BIG MART SALES PREDICTION AND ANALYSIS USING MACHINE LEARNING

Project Report submitted in partial fulfillment of the requirements

for the degree of **Bachelor of Technology**

in

Computer Engineering

By

Siona Panda(B518048)
Chiranjibi Tajan(B516015)

Under the guidance of

Prof. Tushar Ranjan Sahoo

Department of Computer Science & Engineering
International Institute of Information
Technology Bhubaneswar Bhubaneswar,
Odisha, 751003, India
May 2022

With the blessings of the almighty. To
my beloved parents, family members,
and my mother

Computer Engineering
International Institute of Information Technology, Bhubaneswar
Bhubaneswar-751003, Odisha, India.

Supervisor's Certificate

This is to certify that the work in the project report entitled "BIG MART SALES PREDICTION AND ANALYSIS USING MACHINE LEARNING" by Chiranjibi Tajan & Siona Panda, bearing roll numbers B516015 & B518048 is a record of an original research work/work carried out by them under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Engineering. Neither this thesis/report nor any part of it has been submitted for any degree or academic award elsewhere.

Date: 13th May 2022
Place: Bhubaneswar

Prof. Tushar Ranjan Sahoo
Professor
Department of Computer Science and Engineering
International Institute of Information Technology
Bhubaneswar, Odisha-751003

Computer Engineering
International Institute of Information Technology, Bhubaneswar
Bhubaneswar-751003, Odisha, India.

Certificate of Examination

Name: Chiranjibi Tajan(B516015)
Siona Panda(B518048)

Title of Report: **Big Mart Sales Prediction & Analysis using Machine Learning**

We, the below signed, after checking the report mentioned above and the official record book (s) of the student, hereby state our approval of the report submitted in partial fulfillment of the requirements of the degree of Bachelor of Technology in Computer Engineering at International Institute of Information Technology, Bhubaneswar. We are satisfied with the volume, quality, correctness, and originality of the work.

Prof. Tushar Ranjan Sahoo
(Guide)

Prof. RC Balabantaray
(Dean Academic)

External Examiner

Computer Engineering
International Institute of Information Technology, Bhubaneswar
Bhubaneswar-751003, Odisha, India.

Declaration of Originality

We, Chiranjibi Tajan & Siona Panda, bearing roll numbers B516015 & B518048, hereby declare that the report entitled "Big Mart Sales Prediction & Analysis using Machine Learning", presents our original work carried out as bachelor's students of **International Institute of Information Technology, Bhubaneswar**, and to the best of our knowledge, contains no material previously published or written by another person, nor any material presented by us for the award of any degree or diploma of **International Institute of Information Technology, Bhubaneswar**, or any other institution. Any contributions made to this work by others, with whom we have worked at **International Institute of Information Technology, Bhubaneswar**, or elsewhere, are explicitly acknowledged in the report. Works of other authors cited in this dissertation have been duly acknowledged under the sections "Reference" or "Bibliography". We have also submitted our original research records to the scrutiny committee for evaluation of our report.

We are fully aware that in case of any non-compliance detected in the future, the Senate of **International Institute of Information Technology, Bhubaneswar** may withdraw the degree awarded to us based on the present dissertation.

Date:13th May 2022
Place: Bhubaneswar

Chiranjibi Tajan(B516015)
Siona Panda(B518048)
Department of Computer Science and Engineering
International Institute of Information Technology,
Bhubaneswar, Odisha, India

Acknowledgment

We would like to share our sincere gratitude to everyone who has helped us in the making of this project. The completion of this project would not have been possible without the continuous supervision, invaluable tutelage, and treasured support of Prof Tushar Ranjan Sahoo, our esteemed guide during this Project. We would like to extend our sincere gratitude to the faculties of the Computer Science & Engineering Department for their mentorship and guidance, from time to time. Our heartiest appreciation also goes to our classmates and parents who have helped us by critiquing our work and extending their support in times of need. Whilst it would be simple to name them all, it would not be easy to thank them enough.

Chiranjibi Tajan (ID: B516015)
Siona Panda (ID: B518048)

ABSTRACT

Sales forecasting is vital in marketing, retailing, wholesaling, and manufacturing, and it is done differently in different firms. Many Big Marts do not have a superior yearly sales forecasting method nowadays. This is due to a lack of sales estimation skills, resources, and understanding. As a result, there is a pressing need in the contemporary marketplace for the development of a smart prediction system that is quick, adaptable, and accurate. This enables Big Marts to properly allocate cash, estimate realistic sales revenues, and build a better plan for potentially expanding the business, as well as better tactics that will lead to future growth.

Shopping marts and superstores can keep track of individual items' sales data to predict future demand and adjust stock management. This assists them in comprehending the characteristics of products and locations that are critical to improving sales. While the relationship with customers is inextricably linked to increased revenue, Customer Relationship Management (CRM) is the process of categorizing clients into distinct groups so that marketing personnel may employ customized marketing to retain customers (CRM). An integrated and balanced approach to technology, processes, and people is required for a successful CRM implementation.

Many people purchase things on a regular basis when they visit a Big Mart. Customers must move from shelf to shelf in order to find the product of their choice, which takes time. By evaluating client invoices and informing the Big Mart about customer purchasing patterns, this analysis will help to reduce consumer time consumption. Client buying habits can be studied and relevant information collected using Data Mining and association rules, allowing for the determination of customer purchase patterns and the identification of product associations. As a result, shopping can be simplified by grouping related items and the most often purchased things on the same shelf.

Forecasts are critical, particularly in evaluating the accuracy of projecting future demand for items and inventory supply levels, which has been critical, particularly

in supermarkets (Big Marts). If items are not readily available or if supply exceeds demand, total profit may be jeopardized. As a result, product sales forecasting could be essential in reducing losses.

The information gathered can be utilized to forecast future sales volume. This is primarily intended to support future purchases and provide a more precise forecast of sales.

Keywords: Machine Learning, Sales Forecasting, Random Forest, Regression, Xgboost.

Table of Contents

List of Figures

CHAPTER

Introduction

An Overview of Sales prediction using ML
Literature Review
Motivation and Objectives
Project Organization

Chapter 1

Introduction

1.1 Background

In today's modern world, large shopping centers such as malls and marts keep track of sales of items or products, as well as their many dependent and independent aspects, as a key step in forecasting future demand and inventory management. The dataset, which is made up of several dependent and independent variables, is a composite of item attributes, customer data, and data linked to inventory management in a data warehouse. Following that, the data is improved to obtain accurate predictions and to acquire new and intriguing outcomes that provide new insight on our understanding of the task's data.

Different Machine Learning techniques like Linear Regression, Random Forest, Decision Tree, Ridge Regression, and XGBoost are used to gauge deal volume in various Big Mart across different regions to handle the issue of deals expectation of goods based on client's future requests. Deals predict the outcome since they are based on the type of store, the population surrounding the business, and the location in which the store is located, which could be in an urban zone or a country.

Sales are affected by the population statistics surrounding the business, as well as the store's capacity and many other factors. Sales predictions are important in a

retail center since every firm has strong demand. Stronger forecasting is usually beneficial in establishing and improving corporate market strategies, which also helps to raise market awareness.

1.2 An Overview of Sales Prediction Using ML

The amount of data available is growing every day, and such a large volume of unprocessed data must be analyzed carefully in order to produce highly relevant and finely pure gradient findings that meet current standards. Machine Learning (ML) is evolving at a rapid speed, just like Artificial Intelligence (AI) has over the last two decades. ML is an essential aspect of the IT industry and, as a result, a pretty central, albeit often concealed, part of our lives. Because data is very useful in current aspects, the analysis and interpretation of data to produce effective results will expand as technology advances.

In machine learning, one interacts with both supervised and unsupervised tasks, and a classification problem typically serves as a knowledge discovery resource. It creates resources and applies regression to make precise future predictions, with the primary focus on making a system self-efficient, allowing it to perform computations and analyses in order to produce highly accurate and precise findings. Data can be transformed into knowledge utilizing statistical and probabilistic algorithms. As a conceptual key, sampling distributions are used in statistical inferencing.

1.3 Literature Review

Sales forecasting is significant in many sectors, including economics, which describes market patterns and forecasts the prospective region of national market commodities.

The two most fundamental notions of sellers and consumers are supply and demand. For firms to be able to develop future sales plans, it is critical to accurately predict demand. Sales Prediction is the process of anticipating sales for various Big Mart shops in order to adjust the company's future strategies depending on the expected sales. They suggest a method for Big Mart companies to forecast demand.

For prediction analysis, a forecast for Big Mart sales based on Random Forests and Linear Regression was applied, which yielded lower accuracy. To address this, we can utilize the XG boost Algorithm, which will provide greater accuracy and efficiency.

Every business wants to be in the competition in today's market. An excellent concept for a corporation to examine product sales is to use sales forecast.

Nikita Malik [1] has written about using machine learning to anticipate sales. She applied a machine learning method to her work (linear regression, Random Forest etc.). She examined a few things and discovered a link between the product and the retailer. The accuracy ranges between 70 and 80 percent

Aditi Narkhede [2] gathered BigMart data and applied a machine learning method to calculate the RMSE number. She has done some calculations to determine the RMSE value, and it is quite simple to use. She made calculations appear simple.

Rajendra Pamula [3] has discussed flow chart diagrams to help us better understand things. He also used machine learning and data mining in this instance.

The precision ranges from 50 to 60%. To obtain the output, he employed some complicated calculations that are difficult to comprehend.

For sale prediction, **Saju Mohanan [4]** employed data mining techniques and a machine learning system. For prediction, he employed a decision tree and a generalized linear model.
The model's accuracy ranges from 60 to 70 percent. He has also drawn system architecture to simplify things, however, the product is extremely complex.

Pavan Chatradi [5] has written on utilizing machine learning and the xgboost approach to anticipate sales. He has gone through the phases of data cleansing, data transformation, and data reduction. This approach has an accuracy of more than 80%. However, the approach and result displayed are complicated.

The steps involved in predicting are shown below:-

Dataset –> Data Exploration –> Data Cleaning –> Feature
Engineering –> Model Building –> Model Testing –> Result

1.4 Motivation and Objectives

1.4.1 Motivation of the Project

Consider the following scenario: you wish to open an inventory for selling everyday products such as ration items, and you have a central customer specialized location for service. Once all of the information about the types and quantities of foodstuffs, ration items, and everyday products used by the residents of a given area has been gathered.

The information can then be given to specific shop owners or big-box store salespeople who can make the products or services available. So that they may

maintain their inventory in accordance with the data, minimizing the excess products in their inventory that remain there until the distributor returns them.

As a result, new shop owners and existing shop owners who wish to maintain their inventory fresh with new products ready to sell must rely on the dependability of our processed data.

The thought of establishing superior Business Strategies sparked motivation.

The nature of data must be understood once the dataset has been collected and integrated.

To do so, calculate the mathematical statistics (mean, median, range, standard deviation, etc.) for each attribute to identify correlations, general trends, and outliers. This allows you to narrow down the elements that influence product sales. Visualization approaches are also employed to aid the process. Missing, duplicate, and inconsistent values are checked for and repaired after the preliminary analysis. Filters are applied to features that can be grouped together or are not required. Dimension analysis improves the feature selection process even further.

If this phase is skipped, a large number of unneeded features will be analyzed in subsequent steps, potentially resulting in a significant difference in the final result. This facilitates data manipulation and makes the dataset fault-tolerant.

The goal of this thesis is to develop a more accurate predictive model for Bigmart outlet item sales forecast. Using the above objectives, this study aims to assess the performance of predictive models constructed using machine learning and deep learning techniques.

i. Conduct literature research and content analysis based on big mart forecasts and the application of machine learning and deep learning for prediction.

ii. Use the dataset to perform data preparation.

iii. Using multiple learning techniques, create predictive models with the dataset.

iv. Evaluate the models' performance and choose the most accurate one based on the performance metrics.

1.4.2 Objective of the Project

- Existing shop owners who want to keep their inventory fresh.
- New business shop owners who want to sell from Brand to Consumer.
- Provide information to anyone looking for average consumption data for a specific region.
- Another goal is to use the XG Boost Regressor to come up with the best model that is more efficient and offers fast and accurate results.
- To determine crucial aspects that can boost sales and what modifications to the product or store's attributes could be made.
- The main goal of this thesis is to find out which machine learning model performs best when estimating sales for a specific collection of products and retailers.
- As a result, the purpose of this thesis is to improve forecasting models in order to reduce waste and boost product availability.
- The goal is to predict the pattern of sales and the quantities of products to be sold based on some key characteristics gleaned from the raw data.
- This will assist BigMart in increasing sales by learning how to better organise products within stores.

1.5 Project Organization

- Chapter 1 contains an Abstract & a Formal Introduction to the Project
- Chapter 2 contains the System architecture and Approach to the Problem statement
- Chapter 3 contains Test Results analysis and Evaluation of the System
- Chapter 4 contains the Conclusion & Scope of future work
- Chapter 5 contains Bibliography & References

CHAPTER

Methodology

Problem Statement

Proposed Architecture

Approach Description

Methodology

Chapter 2

Methodology

2.1 Problem Statement

By studying Big Mart sales, we can figure out what role certain qualities of an item play and how they affect its sales. To assist BigMart in achieving this goal, a predictive model can be constructed to determine the important elements that can enhance sales in each store and what adjustments to the product or store's attributes could be made.

The dataset was created from 2013 sales data for 1559 products across 10 retailers in various cities. The goal is to create a prediction model and determine the sales of each product at a specific store.

2.2 Proposed Architecture

According to the 2013 data collection, BigMart's data scientists gathered sales data from their 10 stores located in various regions, each of which has 1559 different products. It can be deduced from all of the data what role certain qualities of an item play and how they affect sales.

There might be numerous forms of underlying patterns in raw data, which can provide in-depth knowledge about the subject of interest and provide insights into the situation. However, data should be treated with caution since it may contain null values, redundant values, or various sorts of ambiguity, necessitating pre-processing.

As a result, the dataset should be investigated as thoroughly as feasible.

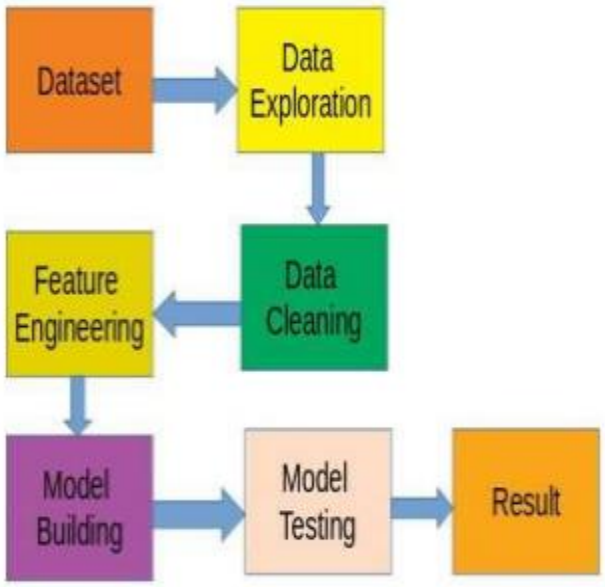


Figure 1 - Workflow diagram

1. Dataset Collection

In our research, we employed a dataset of wide market sales data, which has 12 features. These 12 criteria define the fundamental characteristics of the data being projected. Answer variables and predictors are two types of qualities. We will utilize a dataset that contains 8523 objects from various places and cities. Our dataset is primarily concerned with hypotheses at the store and product level. At the store level, attributes such as area, population density, store capability, and location have been added. Finally, the dataset is split into two parts: training and testing.

