**BIG MART SALES PREDICTION**

**MODEL**

**PROJECT DOCUMENTATION**

**Created by:**

Jenish Maharjan

Kathmandu, Nepal

September 2024

Table of Contents

[1. Project Overview 1](#_Toc176798734)

[1.1. Objective 1](#_Toc176798735)

[1.2. Dataset Source: 1](#_Toc176798736)

[2. Dataset Description 2](#_Toc176798737)

[2.1. Features 2](#_Toc176798738)

[3. Data Cleaning Steps 3](#_Toc176798739)

[3.1. Handling Missing Values 3](#_Toc176798740)

[3.2. Handling Duplicates 6](#_Toc176798741)

[3.3. Handling Data Inconsistencies 6](#_Toc176798742)

[4. Exploratory Data Analysis (EDA) 7](#_Toc176798743)

[4.1. Summary Statistics 7](#_Toc176798744)

[4.2. Histogram 7](#_Toc176798745)

[4.3. Bar Plots for Categorical Variables 10](#_Toc176798746)

[4.4. Relationship Between Numerical Variables 12](#_Toc176798747)

[4.5. Correlation Matrix Heatmap 13](#_Toc176798748)

[5. Feature Engineering 15](#_Toc176798749)

[5.1. Label Encoding 15](#_Toc176798750)

[5.2. Feature Scaling 15](#_Toc176798751)

[5.3. Outlier Detection Using Isolation Forest 15](#_Toc176798752)

[6. Feature Selection 17](#_Toc176798753)

[6.1. Heatmap of Correlation Matrix: 17](#_Toc176798754)

[6.2. Feature Selection 18](#_Toc176798755)

[6.2.1. Removal of Features 18](#_Toc176798756)

[6.2.2. Selection of Features 18](#_Toc176798757)

[7. Model Development 19](#_Toc176798758)

[7.1. Data Preparation 19](#_Toc176798759)

[7.2. Data Splitting: 19](#_Toc176798760)

[7.3. Linear Regression Model 19](#_Toc176798761)

[7.4. Decision Tree Regression 19](#_Toc176798762)

[7.5. Random Forest Regression 20](#_Toc176798763)

[7.6. Model Evaluation 20](#_Toc176798764)

[7.7. Choice of Model 20](#_Toc176798765)

[7.8. Training the Model 20](#_Toc176798766)

[7.9. Visualization of Actual vs Predicted Values: 20](#_Toc176798767)

[8. Deployment 22](#_Toc176798768)

[8.1. Model Preprocessing, Training, and Saving 22](#_Toc176798769)

[8.2. Preprocessing Pipeline 22](#_Toc176798770)

[8.3. Preprocessing the Data 22](#_Toc176798771)

[8.3.1. Splitting the Data 22](#_Toc176798772)

[8.3.2. Training the Linear Regression Model 23](#_Toc176798773)

[8.3.3. Saving the Model and Preprocessor 23](#_Toc176798774)

## Project Overview

### Objective

The primary objective of this project is to predict Item\_Outlet\_Sales using various machine learning models. The dataset contains information about items and outlets, and the goal is to understand the factors influencing sales and build a predictive model.

Retailers often struggle with predicting future sales accurately. Inaccurate sales forecasts can lead to either overstocking or stockouts, both of which can negatively impact business operations and revenue. This project aims to develop a model that provides accurate predictions of item outlet sales, thereby aiding in better decision-making.

### Dataset Source:

* **Dataset Name:** Big Mart Sales data
* **Source:** https://www.kaggle.com/datasets/brijbhushannanda1979/bigmart-sales-data
* **Description:** The dataset includes details about items, their characteristics, and outlet details.

## Dataset Description

The Big Mart Sales dataset contains historical sales data for various Big Mart stores. This dataset is used to predict the sales of items based on various features. The data includes both training and testing sets.

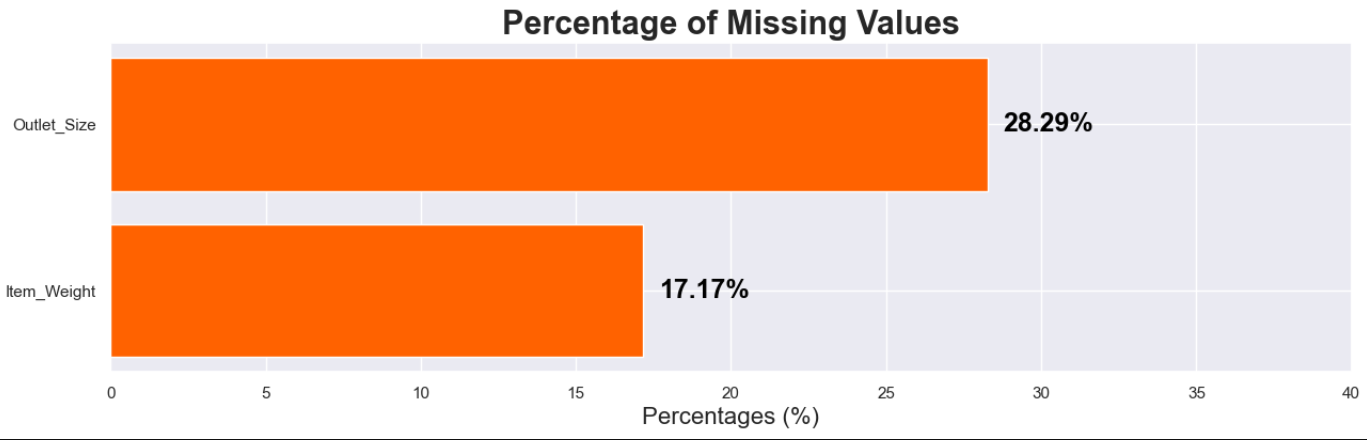
### Features

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N** | **Key Features** | **Description** | **Type** |
| 1 | **Item\_Identifier** | Unique identifier for items. | Categorical |
| 2 | **Item\_Fat\_Content** | Describes whether the item is low fat or regular fat. | Categorical |
| 3 | **Item\_Type** | The category or type of item (e.g., Snack, Beverage). | Categorical |
| 4 | **Item\_Weight** | Weight of the item in kilograms. | Numeric |
| 5 | **Item\_Visibility** | The visibility score of the item in the store (ratio of item’s visibility to total visibility). | Numeric |
| 6 | **Item\_MRP** | Maximum Retail Price of the item. | Numeric |
| 7 | **Outlet\_Identifier** | Unique identifier for outlets. | Categorical |
| 8 | **Outlet\_Size** | Size of the outlet (e.g., Small, Medium, Large). | Categorical |
| 9 | **Outlet\_Location\_Type** | Type of location where the outlet is situated (e.g., Urban, Rural). | Categorical |
| 10 | **Outlet\_Type** | Type of outlet (e.g., Supermarket, Hypermarket). | Categorical |
| 11 | **Outlet\_Establishment\_Year** | Year in which the outlet was established. | Numeric |
| 12 | **Item\_Outlet\_Sales** | The target variable representing the sales amount of the item in the outlet. | Numeric |

