis). Pandas Python is one of those data analysis packages that includes high-level data structures and simple data manipulation capabilities. Providing a simple yet efficient approach to analyse data needs the ability to index, retrieve, divide, connect, restructure, and perform numerous other analysis on both multi and single-dimensional data. Pandas offers quick, adaptable data structures, like data frame CDs, that make it simple and easy to work with structured data.
3. Matplotlib
Matplotlib package aids in visualizing and plotting data to make static, animated and interactive visualizations. Matplotlib is a 2D graphical Python library. It does, however, enable 3D graphics, but only to a limited extent. Because of the graphs and plots that it produces, it is frequently used for data visualisation.
4. Missingno
A highly attractive approach to see the distribution of NaN values is provided by the Missingno library. Missingno is a Python library that is compatible with Pandas.
5. Plotly
Plotly is a free and open-source Python framework for 3D data visualisation. Plotly supports a variety of chart forms, including scatter plots, histograms, line charts, bar charts, box plots, multiple axes,

sparklines, dendrograms, three-dimensional graphs, and more. Plotly also includes contour plots, distinguishing them from other data visualization frameworks.

6. Seaborn

Seaborn is a well-known Python data visualisation package that is based on Matplotlib and provides a high-level interface for making visually appealing and helpful statistical visualisations. Seaborn is particularly well-suited for visualizing complex datasets and is often used in data science, machine learning, and statistical analysis applications.

7. SciPy

Another free and open-source Python data science library that is extensively used for high-level mathematical calculation is SciPy (Scientific Python). Since it extends NumPy and includes a number of user-friendly and efficient scientific computation techniques, it is widely used for scientific and technical calculations. SciPy includes a number of high-level data manipulation and visualisation commands and classes. SciPy is skilled for data processing and system prototyping.

8. Sci-Kit Learn (sklearn)

Scikit-learn, commonly abbreviated as sklearn, is a popular Python library for machine learning that provides a wide range of tools for tasks such as classification, regression, clustering, dimensionality reduction, model evaluation and more. Scikit-learn is built on top of NumPy, SciPy and Matplotlib, and is widely used in machine learning and data science applications for developing predictive models and performing data analysis tasks.

METHODOLOGY

1. Dataset

Through the use of a Kaggle Dataset, the project's necessary data was gathered. Data collection is the process of gathering, acquiring and compiling data from various sources or methods. Datasets are used in a wide range of fields, including but not limited to, data science, machine learning, statistics, social sciences, healthcare, finance, marketing and more. They serve as the foundation for conducting research, analysis, and modeling and are crucial for generating insights, making informed decisions, and developing predictive or prescriptive models.

2. Defining the Problem

It is necessary to accurately define the data problem that is to be solved. The issue or goal of the data analysis and data prediction should be clearly stated. To establish your complete prediction method, it is essential that you comprehend the issue or goal.

Each business will encounter sales projection issues at some point throughout its existence. If done correctly, it may significantly affect the performance and profitability of that firm. Big Mart's data scientists have gathered sales information from 2013 for numerous items across stores in several cities. The goal is to create a prediction model that can forecast the sales of each item at certain stores.

3. Preprocessing Data

Preprocessing includes cleaning, converting, and preparing raw data so that it is appropriate for additional analysis or modeling. It is a critical stage in the data analysis and machine learning pipeline. Preprocessing is often necessary to address issues such as missing data, inconsistent formatting, noise, outliers or irrelevant information in the data, and to standardize or normalize the data to a common format or scale. Properly preprocessing data can significantly impact the accuracy, reliability and interpretability of the results obtained from the data analysis or modeling process.

Plagiarism Scan Report

Report Generated on: Apr 16,2023

Plagiarised

Unique

Total Words:	955
Total Characters:	6810
Plagiarized Sentences:	0
Unique Sentences:	46 (100%)

Content Checked for Plagiarism

4. Exploring and Visualizing Data
Exploring and visualizing data is an important step in the data analysis process that involves examining, summarizing, and visualizing the main characteristics, patterns, and relationships present in the data. Data exploration and visualization techniques are used to gain insights, identify trends, detect patterns, and uncover hidden information in the data, which can help inform decision-making, generate hypotheses, or guide further analysis.