2. Data Exploration

This stage extracts valuable data information from the dataset. The inception year of the outlet varies from 1985 to 2009. These values may not be sufficient in this form. The collection contains 1559 individual items and 10 different outlets. We classify the data based on the hypothesis vs. available evidence, which shows that the size of the outlet property and the weight of the object are missing values, and the least value of the Object view is Zero, which is not practicable. There are 16 distinct values in the Item type attribute. Positively biased variable outlet sales have been seen. To account for skewness, a log is applied to the Answer Variable.

```
In [7]: df.head()
        #understanding rows and column
```

Out[7]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locatic
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	

```
In [7]: df.head()
        #understanding rows and column
```

Out[7]:

at_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

Fig 2 - head function Viewing first five datapoints

```
In [9]: df.dtypes
        #tells datatype of column convert data type
```

Out[9]:

```
Item_Identifier              object
Item_Weight                 float64
Item_Fat_Content             object
Item_Visibility             float64
Item_Type                    object
Item_MRP                    float64
Outlet_Identifier            object
Outlet_Establishment_Year     int64
Outlet_Size                  object
Outlet_Location_Type         object
Outlet_Type                  object
Item_Outlet_Sales           float64
dtype: object
```

Fig 3 - dtypes function to check datatype of attributes

```
In [10]: df.describe()
```

Out[10]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247400
50%	12.600000	0.053931	143.012800	1999.000000	1794.331000
75%	16.850000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

Fig 4 - describe function to view statistical data

3. Data Cleaning

The attributes Outlet size and Element weight was discovered to be empty in the preceding section. We replace missing values for outlet size with the mode value of that attribute, and we substitute mean values for missing values of that particular property of object weight. The missing attributes are numerical, and as the mean and mode replacement lowers, the correlation between the imputed attributes falls. We feel there is no relationship between the calculated and imputed attributes in our model.

4. Feature Engineering

Feature engineering is the process of transforming cleansed data into predictive models in order to better convey a problem. Some noise was discovered during the data examination. This noise is resolved in this phase, and the data is used to develop a suitable model. To make the model perform exactly and successfully, new features are added. For the model to operate better, a few created characteristics can be integrated. The feature engineering process transforms data into a format that algorithms can interpret.

5. Model building

After feature engineering, the data is used to apply numerous algorithms to provide correct results. A model is a set of techniques that makes detecting relationships between numerous datasets easier. By identifying precise data insights, an effective model may anticipate correct results.

2.3 Approach Description

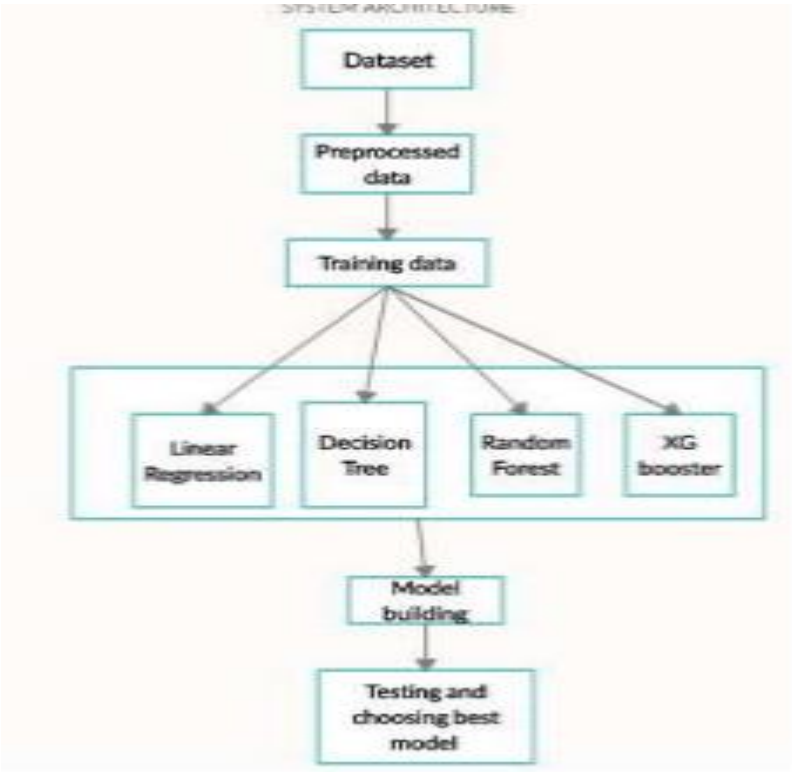


Fig 5 - Algorithms used in prediction

The algorithms used in this framework are:-

- **LINEAR REGRESSION:** The linear regression algorithm attempts to predict results by displaying a graph between an independent variable and a

dependent variable produced from the dataset. It's a type of statistical analysis used to create machine learning models.

Z = a + bE is the general equation for linear regression.

The dependent variable is Z, while the independent variable is E.

- **RIDGE REGRESSION:** When multicollinearity (highly correlated independent variables) influences outcomes, Ridge Regression is utilised. In multicollinearity, the least square estimates (OLS) are objective. Their variations are large and differ from the genuine value. When it comes to regression, a degree of bias might be used. Ridge regression eliminates standard errors in calculations.

- **RANDOM FOREST:** The Random Forest Algorithm is used to combine predictions from several decision trees into a single model. This algorithm creates a forest of decision trees by using a bagging process. It then combines the results of numerous decision trees to produce extremely precise predictions. Random forest generation is the first phase in the Random Forest algorithm. Random forest classifier-generated prediction is the second phase.

- **DECISION TREE:** Decision Trees are methods for supervised machine learning. The way decision trees function is that they break down the full data set represented in the tree's root node into smaller, more homogeneous units while simultaneously developing a decision tree in an incremental manner. Each inner node of a tree provides a donation for a check on one feature,

every branch represents the check's final result, and every leaf in the tree, also known as a terminal node, represents a choice in the regression scenario.

- **XG BOOSTER APPROACH:** The XG Boost algorithm was created with the help of decision trees and gradient boosting. This technique is based on the idea of boosting weaker algorithms within a gradient descent boosting framework. This method is extremely accurate, outperforming almost all other algorithms in terms of prediction accuracy. It is a modification of the Gradient Boosting method. Parallelized tree building and efficient handling of missing data are two features of XG Boost. Cross-validation functionality is built-in, and Cache Awareness and Tree Pruning are the other features.

2.4 Methodology

1. Dataset Description

For the website kaggle.com, We gathered the data from the internet. This work includes both a test and a training dataset, with the test dataset having 5000 data items and the training dataset having 8000 data items.

ATTRIBUTE	DESCRIPTION
Item Identifier	Product Unique Id
Item Weight	Product Weight
Item Fat Content	Fat Content of the Product

Item Visibility	Total percentage of allocation to this store
Item Type Category	Categorization of the product
Item MRP	Product Price
Outlet Identifier	Unique ID of the outlet
Outlet Establishment Year	Store Establishment Year
Outlet Size	Store size
Outlet Location Type	Type of located cities
Outlet Type	Supermarket sort
Item Outlet Sales	Product sales in particular area

Fig 6 - Attributes & their description

Name	Type	Subtype	Description	Segment	Expectation
Item_Identifier	Numeric	Discrete	Product Unique Id	Product	Low Impact
Item_Weight	Numeric	Continuous	Product Weight	Product	Medium Impact
Item_Fat_Content	Categorical	Ordinal	Fat Content of the Product	Product	Medium Impact
Item_Visibility	Numeric	Continuous	Total percentage of allocation to this store	Product	High Impact
Item_Type	Categorical	Nominal	Categorization of the product	Product	High Impact

Item_MRP	Numeric	Discrete	Product Price	Product	Medium Impact
Outlet_Identifier	Numeric	Discrete	Unique ID of the outlet	Store	Low Impact
Outlet_Establishment_Year	Numeric	Discrete	Store Establishment Year	Store	Low Impact
Outlet_Size	Categorical	Ordinal	Store size	Store	High Impact
Outlet_Location_Type	Categorical	Ordinal	Type of located cities	Store	High Impact
Outlet_Type	Categorical	Ordinal	Supermarket sort	Store	High Impact
Item_Outlet_Sales	Numeric	Discrete	Product sales in particular area	Product	Target

Fig 7 - Attributes & their impact on sales

```
[ ]  # getting some information about the dataset
     big_mart_data.info()
```

```
<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
```

Fig 8 - info function to check no. of data points & non-null count of columns

2. Data Preparation

Steps followed while data preparation/data-cleaning:

- ➢ Import the raw data
- ➢ Transform the relevant columns
- ➢ Filter the dataset
- ➢ Keep the relevant columns
- ➢ Drop duplicates
- ➢ Treat the missing values
- ➢ Save the cleaned data

We aim to obtain better accuracy by modifying and deploying multiple models when we evaluate a business problem. However, we will note that we will be battling to improve the model's accuracy after a certain point. To solve difficulties like these, data exploration comes into play. Because machine learning methods require numerical values as input, we use sklearn's pre-processing module's LabelEncoder and OneHotEncoder to convert categorical variables to numerical variables.

3. Implementation

Every model is trained using training data before being used to predict accuracy with test data, and this process is repeated until each subset has been tested once. However, the proposed approach outperforms previous models in terms of future sales projections. One crucial aspect is understanding how the many independent factors are related to the dependent variable or the projected future sales demand.

We can only determine how the dependent variables are affected by the independent variable after we know the correlation.

We need to know what the link is, such as the relationship between item MRP and outlet sales. The degree to which a certain property influences the final dependent variable's projected output, such as item MRP being significantly correlated with item Sales.

Other defined attributes are also discovered to play a role in deciding the item's Sales value to some extent. One of the most crucial item Sales deciding criteria is outlet size.

Platform and Language for Implementation

Python is a general-purpose, interpreted high-level language that is widely used nowadays to solve domain problems rather than deal with system complexities. It is also known as the "batteries included programming language."

It has multiple libraries for scientific purposes and queries, as well as several third-party libraries that help with problem-solving. Numpy, a Python library for scientific calculation, and Matplotlib, a Python library for 2D graphing, were utilized in this project. In addition, the Python Pandas tool has been used to perform data analysis.

Screenshots of Code

Big Mart Sales Prediction

Porblem Statement: The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. The aim is to build a predictive model and predict the sales of each product at a particular outlet.

Using this model, BigMart will try to understand the properties of products and outlets which play a key role in increasing sales.

Please note that the data may have missing values as some stores might not report all the data due to technical glitches. Hence, it will be required to treat them accordingly.

This project will proceed with the following structure:

1. **Hypothesis Generation** — attempt to better understand the problem by formulating pre-conceived hypothesis and possible factors relating to or affecting the outcome which are the sales
2. **Data Exploration** — exploratory data analysis on categorical and continuous variables and formulating inferences from them
3. **Data Cleaning** — impute missing data and vet for any outliers
4. **Feature Engineering** — one hot encode categorical features into new features with binary values for machine learning algorithm
5. **Model Building** — build predictive machine learning model

Hypothesis Generation

A crucial and mostly underated process in the data analysis process whereby we attempt to better understand the problem by formulating pre-conceived hypothesis and possible factors relating to or affecting the outcome before even looking at the data.

Here are some examples:

The Hypotheses

Store-Level

1. Stores located in cities or urban areas may generate higher sales due to **density of population being high**
2. Cities and urban areas may have populations in the **higher income bracket**, thus contributing to higher sales
3. **Larger stores** should technically generate higher sales
4. **Absence of other competitor stores** should allow for higher sales
5. **Effective marketing** should drive higher sales
6. **Good customer service** should yield higher sales

Product-Level

1. Brands affect sales; **branded products** yield higher sales
2. **Effective product marketing, popular products** will increase sales especially through word of mouth
3. **Daily essentials products or products with high utility** will generate more sales
4. **Products on offer or sale** will yield higher sales due to hype and demand

Data Exploration

Exploratory data analysis will be conducted to obtain some inferences about the dataset and maybe also confirm some of our hypothesis from the previous section.

Importing Important Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings # Ignores any warning
warnings.filterwarnings("ignore")

train = pd.read_csv("data/Train.csv")
test = pd.read csv("data/Test.csv")
```

In [2]:
```
train.head()
```

Out[2]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	

In [2]:
```
train.head()
```

Out[2]:

_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

In [3]:
```
test.head()
```

Out[3]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	NaN	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	

In [3]:
```
test.head()
```

Out[3]:

Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	Tier 1	Supermarket Type1
8.300	reg	0.038428	Dairy	87.3198	OUT017	2007	NaN	Tier 2	Supermarket Type1
14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	Tier 3	Grocery Store
7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	Tier 2	Supermarket Type1
NaN	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	Tier 3	Supermarket Type3

In [4]:
```
# shape method gives the shape of dataframe ,as we see 8523 rows with 12 columns.
print ("shape of dataset:")
train.shape
```

```
shape of dataset:
```

Out[4]: (8523, 12)

In [5]:
```
# size method gives the size of dataframe (like rows*columns)
print ("size of dataset:")
train.size
```

```
size of dataset:
```

Out[5]: 102276

In [6]:
```
# to see which column have null values use .isnull() ,as we can see two columns have lots of missing values.
train.isnull().sum()
```

Out[6]:
```
Item_Identifier               0
Item_Weight                1463
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                2410
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
dtype: int64
```

In [7]:
```
#Filter categorical variables
categorical_columns = [x for x in train.dtypes.index if train.dtypes[x]=='object']
#Exclude ID cols and source:
categorical_columns = [x for x in categorical_columns if x not in ['Item_Identifier','Outlet_Identifier']]
#Print frequency of categories
for col in categorical_columns:
    print ('\nFrequency of Categories for varible %s'%col)
    print (train[col].value_counts())
```

```
Frequency of Categories for varible Item_Fat_Content
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64

Frequency of Categories for varible Item_Type
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

Frequency of Categories for varible Outlet_Size
Medium    2793
Small     2388
High       932
Name: Outlet_Size, dtype: int64

Frequency of Categories for varible Outlet_Location_Type
Tier 3    3350
Tier 2    2785
Tier 1    2388
Name: Outlet_Location_Type, dtype: int64

Frequency of Categories for varible Outlet_Type
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
Name: Outlet_Type, dtype: int64
```

In [8]:
```
# Check for null values
train.isnull().sum().sort_values(ascending = False)
```

Out[8]:
```
Outlet_Size                2410
Item_Weight                1463
Item_Outlet_Sales             0
Outlet_Type                   0
Outlet_Location_Type          0
Outlet_Establishment_Year     0
Outlet_Identifier             0
Item_MRP                      0
Item_Type                     0
Item_Visibility               0
Item_Fat_Content              0
Item_Identifier               0
dtype: int64
```

In [9]:
```
train.info()
```