## Data Cleaning Steps

### Handling Missing Values

The dataset contains a notable number of missing values, particularly in the columns ‘Outlet\_Size’ (28.29%) and ‘Item\_Weight’ (17.17%). These missing values can significantly affect the performance of the model. Therefore, appropriate methods were applied to address and treat these gaps in the data.



* + 1. **Item Weight**

To tackle the issue of missing Item\_Weight values in our dataset, we implemented a systematic approach designed to enhance data completeness and ensure the robustness of our model. Here’s a detailed explanation of the steps taken:

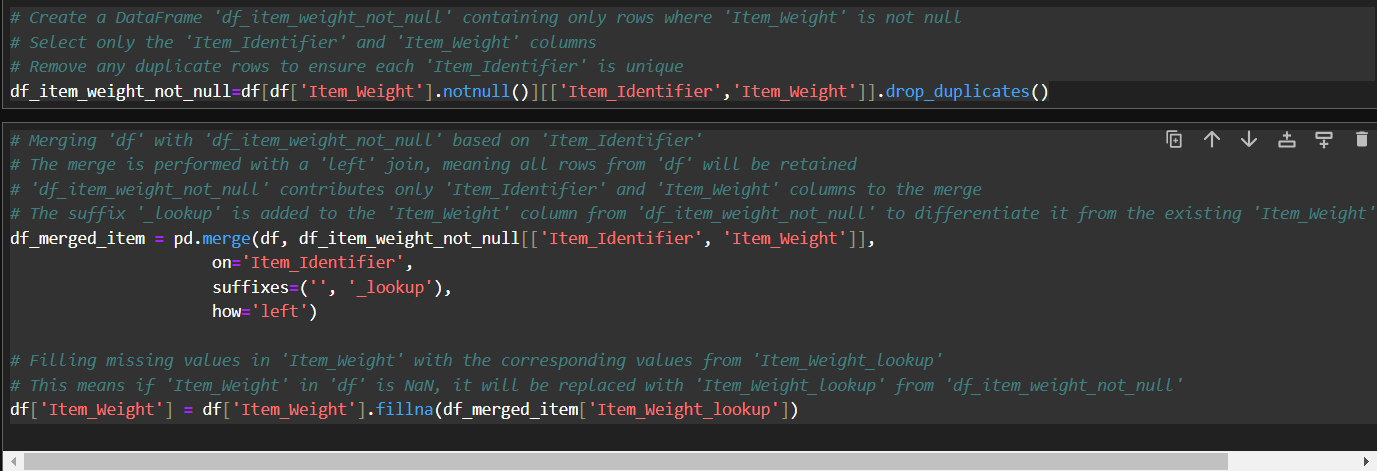
**Identify and Extract Non-Null Weights**:

First, we focused on the rows in the dataset where Item\_Weight values were already present (i.e., not missing). We created a subset of the data containing only these rows and selected two key columns: Item\_Identifier and Item\_Weight. This subset was then cleaned to remove any duplicate entries, ensuring that each Item\_Identifier was unique. This clean subset acts as a reference for known Item\_Weight values.

**Merge Data to Align Weights**:

Next, we combined this clean subset with the original dataset. The purpose of this merge was to align and integrate the known Item\_Weight values with the original dataset. During the merge, we included an additional column, Item\_Weight\_lookup, which holds the Item\_Weight values from our clean subset. This additional column allows us to differentiate between the original and the newly added weights.

**Fill Missing Values**: With the merged dataset, we updated the original Item\_Weight column by filling in any missing values. Specifically, for each entry in the original dataset where Item\_Weight was missing, we replaced it with the corresponding value from the Item\_Weight\_lookup column. This step ensures that gaps in the Item\_Weight data are filled with the best available information, derived from our reference subset.



* + 1. **Outlet Size**

To address missing values in the Outlet\_Size column, we took the following steps:

**Calculate Total Sales by Outlet**: We began by grouping the dataset based on Outlet\_Identifier and calculating the total sales (Item\_Outlet\_Sales) for each outlet. This aggregation was rounded to two decimal places to ensure accuracy.

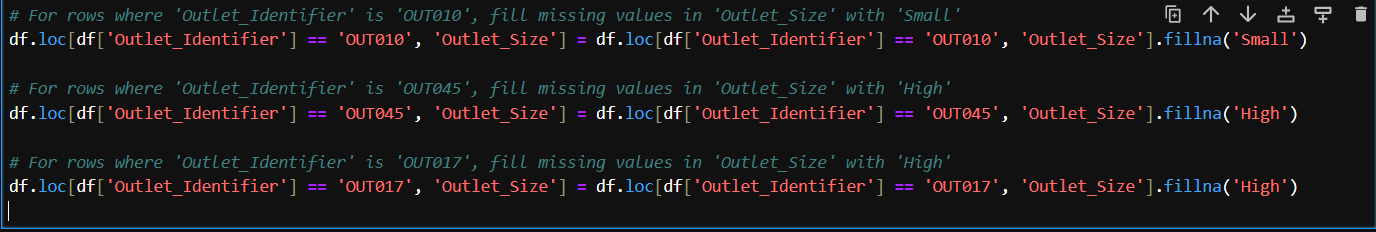
**Identify Missing Outlet\_Size**: Next, we identified which Outlet\_Identifier entries had missing values in the Outlet\_Size column. We extracted the unique Outlet\_Identifier values where Outlet\_Size was not provided, which revealed that three specific outlets had missing size information.