5. Splitting and Training Data
Splitting the data into training and testing datasets is necessary for any data analysis. The training dataset is used to train the prediction model, while the testing dataset is used to evaluate the performance of the trained model. Preprocessing the training and testing datasets separately or together ensures that any data transformations or imputations are applied consistently.

6. Model Building
Machine Learning prediction models are trained in this process and then later on evaluated using the data. Choose an appropriate prediction model based on the characteristics of your data and the problem you are trying to solve. Store the generated output in the required format.

EXPLORATORY DATA ANALYSIS
Exploratory Data Analysis is a critical step in the data analysis process that involves examining and analyzing data to understand its structure, patterns, relationships and characteristics. EDA helps in gaining insights, identifying trends, detecting anomalies and making data-driven decisions. Some key objectives of EDA include:
1. Data Understanding:
EDA involves thoroughly examining the data to understand its content, structure and quality. This includes checking data types, variable distributions, missing values, outliers and other data characteristics.
2. Data Visualization:
EDA often involves creating visualizations such as plots, charts and graphs to explore data visually. Visualizations can help in identifying patterns, trends and relationships in the data, making it easier to interpret and understand.
3. Data Summarization and Aggregation:
EDA involves aggregating and summarizing data to obtain descriptive statistics such as mean, median, mode, standard deviation and others. This can help in gaining insights into the central tendency, variability and distribution of the data.
4. Data Profiling:
EDA may involve creating data profiles, which provide an overview of the data characteristics, such as unique values, data ranges, data distributions and data quality metrics. Data profiling helps in identifying data issues and understanding the data's overall quality.
5. Data Cleansing and Preprocessing:
EDA may also involve identifying and addressing data quality issues, such as missing values, outliers, inconsistencies and errors. This may include data imputation, data cleaning, and data transformation techniques to prepare the data for further analysis.

PROJECT

Importing Libraries And Extracting Dataset

From the train dataset description, we get a clear idea about the mean value, standard deviation, minimum and maximum value of the numerical values and for categorical values we idea about their uniqueness, mostly occurred values and their frequency.

It is obvious from running train.info() that the Item_Weight and Outlet_Size variables in the train dataset have NaN values (Null values). Additionally, we can observe that the dataset has 7 categorical features (Item_Identifier, Item_Fat_Content, Item_Type, Outlet_Identifier, Outlet_Size, Outlet_Location_Type, and Outlet_Type) in addition to 5 numerical features (Item_Weight, Item_Visibility, Item_MRP, Outlet_Establishment_Year, and Item_Outlet_Sales).

From the test dataset description, we get a clear idea about the mean value, standard deviation, minimum and maximum value of the numerical values and for categorical values we idea about their uniqueness, mostly occurred values and their frequency.

It is obvious from running test.info() that the Item_Weight and Outlet_Size variables in the test dataset have NaN values. Also, we can see that the dataset has 4 Numerical features(Item_Weight, Item_Visibility, Item_MRP and Outlet_Establishment_Year) and 7 Categorical features(Item_Identifier, Item_Fat_Content, Item_Type, Outlet_Identifier, Outlet_Size, Outlet_Location_Type and Outlet_Type).

Exploratory Data Analysis And Data Preprocessing

Combining the training and testing datasets together for pre-processing both datasets in a single go. This will help to easily pre-process the data without repeating the same steps for both datasets.

In Item_Type, there are 16 various types of items that are listed in the dataset. Also, the number of times the items are repeated in the dataset is mentioned.

Finding the missing values:

Almost all the columns have no missing values except the "Item_Weight", "Outlet_Size" and "Item_Outlet_Sales" columns.
The total of missing values in each column is less than 25%, which means that imputation can still be done to fill in the missing values in the two columns("Item_Weight" and "Outlet_Size"). There are 2439 missing values in the "Item_Weight" column and 4016 missing values in the "Outlet_Size" column.

Replacing missing values with substitute values:
The "Item_Weight" column missing values can be replaced by their mean and the "Outlet_Size" column missing values can be replaced by their mode.