```
<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
```

In [10]:
```
train.describe(include='all')
```

Out[10]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_L
count	8523	7060.000000	8523	8523.000000	8523	8523.000000	8523	8523.000000	6113	
unique	1559	NaN	5	NaN	16	NaN	10	NaN	3	
top	FDG33	NaN	Low Fat	NaN	Fruits and Vegetables	NaN	OUT027	NaN	Medium	
freq	10	NaN	5089	NaN	1232	NaN	935	NaN	2793	
mean	NaN	12.857645	NaN	0.066132	NaN	140.992782	NaN	1997.831867	NaN	
std	NaN	4.643456	NaN	0.051598	NaN	62.275067	NaN	8.371760	NaN	
min	NaN	4.555000	NaN	0.000000	NaN	31.290000	NaN	1985.000000	NaN	
25%	NaN	8.773750	NaN	0.026989	NaN	93.826500	NaN	1987.000000	NaN	
50%	NaN	12.600000	NaN	0.053931	NaN	143.012800	NaN	1999.000000	NaN	
75%	NaN	16.850000	NaN	0.094585	NaN	185.643700	NaN	2004.000000	NaN	
max	NaN	21.350000	NaN	0.328391	NaN	266.888400	NaN	2009.000000	NaN	

In [10]:
```
train.describe(include='all')
```

Out[10]:

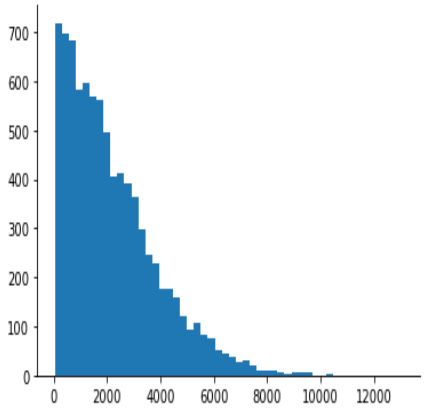
at_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
8523	8523.000000	8523	8523.000000	8523	8523.000000	6113	8523	8523	8523.000000
5	NaN	16	NaN	10	NaN	3	3	4	NaN
Low Fat	NaN	Fruits and Vegetables	NaN	OUT027	NaN	Medium	Tier 3	Supermarket Type1	NaN
5089	NaN	1232	NaN	935	NaN	2793	3350	5577	NaN
NaN	0.066132	NaN	140.992782	NaN	1997.831867	NaN	NaN	NaN	2181.288914
NaN	0.051598	NaN	62.275067	NaN	8.371760	NaN	NaN	NaN	1706.499616
NaN	0.000000	NaN	31.290000	NaN	1985.000000	NaN	NaN	NaN	33.290000
NaN	0.026989	NaN	93.826500	NaN	1987.000000	NaN	NaN	NaN	834.247400
NaN	0.053931	NaN	143.012800	NaN	1999.000000	NaN	NaN	NaN	1794.331000
NaN	0.094585	NaN	185.643700	NaN	2004.000000	NaN	NaN	NaN	3101.296400
NaN	0.328391	NaN	266.888400	NaN	2009.000000	NaN	NaN	NaN	13086.964800

Exploratory Data Analsis(EDA)

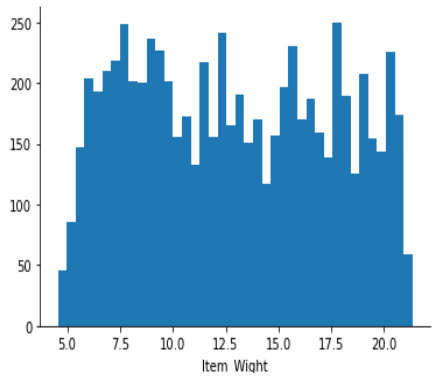
In [11]:
```
#Check for duplicates
idsUnique = len(set(train.Item_Identifier))
idsTotal = train.shape[0]
idsDupli = idsTotal - idsUnique
print("There are " + str(idsDupli) + " duplicate IDs for " + str(idsTotal) + " total entries")
```

```
There are 6964 duplicate IDs for 8523 total entries
```

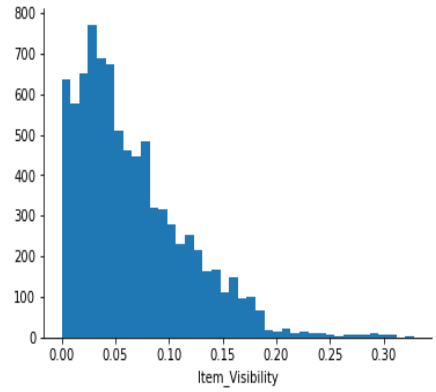
In [12]:
```
plt.hist(train['Item_Outlet_Sales'],bins=50)
sns.despine()
```



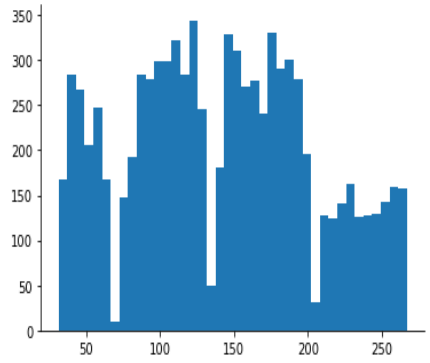
In [13]:
```
plt.hist(train['Item_Weight'],bins=40)
plt.xlabel('Item_Wight')
sns.despine()
```



In [14]:
```
plt.hist(train['Item_Visibility'],bins=40)
plt.xlabel('Item_Visibility')
sns.despine()
```



In [15]:
```
plt.hist(train['Item_MRP'],bins=40)
sns.despine()
```

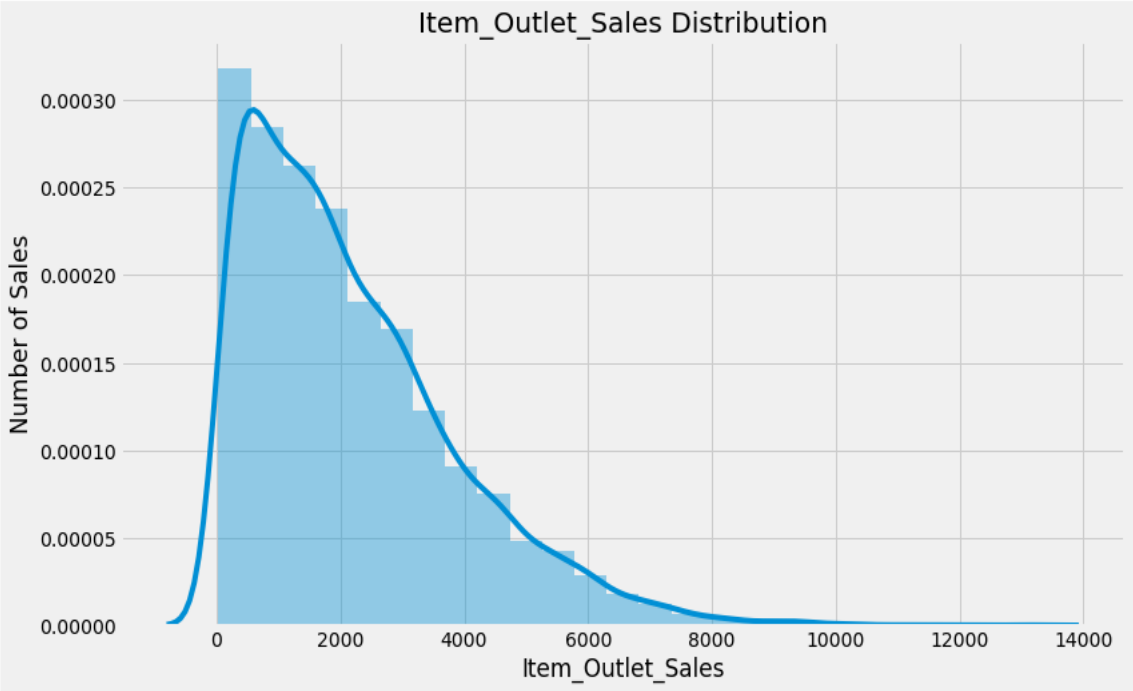


Univariate Analysis

Distribution of the target variable: Item_Outlet_Sales

In [16]:
```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(12,7))
sns.distplot(train.Item_Outlet_Sales, bins = 25)
plt.ticklabel_format(style='plain', axis='x', scilimits=(0,1))
plt.xlabel("Item_Outlet_Sales")
plt.ylabel("Number of Sales")
plt.title("Item_Outlet_Sales Distribution")
```

Out[16]: Text(0.5, 1.0, 'Item_Outlet_Sales Distribution')



In [17]:
```
print ("Skew is:", train.Item_Outlet_Sales.skew())
print("Kurtosis: %f" % train.Item_Outlet_Sales.kurt())
```

Skew is: 1.1775306028542798
Kurtosis: 1.615877

Numeric Features

In [18]:
```
numeric_features = train.select_dtypes(include=[np.number])
numeric_features.dtypes
```

Out[18]:
```
Item_Weight                float64
Item_Visibility            float64
Item_MRP                   float64
Outlet_Establishment_Year    int64
Item_Outlet_Sales          float64
dtype: object
```

Correlation between Numerical Predictors and Target variable

In [19]:
```
corr =numeric_features.corr()
corr
```

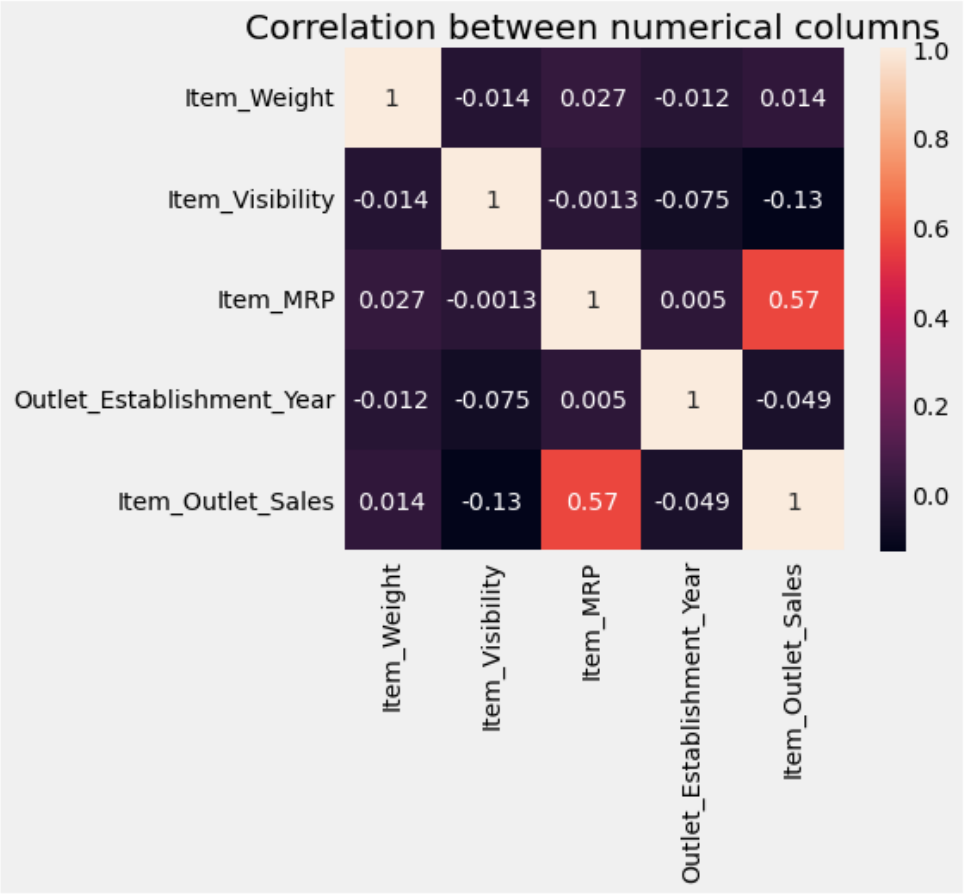
Out[19]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
Item_Weight	1.000000	-0.014048	0.027141	-0.011588	0.014123
Item_Visibility	-0.014048	1.000000	-0.001315	-0.074834	-0.128625
Item_MRP	0.027141	-0.001315	1.000000	0.005020	0.567574
Outlet_Establishment_Year	-0.011588	-0.074834	0.005020	1.000000	-0.049135
Item_Outlet_Sales	0.014123	-0.128625	0.567574	-0.049135	1.000000

In [20]:
```
print(corr['Item_Outlet_Sales'].sort_values(ascending=False))
```

```
Item_Outlet_Sales          1.000000
Item_MRP                   0.567574
Item_Weight                0.014123
Outlet_Establishment_Year -0.049135
Item_Visibility           -0.128625
Name: Item_Outlet_Sales, dtype: float64
```

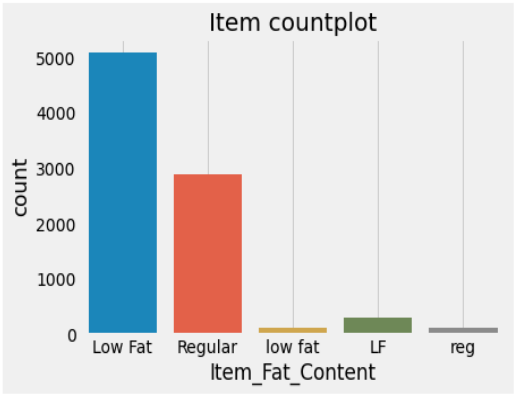
In [21]:
```
#correlation matrix
plt.figure(figsize=(7,5))
sns.heatmap(train.corr(),annot=True,square=True)
sns.despine()
plt.title("Correlation between numerical columns")
plt.show()
```



Categorical Predictors

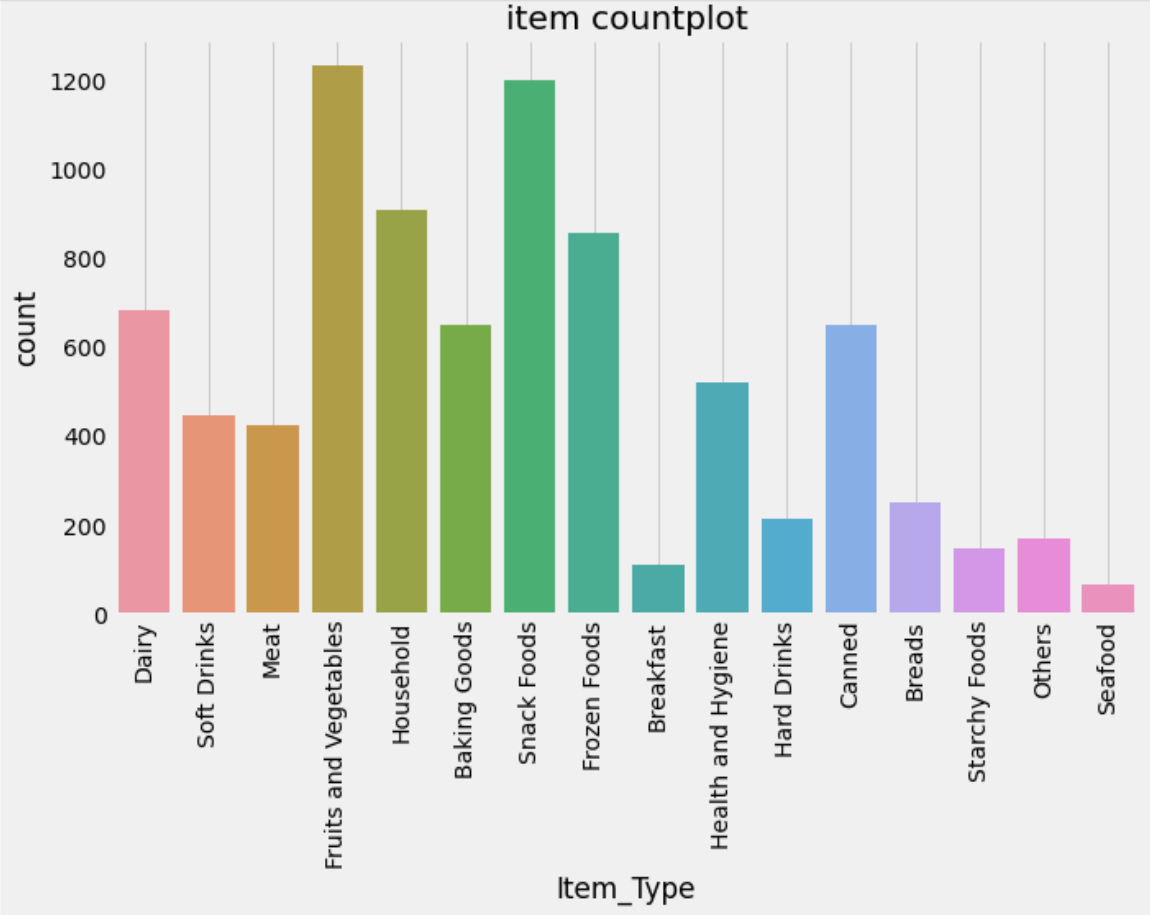
Distribution of the variable Item_Fat_Content

In [22]:
```
sns.countplot(x='Item_Fat_Content',data=train)
plt.title('Item countplot')
plt.grid()
plt.show()
```



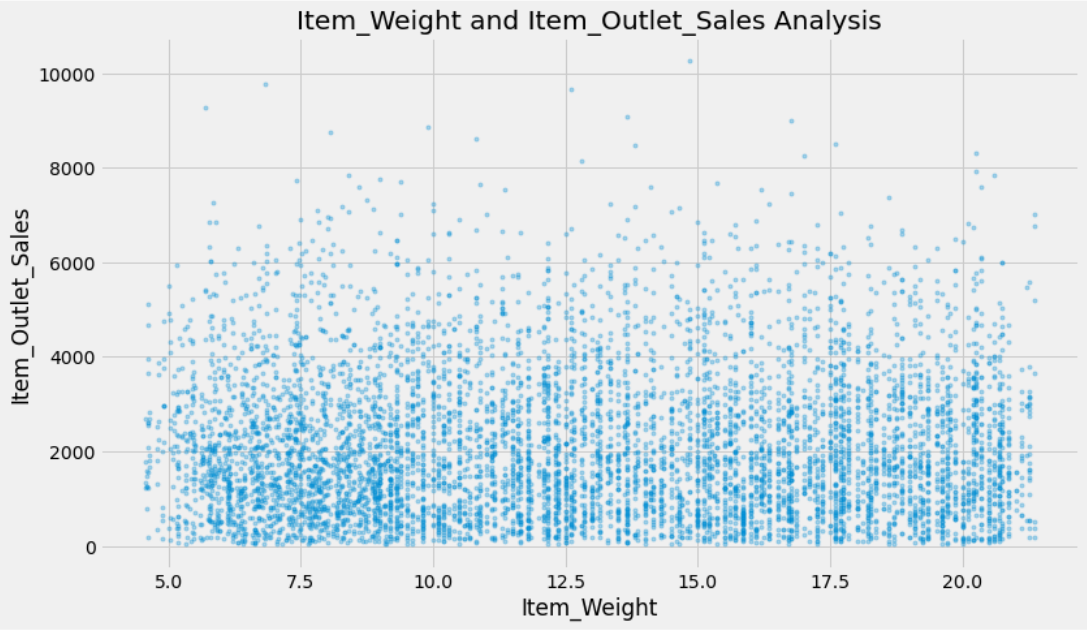
Distribution of the variable Item_Type

In [23]:
```
plt.figure(figsize=(10,6))
sns.countplot(x='Item_Type',data=train)
plt.title('item countplot')
plt.grid()