**Assign Outlet\_Size Based on Sales**: To determine appropriate Outlet\_Size values, we examined the total sales figures for each outlet with missing Outlet\_Size. The assignments were made as follows:

* **Outlet OUT010**: With total sales of 188,340.17, this outlet had relatively low sales compared to others. Thus, we assigned Outlet\_Size as 'Small'.
* **Outlets OUT045 and OUT017**: Both of these outlets had total sales exceeding 2 million. Given their high sales figures, we assigned Outlet\_Size as 'High' for these outlets.

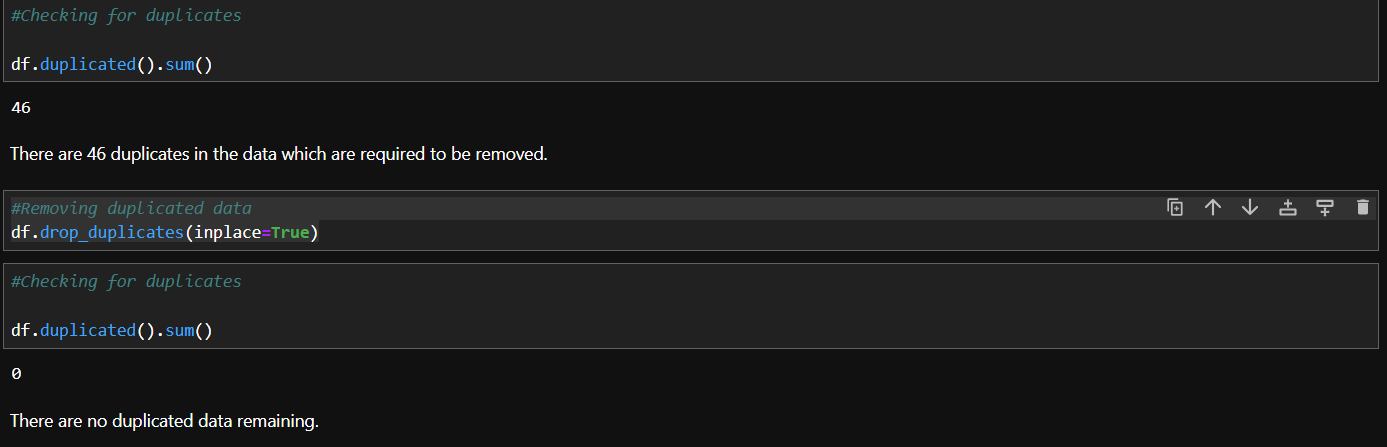
**Update Missing Values**: We updated the Outlet\_Size column for the identified outlets by filling in the missing values with the assigned sizes based on their sales figures. This ensured that each outlet now has a complete and appropriate Outlet\_Size designation.





### Handling Duplicates

To ensure the accuracy and integrity of the dataset, we removed duplicate entries by applying the following step:We used the drop\_duplicates() function to eliminate any repeated rows from the dataset. This operation was performed in-place, meaning that the original DataFrame was updated directly to remove duplicates.



### Handling Data Inconsistencies

* + 1. **Item\_Fat\_Content**

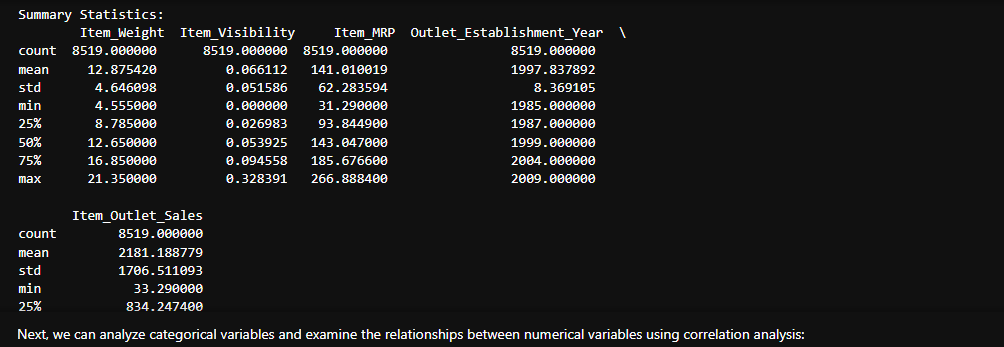
To maintain consistency and ensure uniformity in the dataset, we standardized the values in the Item\_Fat\_Content column by replacing variations with standardized terms. We updated the Item\_Fat\_Content column to unify the representation of fat content information. Specifically:

* 'LF' and 'low fat' were replaced with 'Low Fat'.
* 'reg' was replaced with 'Regular'.