Modifying the inconsistent values:

Here, there are typos line case sensitive words and shortcuts. 'Low Fat' and 'Regular' must be the real unique values. Before implementing the model, this adjustment is required.

The Outlet_Establishment_Year contains the year in which the outlet was established. If we can get information on how many years the outlet has been working, then that data would be more useful.

Converting Categorical Variables into Numerical Values:

Converting And Splitting The Dataset

Visualizing Correlation:

Model Building
Removing less correlated values:

Normalization:

Identifying the best fitting model:

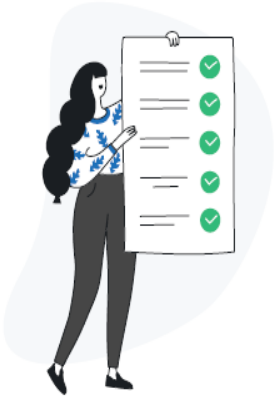
Fitting linear regression model:

Fitting lasso regression model:

Fitting ridge regression model:

Deciding Best-Fitting Model
RMSE:
Linear Regression: 0.537468458180603
Lasso Regression: 0.5524229989252929
Ridge Regression: 0.5374699611092182
R2_score:
Linear Regression: 72.96248381320213 %
Lasso Regression: 71.43696644494459 %
Ridge Regression: 72.96248381320213 %
A model is said to be the best fit if the R2_score is closer to 100% and has a better RMSE score.
1. Ridge and Linear Regression are more accurate than Lasso Regression.
2. R2_Score values for Ridge Regression and Linear Regression are identical. However, the RMSE value of the Linear Regression model is higher.
As a result, the best-fit model for the provided dataset is: LINEAR REGRESSION

Predicting Data



No Plagiarism Found

Plagiarism Scan Report

Report Generated on: Apr 16,2023

0% Plagiarised	100% Unique

Total Words:	308
Total Characters:	2362
Plagiarized Sentences:	0
Unique Sentences:	14 (100%)

Content Checked for Plagiarism

CONCLUSION

Our forecasts assist Big Mart in refining its procedures and plans, allowing them to maximize its profit. The expected outcomes will be extremely beneficial for the company's leaders to understand their sales and profitability. This will also provide them insight into their future Big Mart sites or centers.

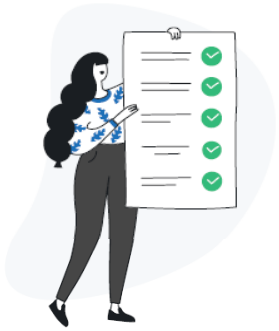
Sales forecasting is essential to the business sector in all industries. With the help of the sales forecasts, sales revenue analysis will help to get the details needed to estimate both the revenue and the income. Different types of Machine Learning techniques such as Linear Regression, Lasso Regression and Ridge Regression have been evaluated on sales data to find the critical factors that influence sales to provide a solution for forecasting sales.

Linear Regression has the highest accuracy when distinguished together. Hence, we can say that Linear Regression is the better algorithm for efficient sales analysis. This methodology is primarily used by shopping marts, groceries, Brand outlets, etc. The data analysis applied to the predictive machine learning models provides a very effective way to manage sales. This approach is very much encouraged in today's world since it aids many companies, enterprises, researchers and brands for outcomes that lead to the management of their profits, sales, inventory management, data research and customer demand.

In this project, the basics of machine learning and the associated data processing and modeling algorithms have been described, followed by their application for the task of sales prediction in Big Mart shopping centers at different locations. When put into practice, the predicted results demonstrate the relationship between the many factors taken into account and how a specific site of a medium size achieved the best sales, suggesting that other retail locations should adopt a similar strategy for increased sales.

REFERENCES

1. https://www.kaggle.com/dataset
2. https://www.google.com
3. https://matplotlib.org/stable/tutorials/colors/colormaps.html#classes-of-colormaps
4. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html
5. https://www.projectpro.io/article/top-5-libraries-for-data-science-in-python/196#mcetoc_1fu5v5t8dp
6. https://www.tutorialspoint.com/python/index.htm
7. https://www.tutorialspoint.com
8. https://www.geeksforgeeks.org



No Plagiarism Found