N=16
ind=np.arange(N)
s=(train['Item_Type'].unique())
plt.xticks(ind,s,rotation='90')
plt.show()
```



In [24]:
```
plt.figure(figsize=(12,7))
plt.xlabel("Item_Weight")
plt.ylabel("Item_Outlet_Sales")
plt.title("Item_Weight and Item_Outlet_Sales Analysis")
plt.plot(train.Item_Weight, train["Item_Outlet_Sales"],'.', alpha = 0.3)
```

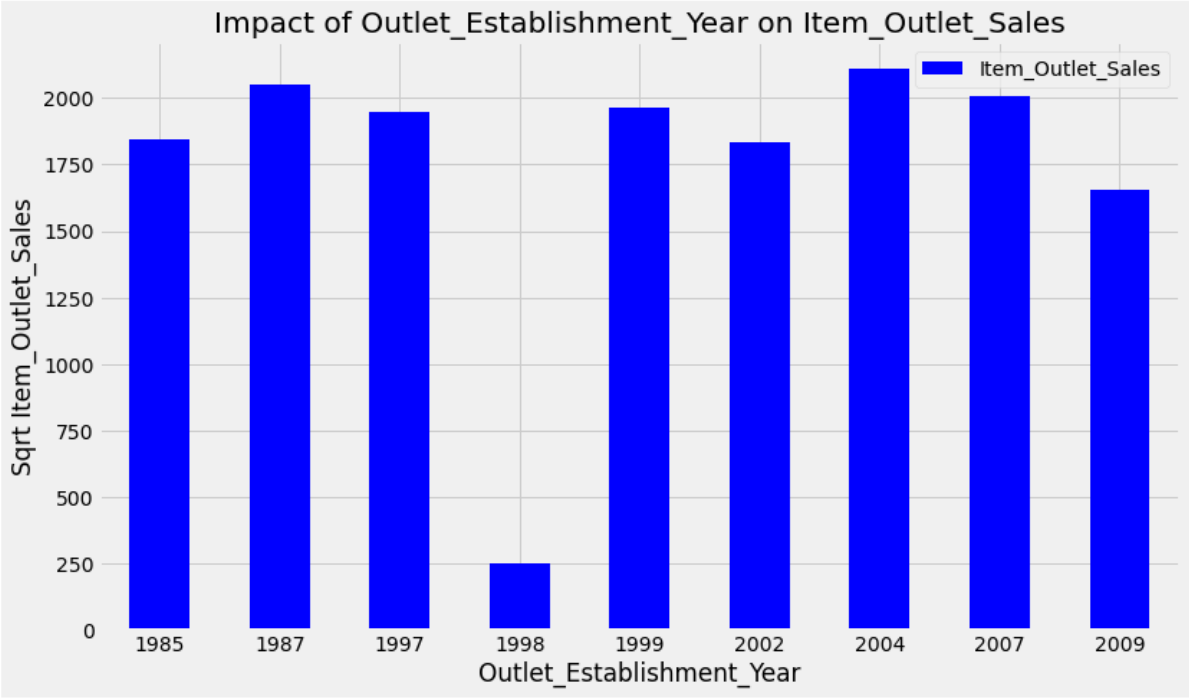
Out[24]: [<matplotlib.lines.Line2D at 0x1bf34f25850>]



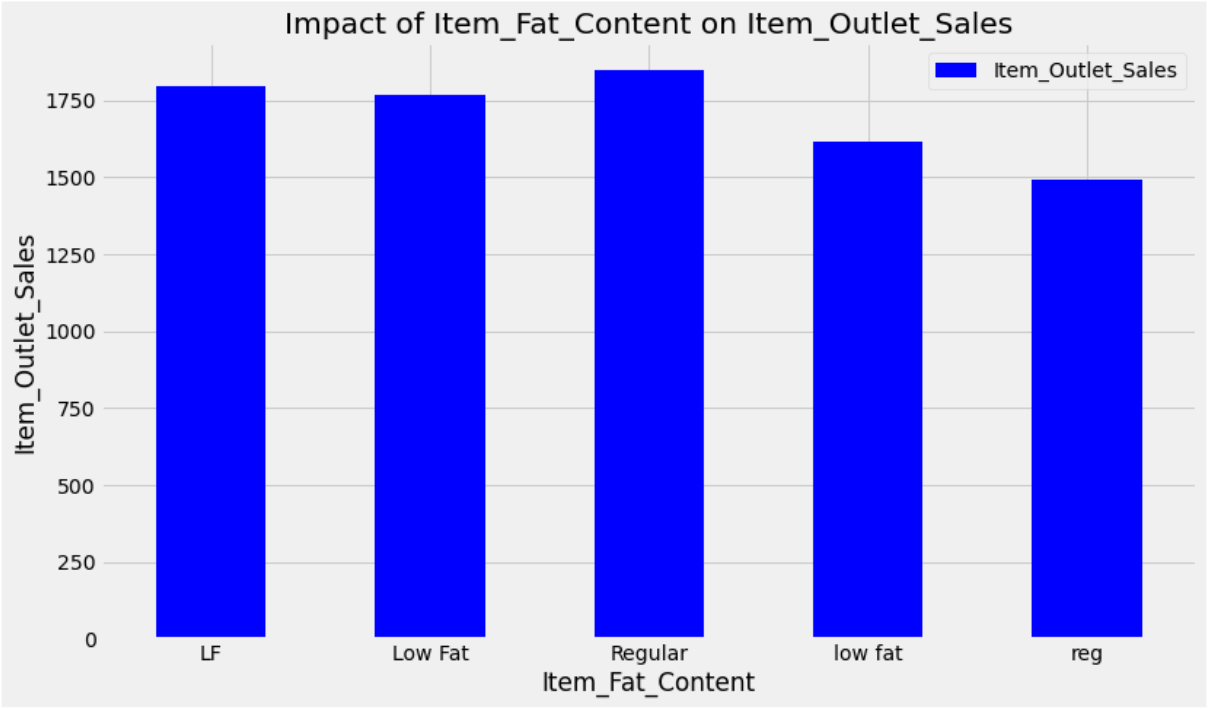
In [25]:
```
plt.figure(figsize=(12,7))
plt.xlabel("Item_Visibility")
plt.ylabel("Item_Outlet_Sales")
plt.title("Item_Visibility and Item_Outlet_Sales Analysis")
plt.plot(train.Item_Visibility, train["Item_Outlet_Sales"],'.', alpha = 0.3)
```

Out[25]: [<matplotlib.lines.Line2D at 0x1bf34f031c0>]

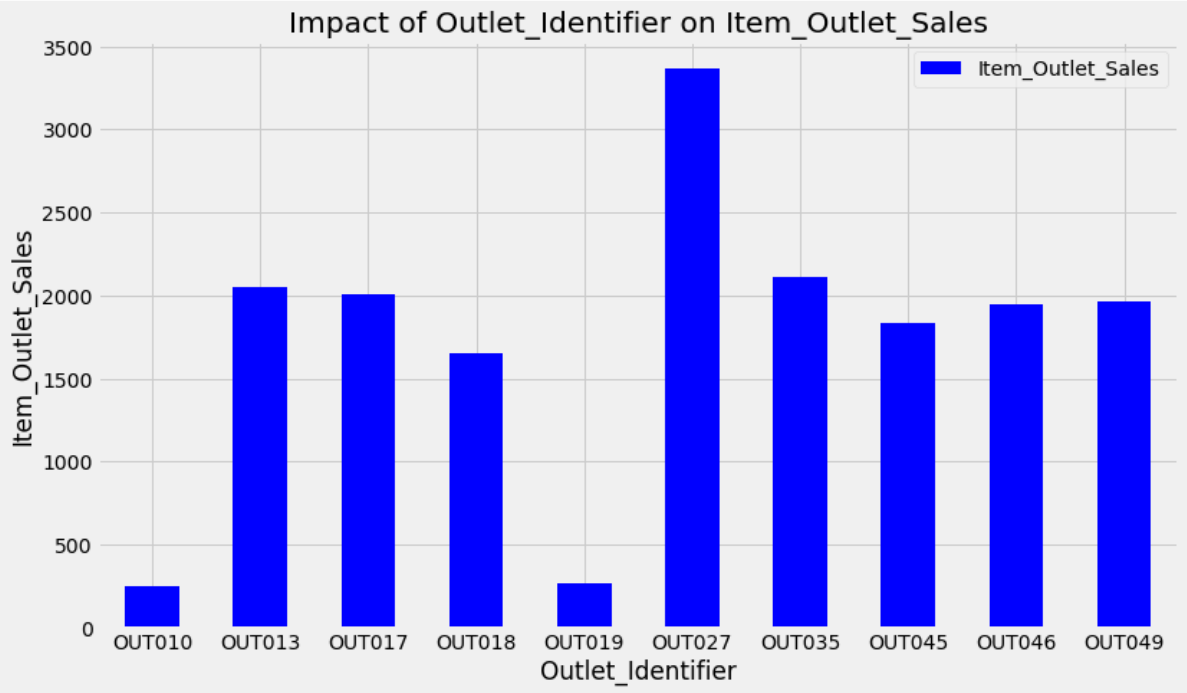
In [26]:
```
Outlet_Establishment_Year_pivot = \
train.pivot_table(index='Outlet_Establishment_Year', values="Item_Outlet_Sales", aggfunc=np.median)
Outlet_Establishment_Year_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Outlet_Establishment_Year")
plt.ylabel("Sqrt Item_Outlet_Sales")
plt.title("Impact of Outlet_Establishment_Year on Item_Outlet_Sales")
plt.xticks(rotation=0)
plt.show()
```



In [27]:
```
Item_Fat_Content_pivot = \
train.pivot_table(index='Item_Fat_Content', values="Item_Outlet_Sales", aggfunc=np.median)
Item_Fat_Content_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Item_Fat_Content")
plt.ylabel("Item_Outlet_Sales")
plt.title("Impact of Item_Fat_Content on Item_Outlet_Sales")
plt.xticks(rotation=0)
plt.show()
```



In [28]:
```
Outlet_Identifier_pivot = \
train.pivot_table(index='Outlet_Identifier', values="Item_Outlet_Sales", aggfunc=np.median)
Outlet_Identifier_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Outlet_Identifier")
plt.ylabel("Item_Outlet_Sales")
plt.title("Impact of Outlet_Identifier on Item_Outlet_Sales")
plt.xticks(rotation=0)
plt.show()
```



In [29]:
```
train.pivot_table(values='Outlet_Type', columns='Outlet_Identifier',aggfunc=lambda x:x.mode())
```

Out[29]:

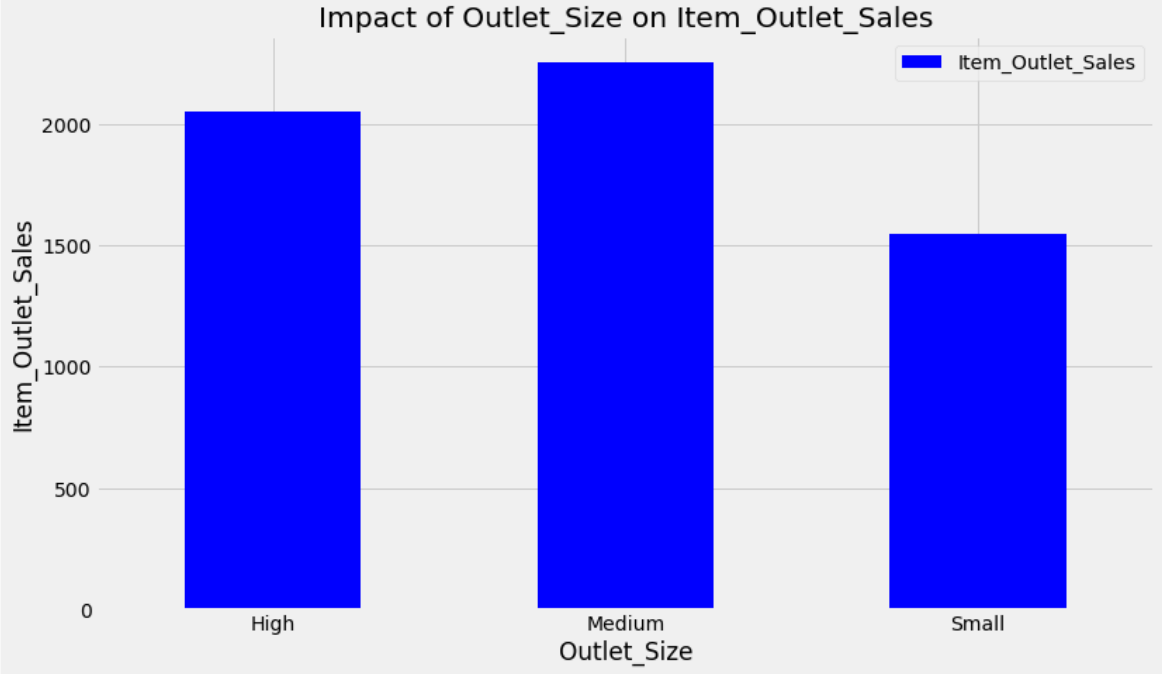
Outlet_Identifier	OUT010	OUT013	OUT017	OUT018	OUT019	OUT027	OUT035	OUT045	OUT046	OUT049
Outlet_Type	Grocery Store	Supermarket Type1	Supermarket Type1	Supermarket Type2	Grocery Store	Supermarket Type3	Supermarket Type1	Supermarket Type1	Supermarket Type1	Supermarket Type1

In [30]:
```
train.pivot_table(values='Outlet_Type', columns='Outlet_Size',aggfunc=lambda x:x.mode())
```

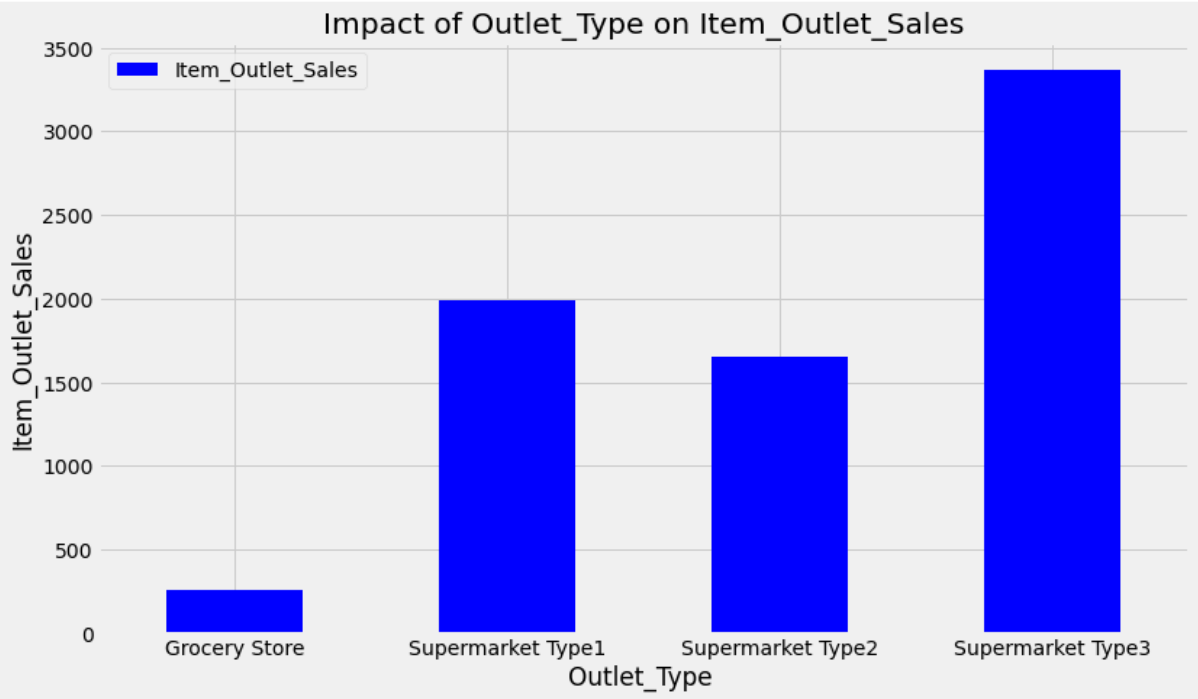
Out[30]:

Outlet_Size	High	Medium	Small
Outlet_Type	Supermarket Type1	Supermarket Type3	Supermarket Type1

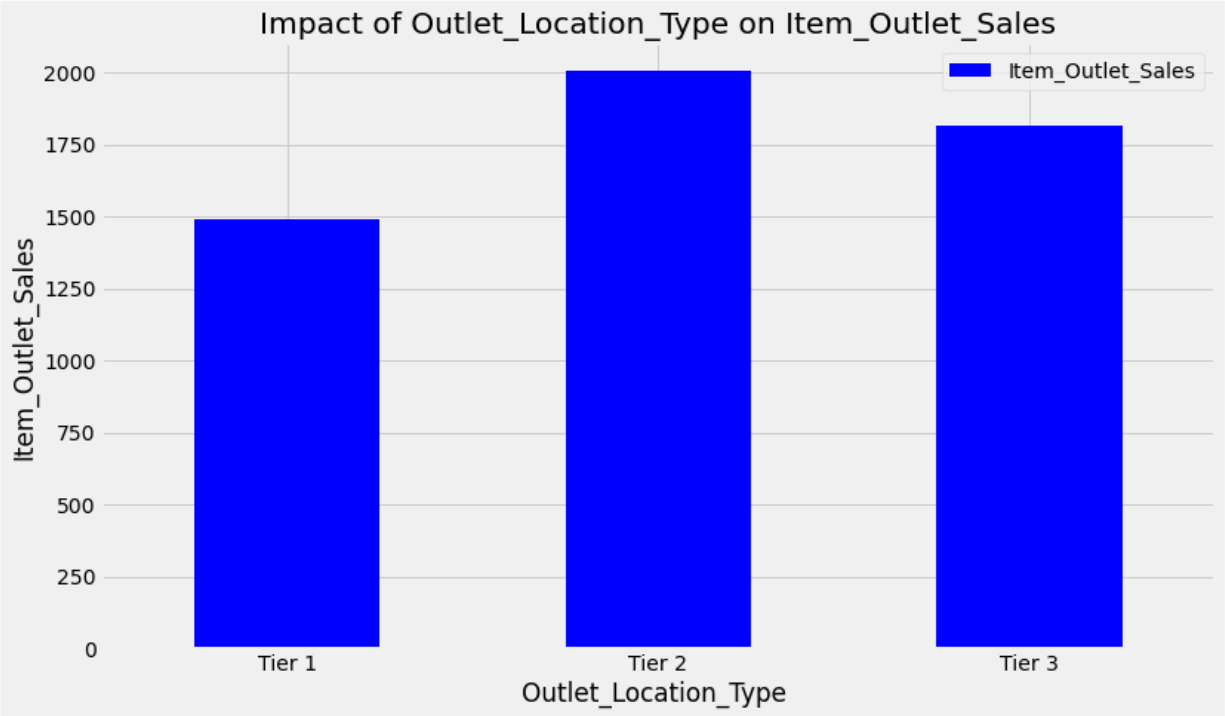
In [31]:
```
Outlet_Size_pivot = \
train.pivot_table(index='Outlet_Size', values="Item_Outlet_Sales", aggfunc=np.median)
Outlet_Size_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Outlet_Size")
plt.ylabel("Item_Outlet_Sales")
plt.title("Impact of Outlet_Size on Item_Outlet_Sales")
plt.xticks(rotation=0)
plt.show()
```



In [32]:
```
Outlet_Type_pivot = \
train.pivot_table(index='Outlet_Type', values="Item_Outlet_Sales", aggfunc=np.median)
Outlet_Type_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Outlet_Type ")
plt.ylabel("Item_Outlet_Sales")
plt.title("Impact of Outlet_Type on Item_Outlet_Sales")
plt.xticks(rotation=0)
plt.show()
```



In [33]:
```
Outlet_Location_Type_pivot = \
train.pivot_table(index='Outlet_Location_Type', values="Item_Outlet_Sales", aggfunc=np.median)
Outlet_Location_Type_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Outlet_Location_Type ")
plt.ylabel("Item_Outlet_Sales")
plt.title("Impact of Outlet_Location_Type on Item_Outlet_Sales")
plt.xticks(rotation=0)
plt.show()
```



In [34]:
```
train.pivot_table(values='Outlet_Location_Type', columns='Outlet_Type',aggfunc=lambda x:x.mode())
```

Out[34]:

Outlet_Type	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3
Outlet_Location_Type	Tier 3	Tier 2	Tier 3	Tier 3

Data Pre-Processing

Looking for missing values

In [35]:
```
# Join Train and Test Dataset
#Create source column to later separate the data easily
train['source']='train'
test['source']='test'