## Exploratory Data Analysis (EDA)

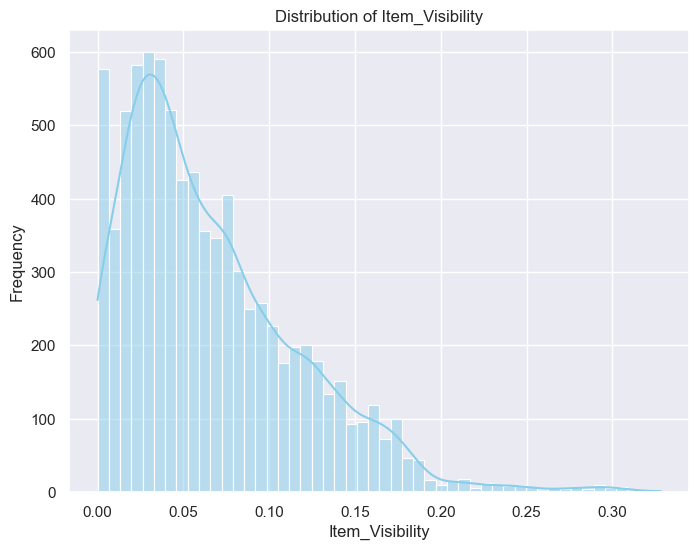
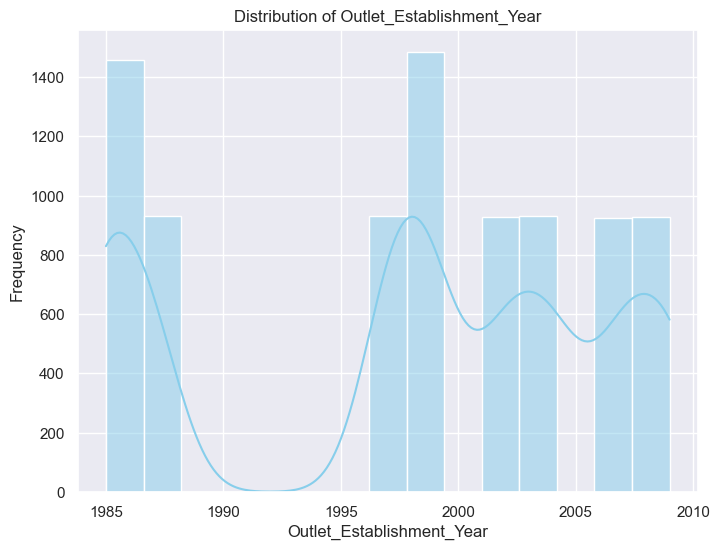
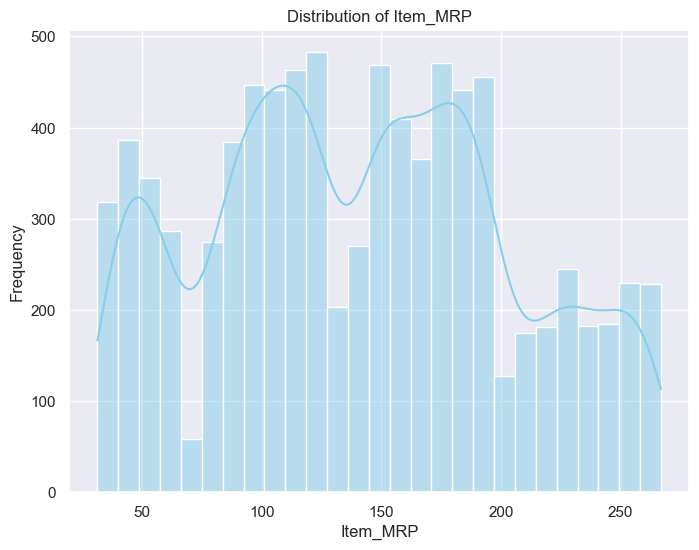
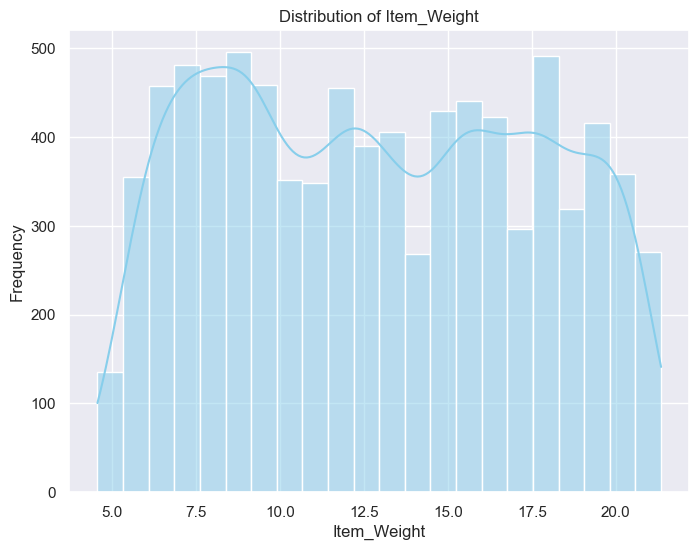
### Summary Statistics

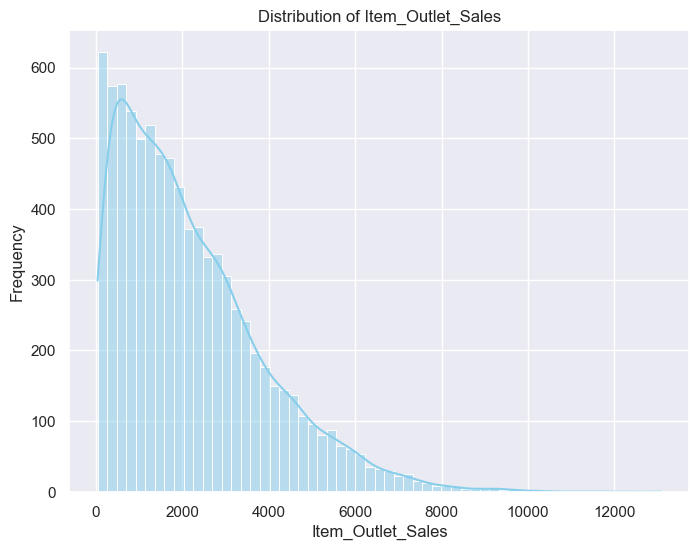
We compute and review summary statistics for numerical columns.



### Histogram

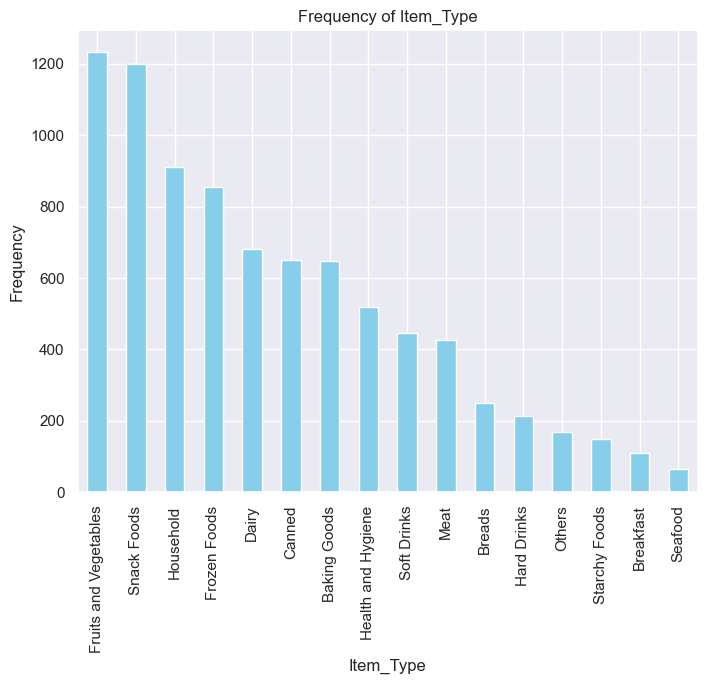
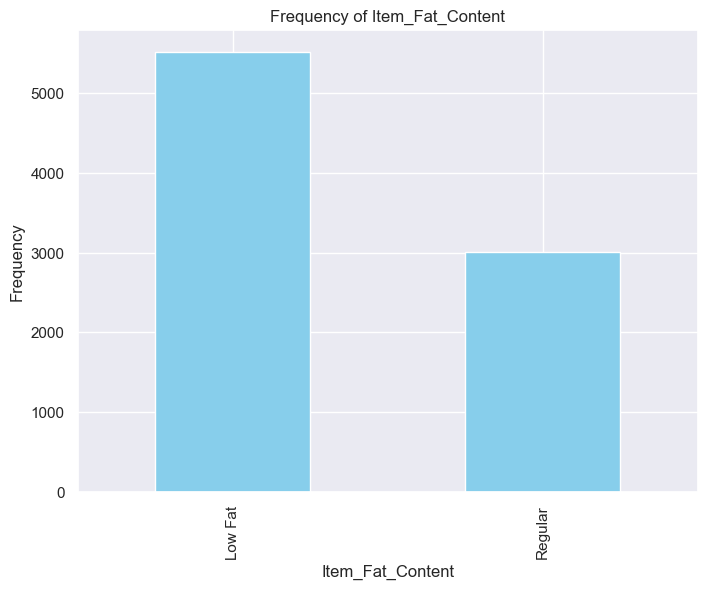
We plot histograms to visualize the distributionn of numerical variables.

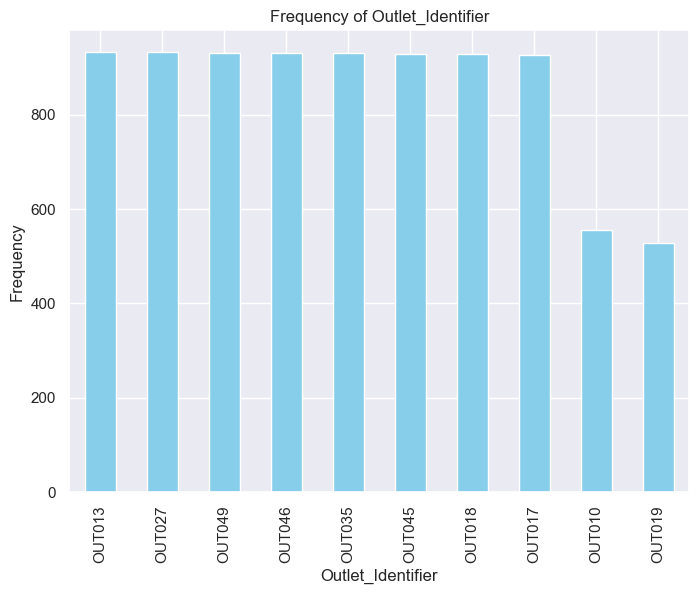




### Bar Plots for Categorical Variables

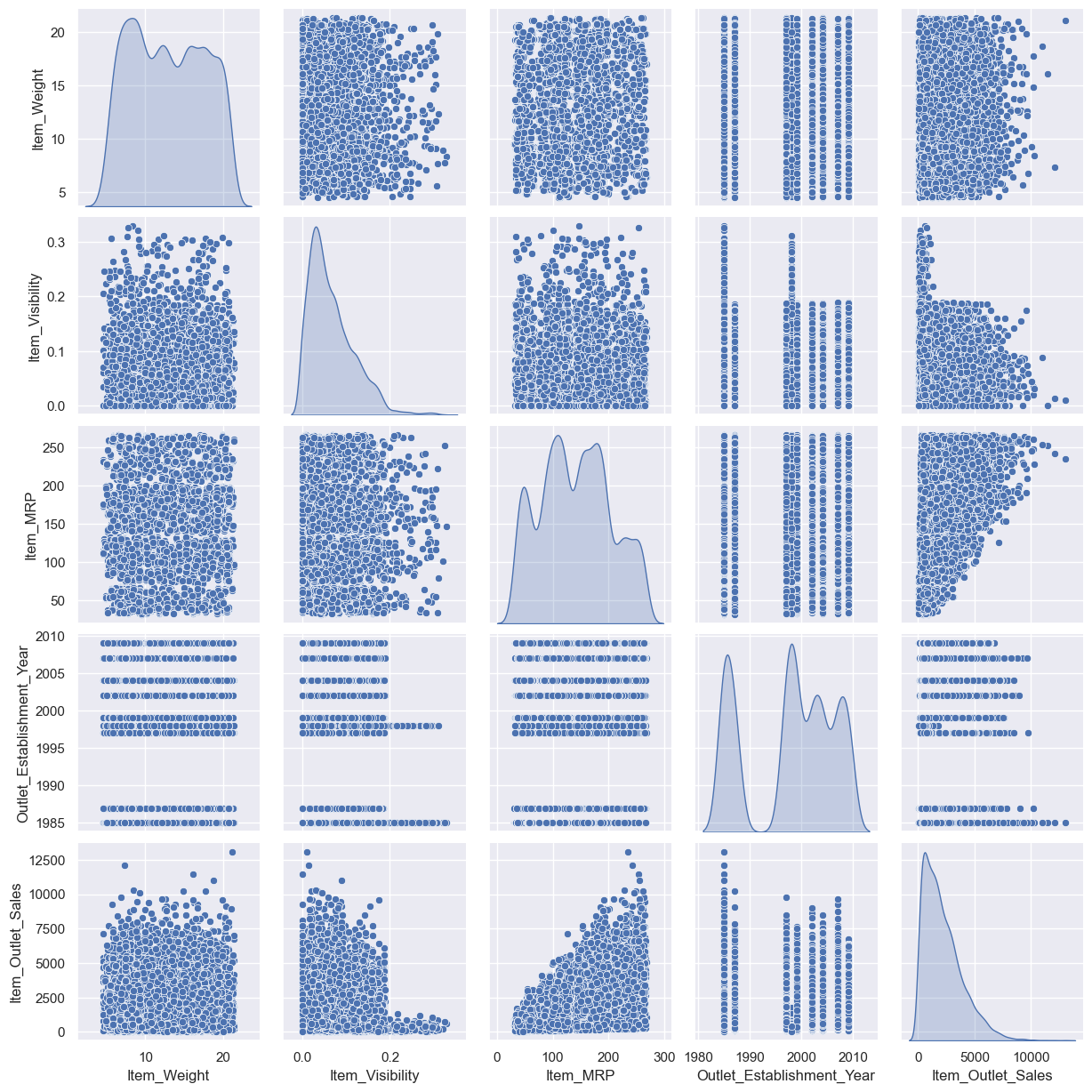
We plot bar charts to visualize the frequency of categorical variable values.