data = pd.concat([train,test], ignore_index = True)
print(train.shape, test.shape, data.shape)
```

```
(8523, 13) (5681, 12) (14204, 13)
```

Imputing Missing Values

In [36]:
```
#Check the percentage of null values per variable
data.isnull().sum()/data.shape[0]*100 #show values in percentage
```

Out[36]:
```
Item_Identifier             0.000000
Item_Weight                17.171219
Item_Fat_Content            0.000000
Item_Visibility             0.000000
Item_Type                   0.000000
Item_MRP                    0.000000
Outlet_Identifier           0.000000
Outlet_Establishment_Year   0.000000
Outlet_Size                28.273726
Outlet_Location_Type        0.000000
Outlet_Type                 0.000000
Item_Outlet_Sales          39.995776
source                      0.000000
dtype: float64
```

Imputing the mean for Item_Weight missing values

In [37]:
```
#aggfunc is mean by default! Ignores NaN by default

item_avg_weight = data.pivot_table(values='Item_Weight', index='Item_Identifier')
print(item_avg_weight)
```

```
                Item_Weight
Item_Identifier
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 x 1 columns]
```

In [38]:
```
def impute_weight(cols):
    Weight = cols[0]
    Identifier = cols[1]

    if pd.isnull(Weight):
        return item_avg_weight['Item_Weight'][item_avg_weight.index == Identifier]
    else:
        return Weight
print ('Orignal #missing: %d'%sum(data['Item_Weight'].isnull()))
data['Item_Weight'] = data[['Item_Weight','Item_Identifier']].apply(impute_weight,axis=1).astype(float)
print ('Final #missing: %d'%sum(data['Item_Weight'].isnull()))
```

```
Orignal #missing: 2439
Final #missing: 0
```

Imputing Outlet_Size missing values with the mode

In [39]:
```
#Import mode function:
from scipy.stats import mode

#Determing the mode for each
outlet_size_mode = data.pivot_table(values='Outlet_Size', columns='Outlet_Type',aggfunc=lambda x:x.mode())

outlet_size_mode
```

Out[39]:

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

In [40]:
```
def impute_size_mode(cols):
    Size = cols[0]
    Type = cols[1]
    if pd.isnull(Size):
        return outlet_size_mode.loc['Outlet_Size'][outlet_size_mode.columns == Type][0]
    else:
        return Size
print ('Orignal #missing: %d'%sum(data['Outlet_Size'].isnull()))
data['Outlet_Size'] = data[['Outlet_Size','Outlet_Type']].apply(impute_size_mode,axis=1)
print ('Final #missing: %d'%sum(data['Outlet_Size'].isnull()))
```

```
Orignal #missing: 4016
Final #missing: 0
```

Feature Engineering

Should we combine Outlet_Type?

In [41]:
```
#Creates pivot table with Outlet_Type and the mean of #Item_Outlet_Sales. Agg function is by default mean()
data.pivot_table(values='Item_Outlet_Sales', columns='Outlet_Type')
```

Out[41]:

	Outlet_Type	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3
Item_Outlet_Sales		339.8285	2316.181148	1995.498739	3694.038558

In [42]:
```
#aggfunc is mean by default! Ignores NaN by default

visibility_item_avg = data.pivot_table(values='Item_Visibility', index='Item_Identifier')
print(visibility_item_avg)
```

```
                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]
```

Item_Visibility minimum value is 0

In [43]:
```
def impute_visibility_mean(cols):
    visibility = cols[0]
    item = cols[1]
    if visibility == 0:
        return visibility_item_avg['Item_Visibility'][visibility_item_avg.index == item]
    else:
        return visibility
print ('Original #zeros: %d'%sum(data['Item_Visibility']==0))
data[['Item_Visibility','Item_Identifier']].apply(impute_visibility_mean,axis=1).astype(float)
print ('Final #zeros: %d'%sum(data['Item_Visibility'].isnull()))
```

```
Original #zeros: 879
Final #zeros: 0
```

Determine the years of operation of a store

In [44]:
```
#Remember the data is from 2013
data['Outlet_Years'] = 2013 - data['Outlet_Establishment_Year']
data['Outlet_Years'].describe()
```

Out[44]:
```
count    14204.000000
mean        15.169319
std          8.371664
min          4.000000
25%          9.000000
50%         14.000000
75%         26.000000
max         28.000000
Name: Outlet_Years, dtype: float64
```

Create a broad category of Item_Type

In [45]:
```
#Get the first two characters of ID:
data['Item_Type_Combined'] = data['Item_Identifier'].apply(lambda x: x[0:2])
#Rename them to more intuitive categories:
data['Item_Type_Combined'] = data['Item_Type_Combined'].map({'FD':'Food',
'NC':'Non-Consumable',                                          'DR':'Drinks'})
data['Item_Type_Combined'].value_counts()
```

Out[45]:
```
Food              10201
Non-Consumable     2686
Drinks             1317
Name: Item_Type_Combined, dtype: int64
```

Modify categories of Item_Fat_Content

In [46]:
```
#Change categories of low fat:
print('Original Categories:')
print(data['Item_Fat_Content'].value_counts())
print('\nModified Categories:')
data['Item_Fat_Content'] = data['Item_Fat_Content'].replace({'LF':'Low Fat',
'reg':'Regular',
'low fat':'Low Fat'})
print(data['Item_Fat_Content'].value_counts())
```

```
Original Categories:
Low Fat    8485
Regular    4824
LF          522
reg         195
low fat     178
Name: Item_Fat_Content, dtype: int64

Modified Categories:
Low Fat    9185
Regular    5019
Name: Item_Fat_Content, dtype: int64
```

In [47]:
```
#Mark non-consumables as separate category in low_fat:
data.loc[data['Item_Type_Combined']=="Non-Consumable",'Item_Fat_Content'] = "Non-Edible"
data['Item_Fat_Content'].value_counts()
```

Out[47]:
```
Low Fat       6499
Regular       5019
Non-Edible    2686
Name: Item_Fat_Content, dtype: int64
```

Feature Transformations

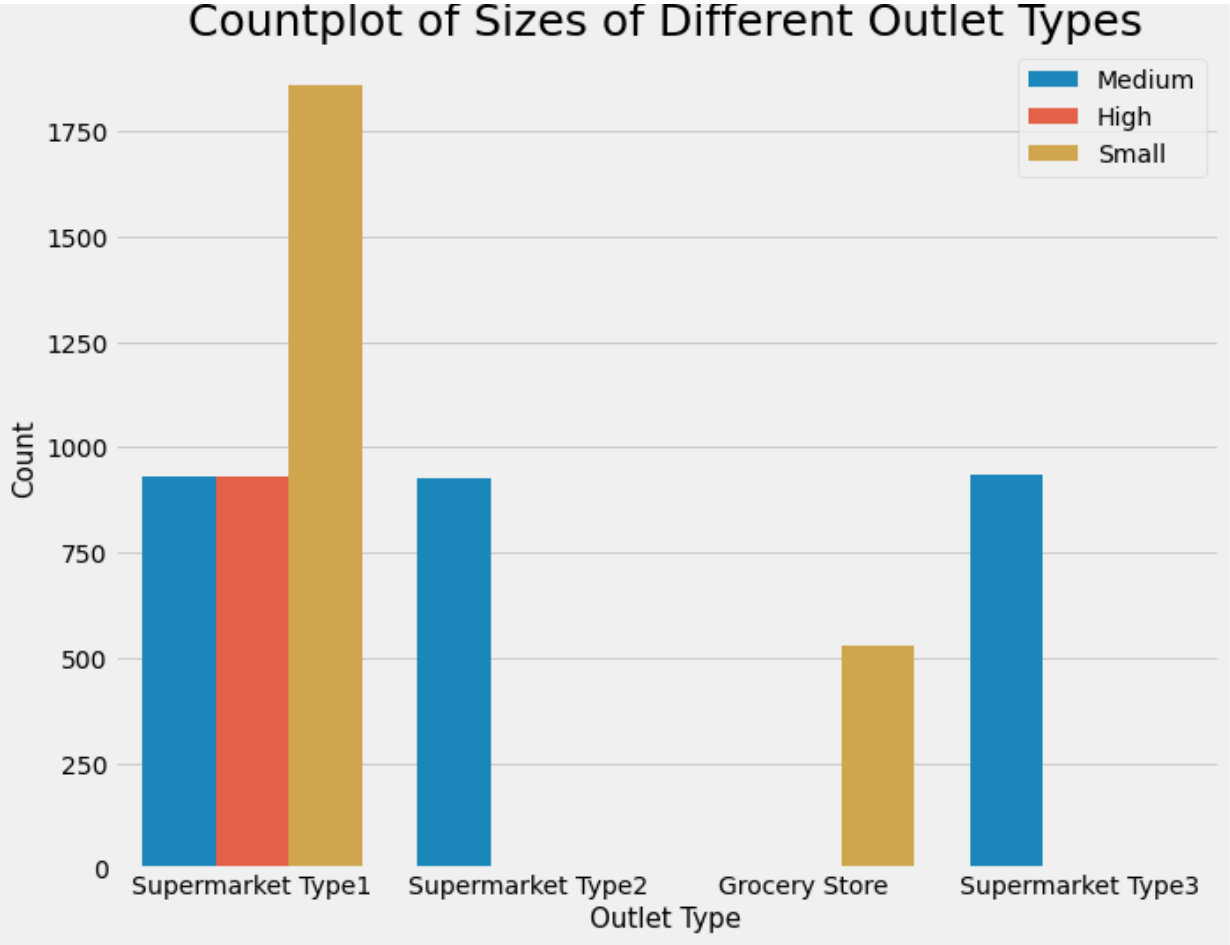
Creating variable Item_Visibility_Mean_Ratio

In [48]:
```
func = lambda x: x['Item_Visibility']/visibility_item_avg['Item_Visibility'][visibility_item_avg.index == x['Item_Identifier']]
[0]
data['Item_Visibility_MeanRatio'] = data.apply(func,axis=1).astype(float)
data['Item_Visibility_MeanRatio'].describe()
```

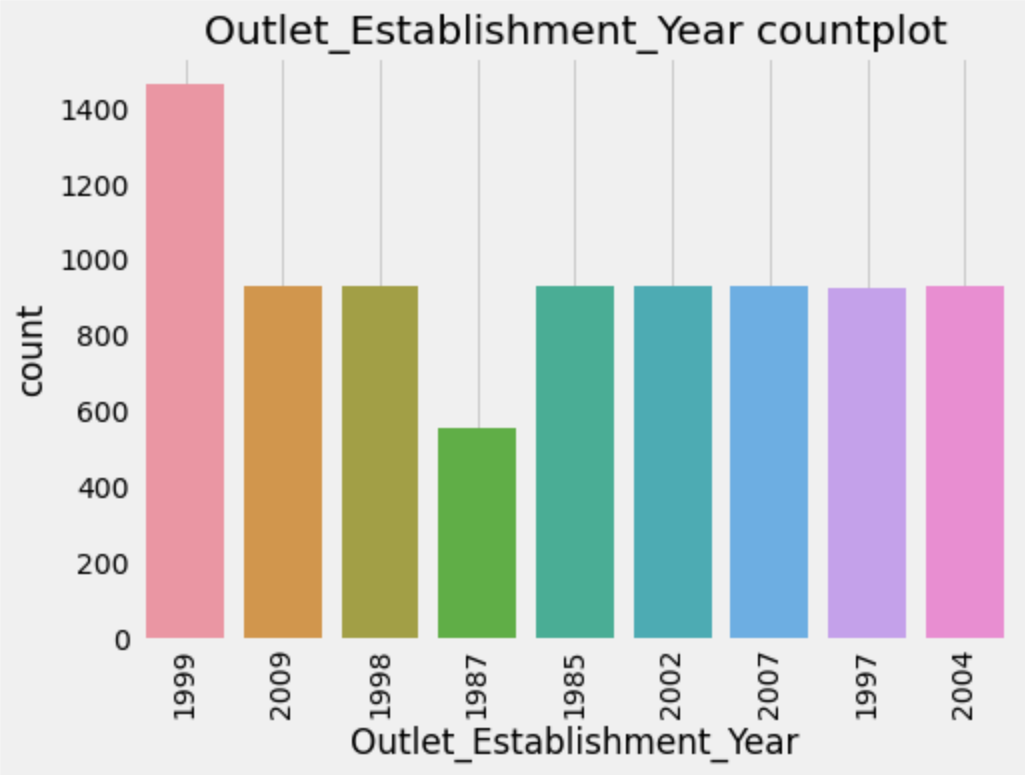
Out[48]:
```
count    14204.000000
mean         1.000000
std          0.348382
min          0.000000
25%          0.921522
50%          0.962037
75%          1.042007
max          3.010094
Name: Item_Visibility_MeanRatio, dtype: float64
```

Distribution of the variable Outlet_Type

In [49]:
```
# Plot bar chart of different outlet sizes
plt.figure(figsize = (10, 8))
sns.countplot(x = 'Outlet_Type', hue = 'Outlet_Size', data = train)
plt.legend(loc='upper right')
plt.title('Countplot of Sizes of Different Outlet Types', fontsize = 25)
plt.xlabel('Outlet Type', fontsize = 15)
plt.ylabel('Count', fontsize = 15)