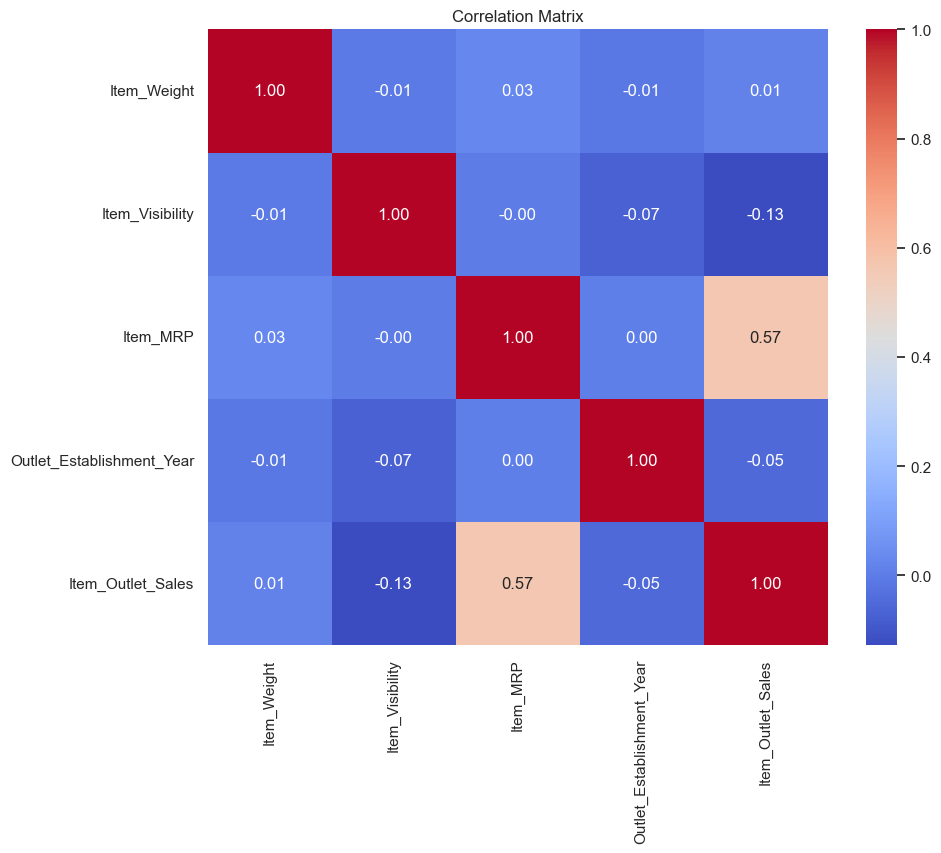
### Relationship Between Numerical Variables

We use pair plots to visualize relationships between numerical variables.



### Correlation Matrix Heatmap

We compute and visualize the correlation matrix to understand the relationships between numerical variables.



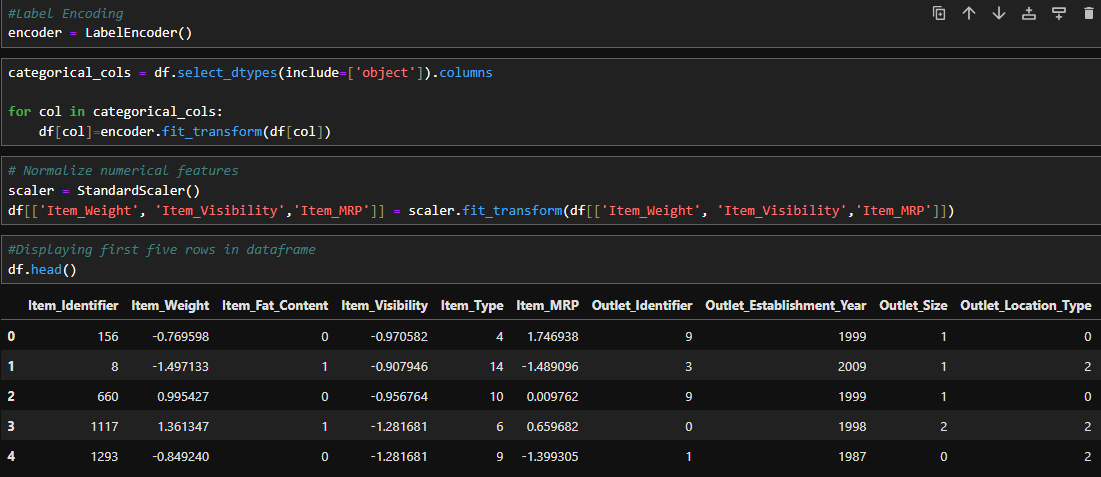
## Feature Engineering

### Label Encoding

Categorical variables are encoded numerically for modeling. Categorical features are transformed into numerical format using LabelEncoder. This technique converts each category into a unique integer value.

### Feature Scaling

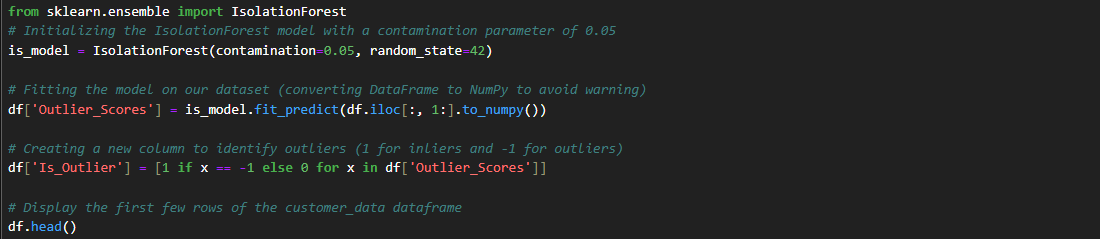
Numeric features (Item\_Weight, Item\_Visibility, Item\_MRP) are scaled using StandardScaler to normalize the data. This helps in bringing all features to the same scale, improving the performance of the machine learning algorithms.