plt.show()
```

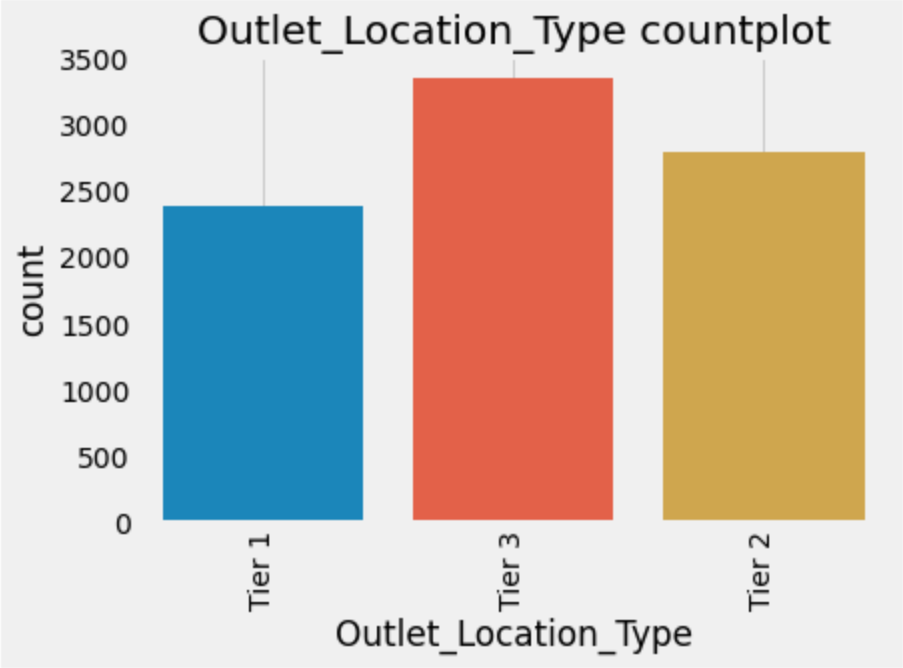


In [50]:
```
plt.figure(figsize=(7,5))
sns.countplot(x='Outlet_Establishment_Year',data=train)
plt.title('Outlet_Establishment_Year countplot')
plt.grid()

N=9
ind=np.arange(N)
s=(train['Outlet_Establishment_Year'].unique())
plt.xticks(ind,s,rotation='vertical')
plt.show()
```

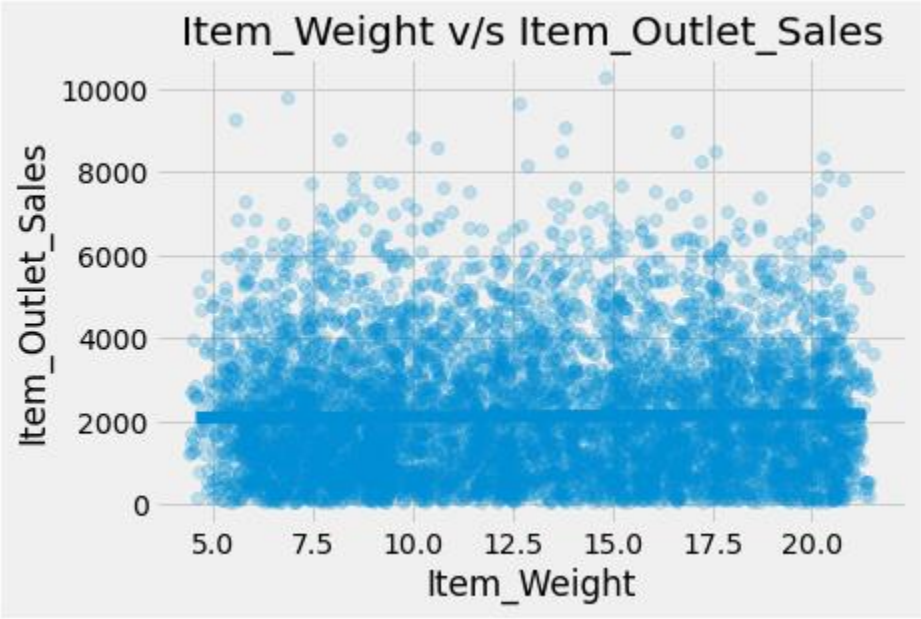


Distribution of the variable Outlet_Location_Type

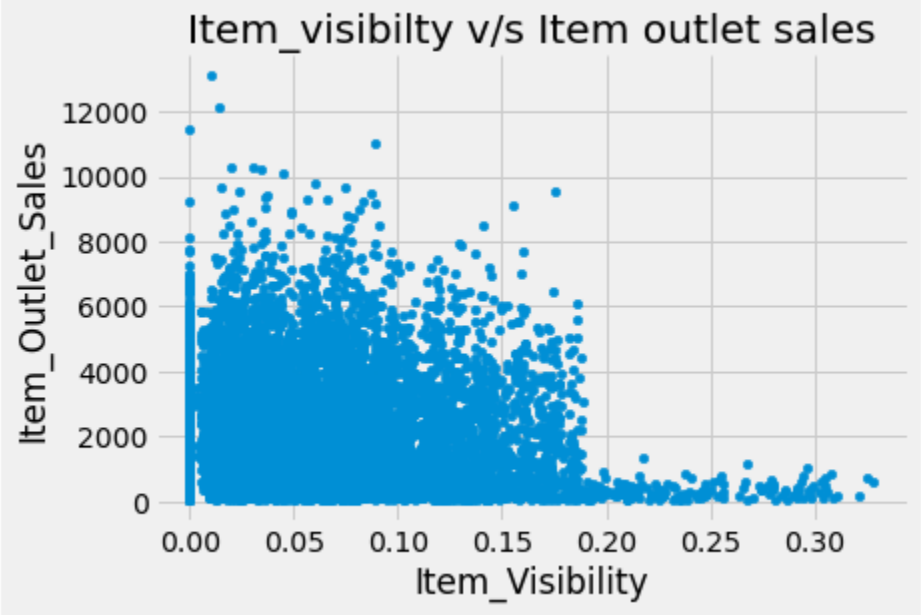
In [51]:
```
sns.countplot(x='Outlet_Location_Type',data=train)
plt.title('Outlet_Location_Type countplot')
plt.grid()

N=3
ind=np.arange(N)
s=(train['Outlet_Location_Type'].unique())
plt.xticks(ind,s,rotation='vertical')
plt.show()
```



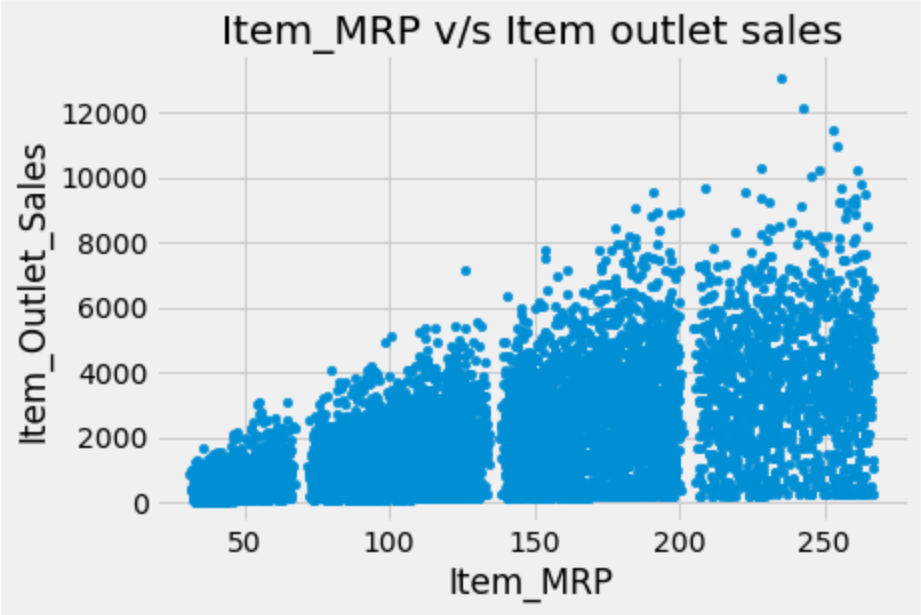
In [52]:
```
sns.regplot(x='Item_Weight',y='Item_Outlet_Sales',data=train,x_jitter=0.2,scatter_kws={'alpha':0.2})
plt.title("Item_Weight v/s Item_Outlet_Sales")
plt.show()
```



In [53]:
```
train.plot.scatter(x='Item_Visibility' ,y='Item_Outlet_Sales').set(title='Item_visibilty v/s Item outlet sales',
      xlabel='Item_Visibility',
      ylabel='Item_Outlet_Sales')
sns.despine()
```

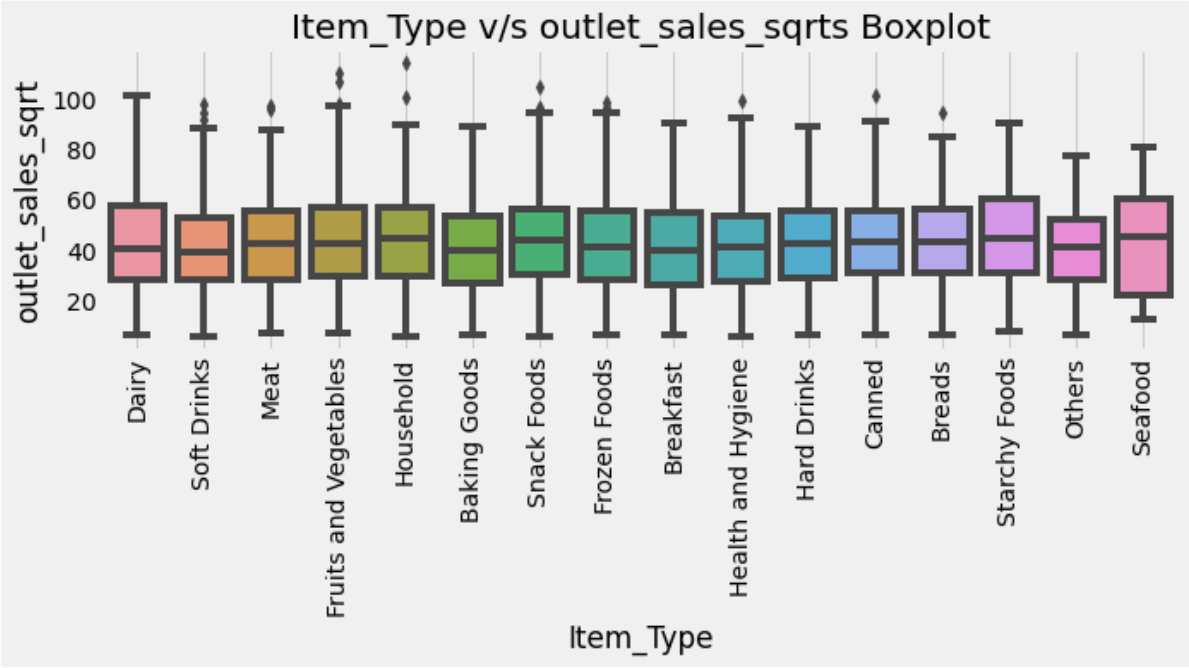


In [54]:
```
train.plot.scatter(x='Item_MRP' ,y='Item_Outlet_Sales').set(title='Item_MRP v/s Item outlet sales',
      xlabel='Item_MRP',
      ylabel='Item_Outlet_Sales')
sns.despine()
```

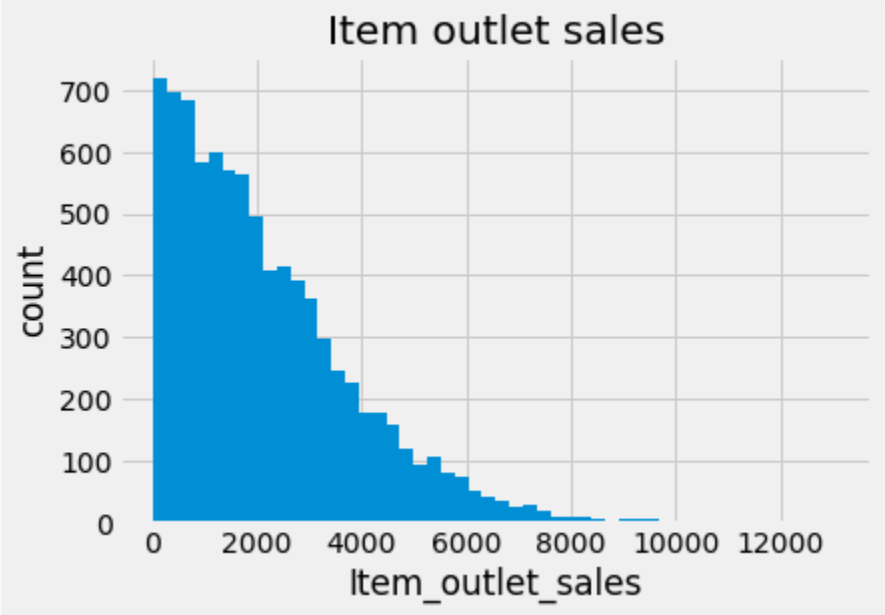


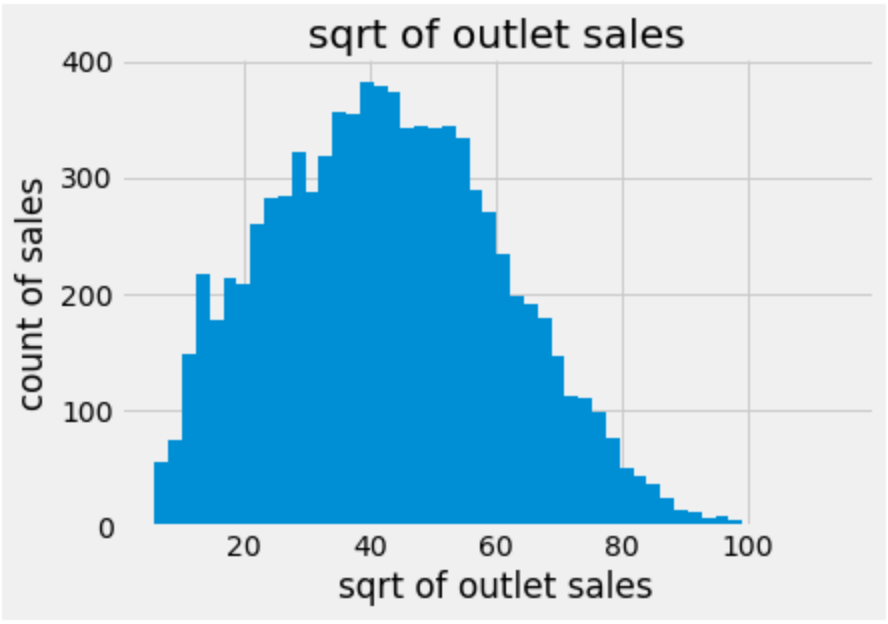
In [55]:
```
train['outlet_sales_sqrt']=train['Item_Outlet_Sales'].apply(np.sqrt)
```

In [56]:
```
plt.figure(figsize=(10,3))
sns.boxplot(x='Item_Type',y='outlet_sales_sqrt',data=train)
plt.title("Item_Type v/s outlet_sales_sqrts Boxplot")
N=16
ind=np.arange(N)
s=(train['Item_Type'].unique())
plt.xticks(ind,s,rotation='vertical')
plt.grid()
plt.show()
```

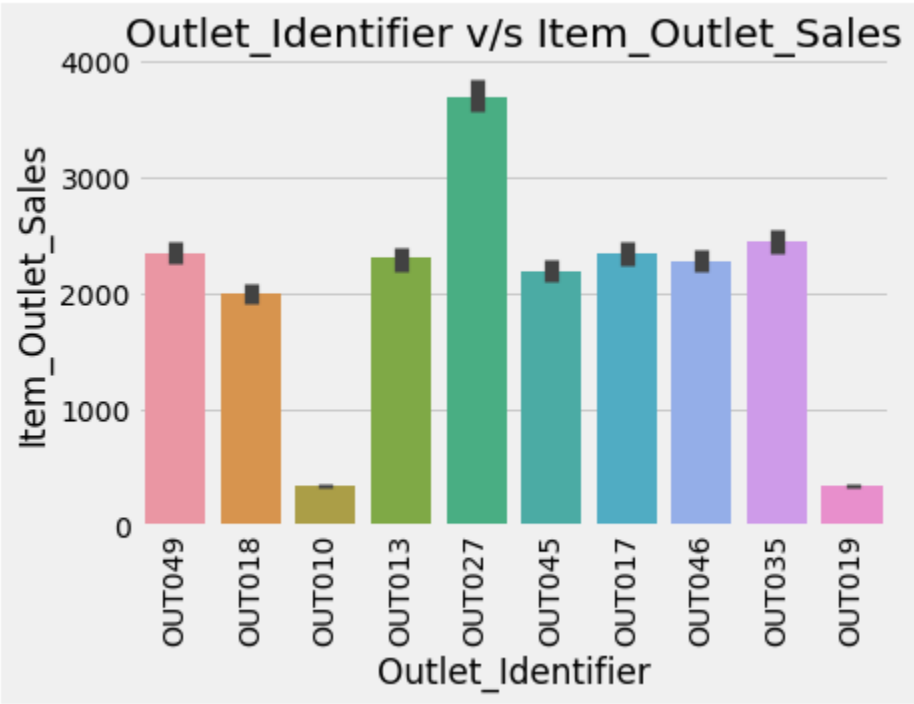


In [57]:
```
plt.hist(train['Item_Outlet_Sales'],bins=50)
plt.title('Item outlet sales')
plt.xlabel('Item_outlet_sales')
plt.ylabel('count')
plt.show()
plt.hist(train['outlet_sales_sqrt'],bins=50)
plt.title('sqrt of outlet sales')
plt.xlabel('sqrt of outlet sales')
plt.ylabel('count of sales')
sns.despine()
```





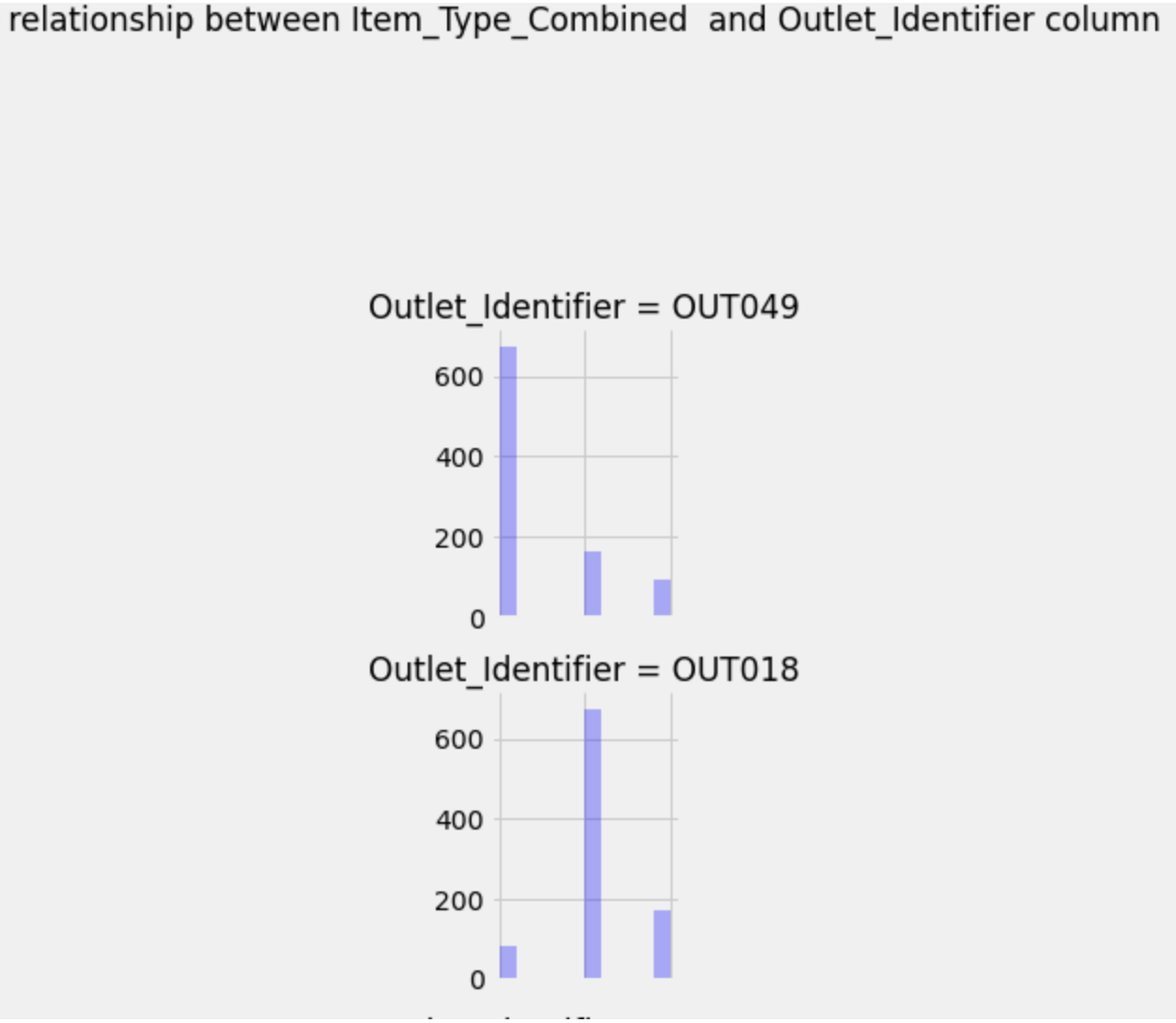
In [58]:
```
sns.barplot(x='Outlet_Identifier',y='Item_Outlet_Sales',data=train)