### Outlier Detection Using Isolation Forest

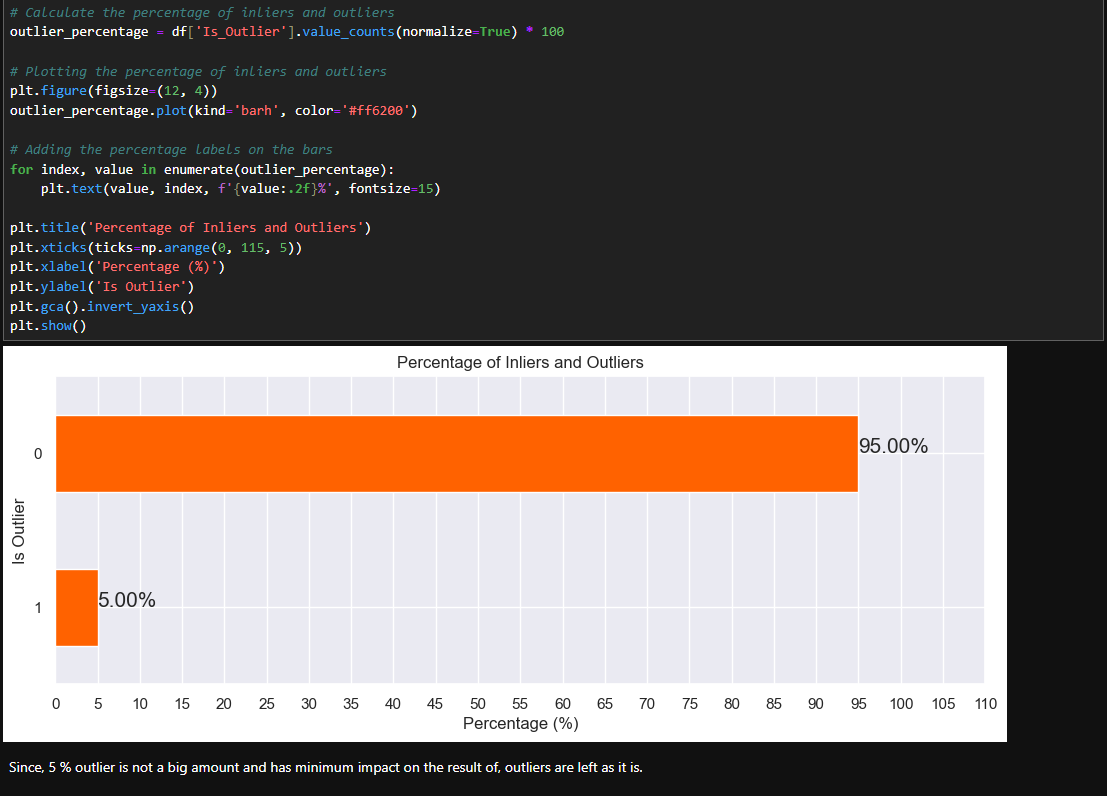
**Initialization and Fitting Isolation Forest:**

The Isolation Forest algorithm detects anomalies (outliers) by isolating observations. We use a contamination parameter to specify the expected proportion of outliers.



**Percentage of Inliers and Outliers:**

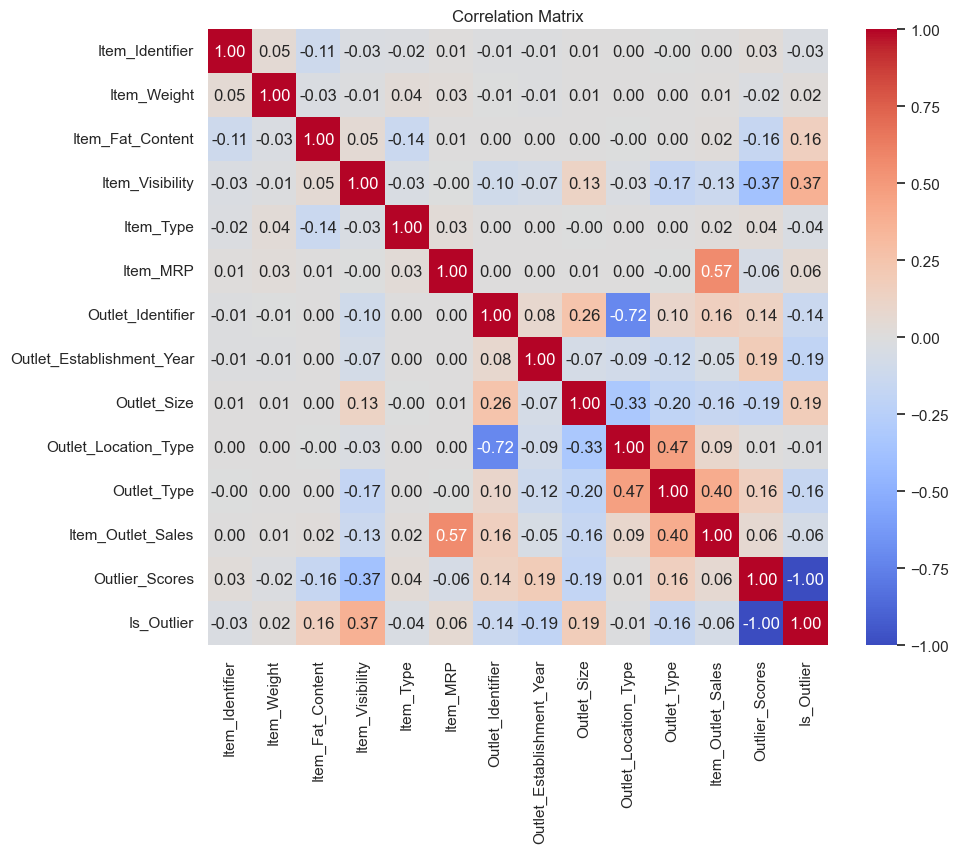
We calculate and visualize the percentage of inliers and outliers using a horizontal bar plot.



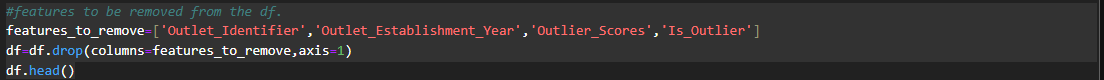
## Feature Selection

### Heatmap of Correlation Matrix:

We use a heatmap to visualize the correlation between numerical features. Correlation measures the strength and direction of the linear relationship between pairs of features. The **correlation matrix** heatmap shows the relationships between numerical features, which can help in understanding feature dependencies and multicollinearity.



### Feature Selection



### Removal of Features

Following features were removed due to various reasons explained below.

* **Outlet\_Identifier:**Removed due to high correlation with Outlet\_Location\_Type and similar role as other features.
* **Outlet\_Establishment\_Year:**Removed as it was deemed to have minimal impact on sales predictions.
* **Outlier Detection Features:**Features related to outlier detection (Outlier\_Scores, Is\_Outlier) were excluded as they were used for initial analysis and do not contribute to the final model.

### Selection of Features

The final feature set includes relevant features that significantly impact the sales prediction, such as ‘Item\_Fat\_Content’, ‘Item\_Type’, ‘Item\_Weight’, ‘Item\_Visibility’, ‘Item\_MRP’, ‘Outlet\_Location\_Type’, and ‘Outlet\_Type’.

## Model Development

### Data Preparation

After removing the columns deemed less relevant (Outlet\_Identifier and Outlet\_Establishment\_Year) and handling outliers, we prepared the dataset for model training. The target variable is Item\_Outlet\_Sales, and the remaining columns serve as features.

### Data Splitting:

The dataset is split into training and testing sets to evaluate model performance. The split is set to 80% for training and 20% for testing.

The shapes of the training and testing datasets are:

* **Full Dataset:** (8519, 9)
* **Training Set:** (6815, 9)
* **Testing Set:** (1704, 9)

### Linear Regression Model

Linear Regression is a fundamental and widely-used statistical technique for modeling the relationship between a dependent variable and one or more independent variables. In the context of this project, Linear Regression is employed to predict Item\_Outlet\_Sales based on various features related to items and outlets.

### Decision Tree Regression

We train a Decision Tree Regressor with specified hyperparameters to predict Item\_Outlet\_Sales. We evaluate the Decision Tree model using the R² score and perform cross-validation to assess its performance. We perform Grid Search to find the optimal hyperparameters for the Decision Tree Regressor.

### Random Forest Regression

We train a Random Forest Regressor to predict Item\_Outlet\_Sales. We evaluate the Random Forest model using the R² score and cross-validation.

### Model Evaluation

* **Linear Regression** was trained and evaluated, providing a baseline with an R² score of 0.53 and RMSE of 1187.69.
* **Decision Tree Regressor** was trained with a cross-validated RMSE of 1127.77. Hyperparameter tuning improved performance slightly with an R² score of 0.54.
* **Random Forest Regressor** showed the best performance with an R² score of 0.61 and cross-validation RMSE of 1081.94.

### Choice of Model

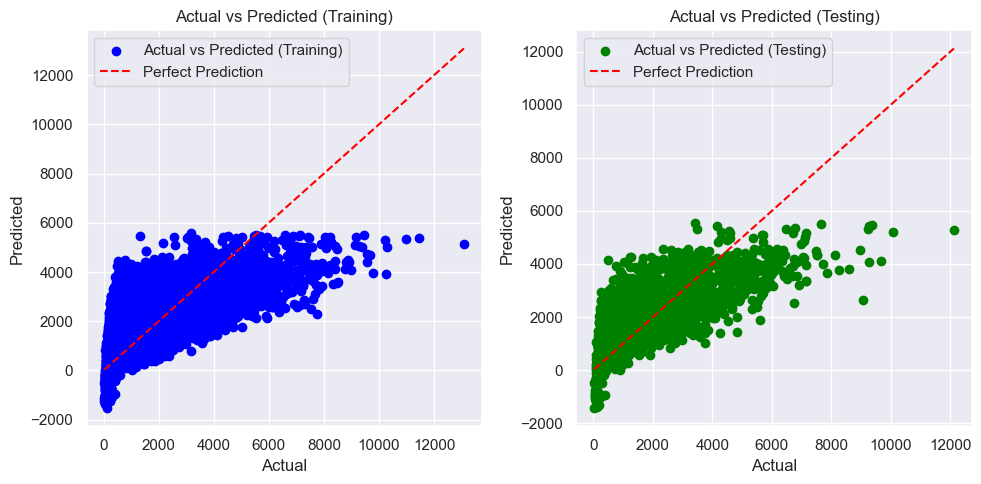
**Linear Regression** was chosen due to its simplicity and interpretability for predicting sales. Alternative models such as Random Forest Regressor and Gradient Boosting Machines were also considered for potential improvements.

### Training the Model

The dataset was split into training and testing sets. The Linear Regression model was trained on the training set. We train a Linear Regression model to predict Item\_Outlet\_Sales based on the features.

### Visualization of Actual vs Predicted Values:

We plot the actual versus predicted values for both training and testing datasets.



## Deployment

### Model Preprocessing, Training, and Saving

The trained Linear Regression model and preprocessing pipeline are saved using pickle. This ensures that the model and preprocessing steps can be reused for future predictions without retraining.

### Preprocessing Pipeline

To ensure a consistent approach to feature processing and model training, we set up a preprocessing pipeline that handles both numeric and categorical features. This pipeline includes the following steps:

**1. Numeric Features:**

* **Imputation:** Missing values are handled using median imputation to fill in missing data with the median value of each feature.
* **Scaling:** Standard scaling is applied to normalize numeric features, ensuring they contribute equally to the model by scaling them to have a mean of 0 and a standard deviation of 1.

**2. Categorical Features:**

* **Imputation:** Missing values are handled using the most frequent value for each feature.
  + **One-Hot Encoding**: Categorical features are encoded using label encoding.

### Preprocessing the Data

The preprocessing steps are applied to the dataset to transform the raw feature data into a format suitable for training the model.

### Splitting the Data

The preprocessed data is then split into training and testing sets. This division ensures that the model is evaluated on unseen data, providing a measure of its generalizability.

### Training the Linear Regression Model

With the preprocessed data, we train a Linear Regression model. This model will be used to predict Item\_Outlet\_Sales based on the transformed features.

### Saving the Model and Preprocessor

To facilitate future use and deployment, we save both the trained model and the preprocessing pipeline using Python's pickle module. This step allows us to load and use the model and preprocessing steps without retraining. Used pickle to save the trained model and preprocessing pipeline for future use.