N=10
ind=np.arange(N)
plt.title("Outlet_Identifier v/s Item_Outlet_Sales")
s=(train['Outlet_Identifier'].unique())
plt.xticks(ind,s,rotation='vertical')
plt.show()
```

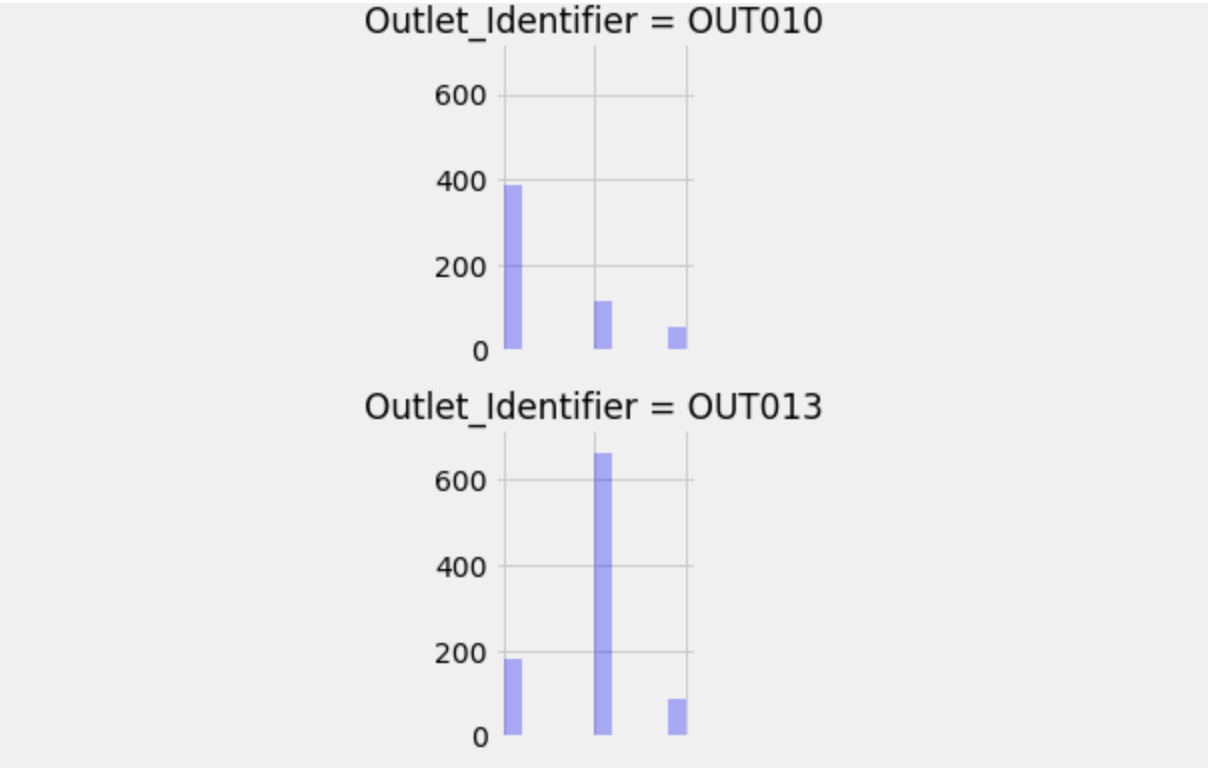


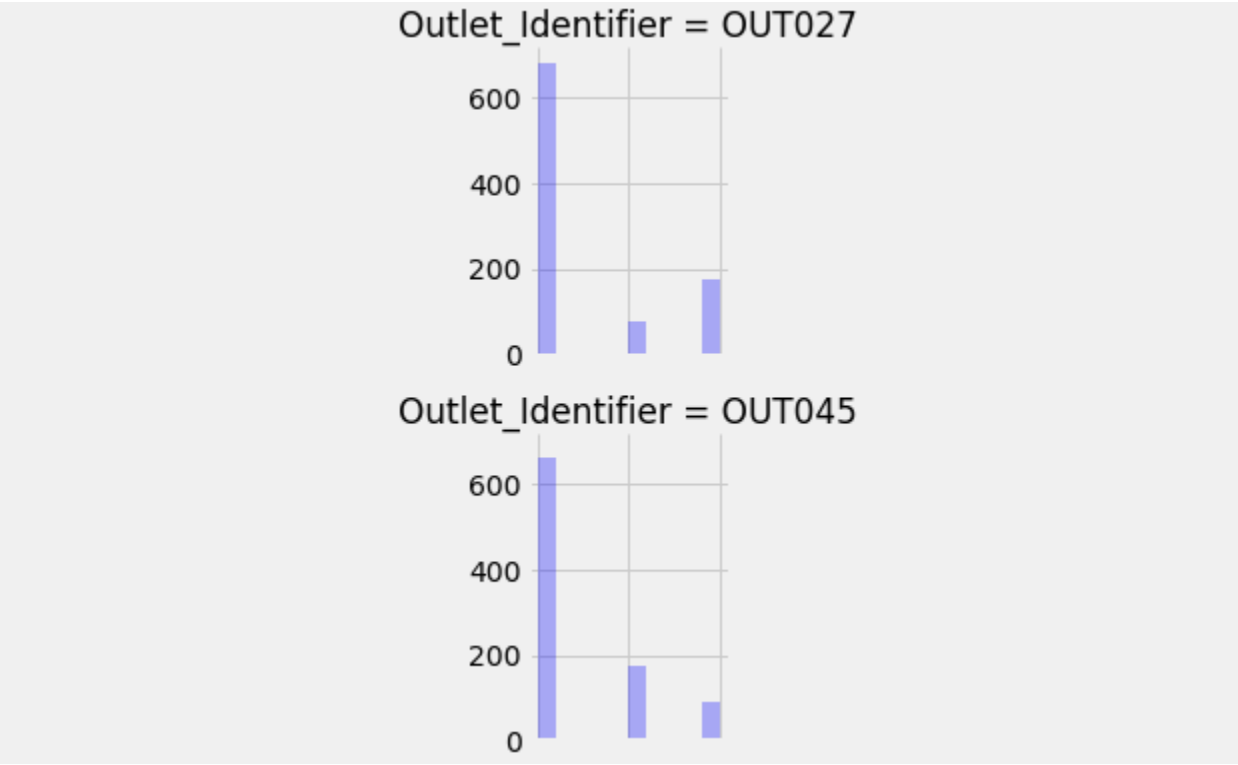
In [59]:
```
#Get the first two characters of ID:
train['Item_Type_Combined'] = train['Item_Identifier'].apply(lambda x: x[0:2])
#Rename them to more intuitive categories:
train['Item_Type_Combined'] = train['Item_Type_Combined'].map({'FD':'Food',
                                                               'NC':'Non-Consumable',
                                                               'DR':'Drinks'})
train['Item_Type_Combined'].value_counts()
```

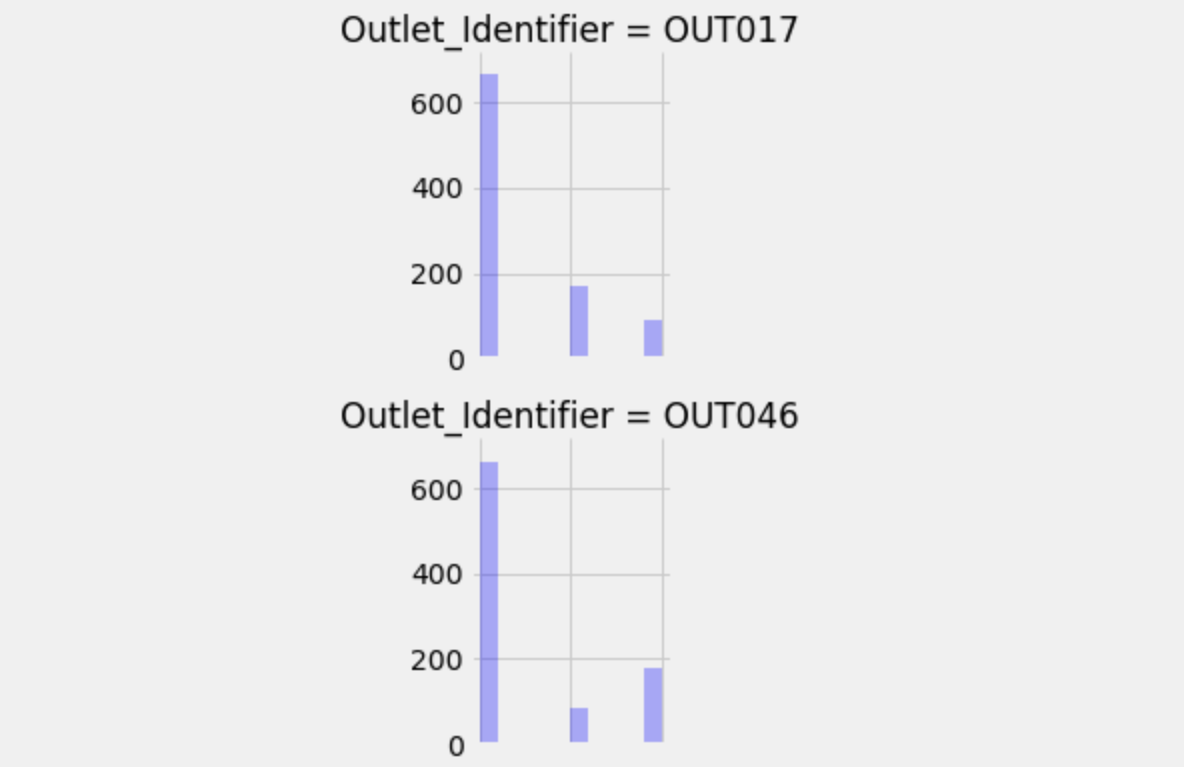
Out[59]:
```
Food             6125
Non-Consumable    1599
Drinks             799
Name: Item_Type_Combined, dtype: int64
```

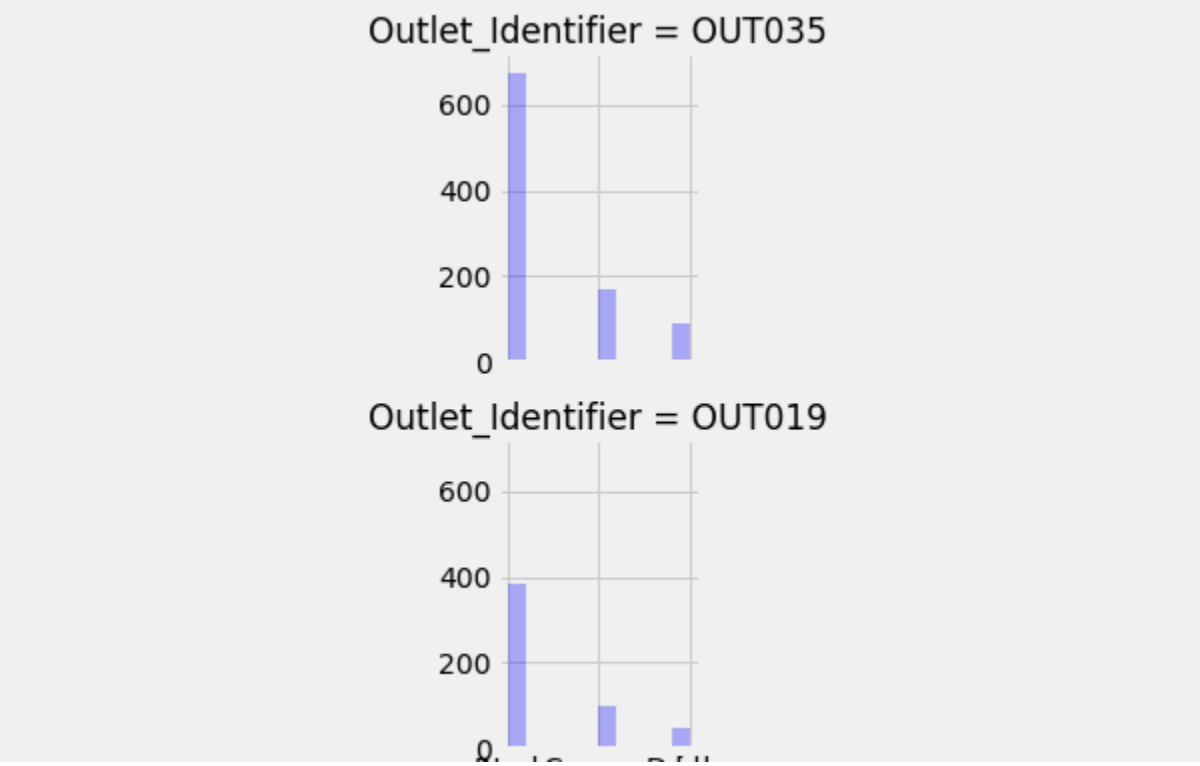
In [60]:
```
grid=sns.FacetGrid(train,col='Outlet_Identifier',col_wrap=1)
grid.map(plt.hist,'Item_Type_Combined',alpha=0.3,color='b')
plt.subplots_adjust(top=0.9)
grid.fig.suptitle(' relationship between Item_Type_Combined  and Outlet_Identifier column ')
plt.show()
```


relationship between Item_Type_Combined and Outlet_Identifier column
Outlet_Identifier = OUT049
600
400
200
0
Outlet_Identifier = OUT018
600
400
200
0

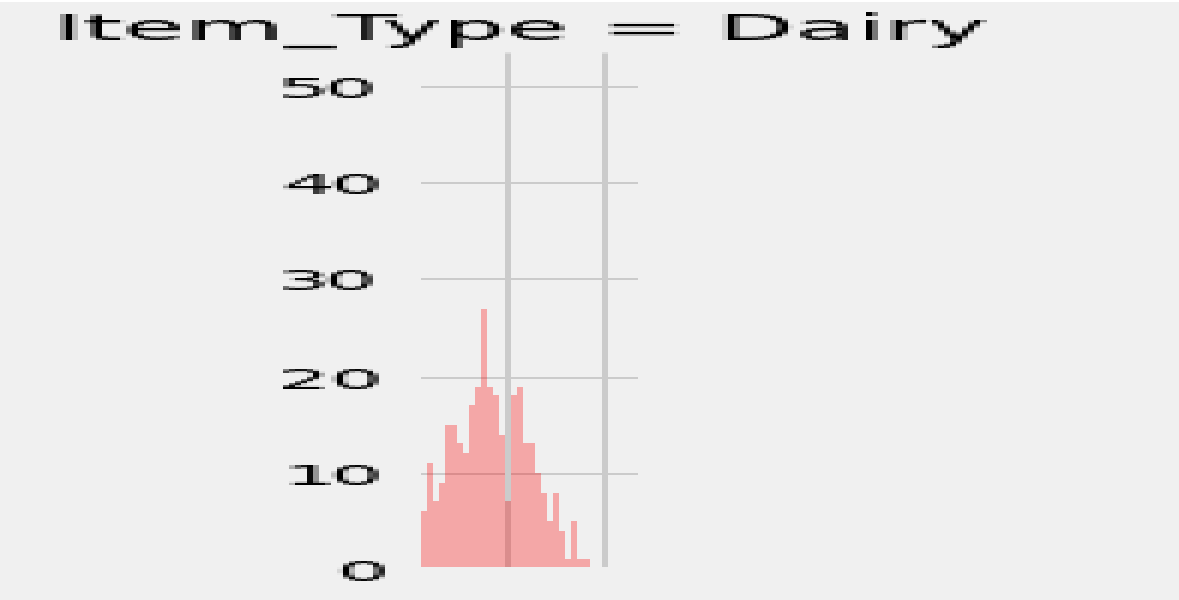

Outlet_Identifier = OUT010
600
400
200
0
Outlet_Identifier = OUT013
600
400
200
0

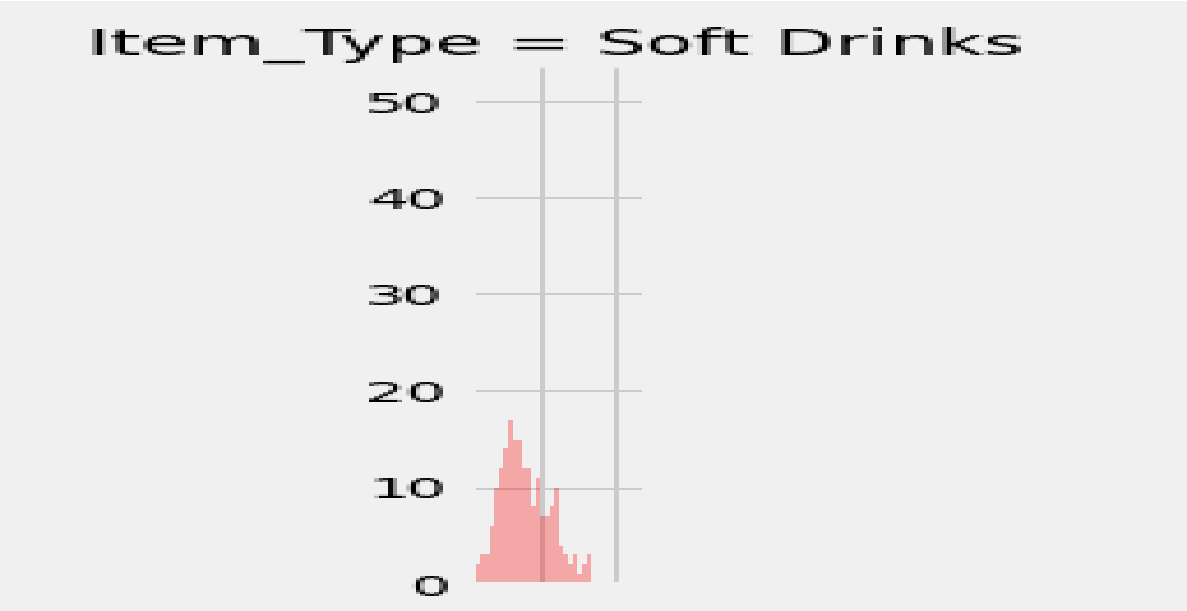

Outlet_Identifier = OUT027
600
400
200
0
Outlet_Identifier = OUT045
600
400
200
0

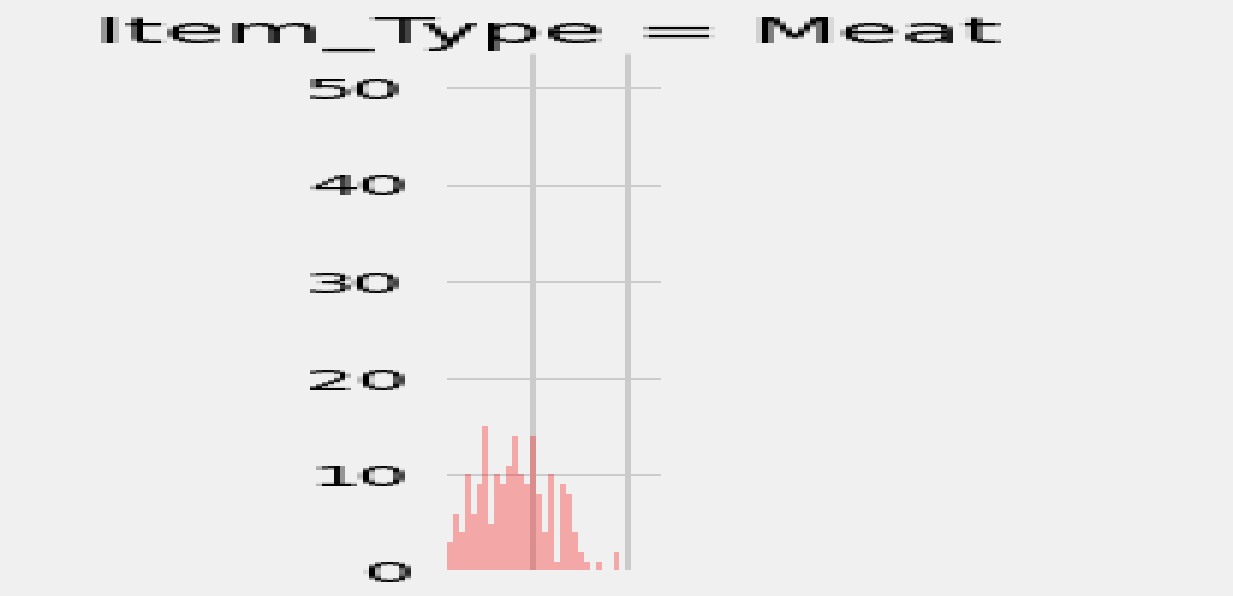


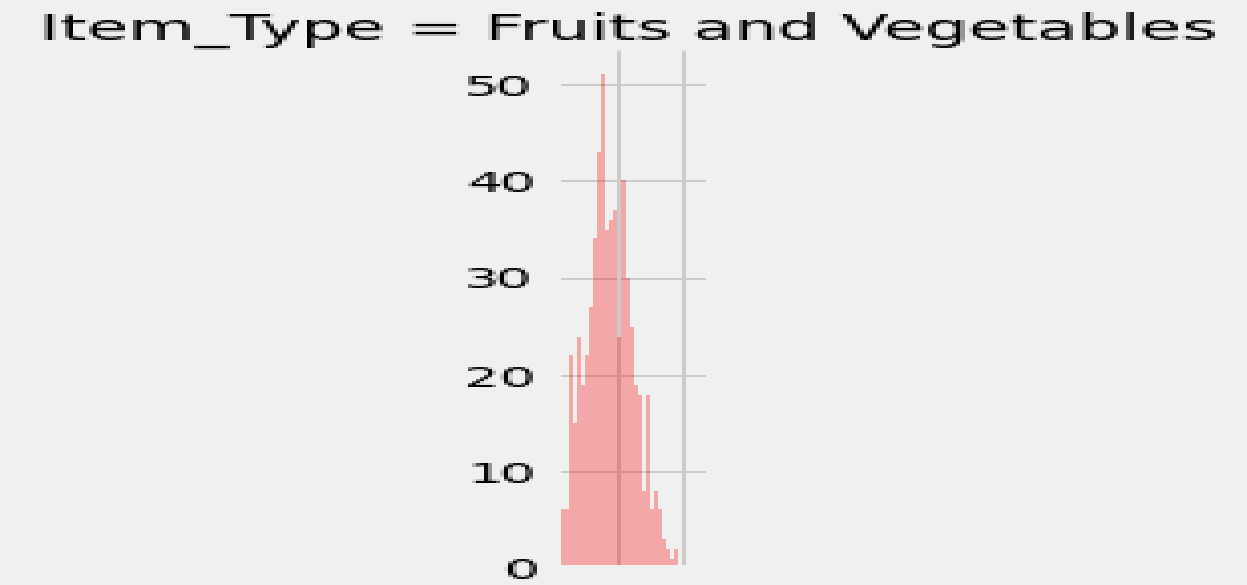


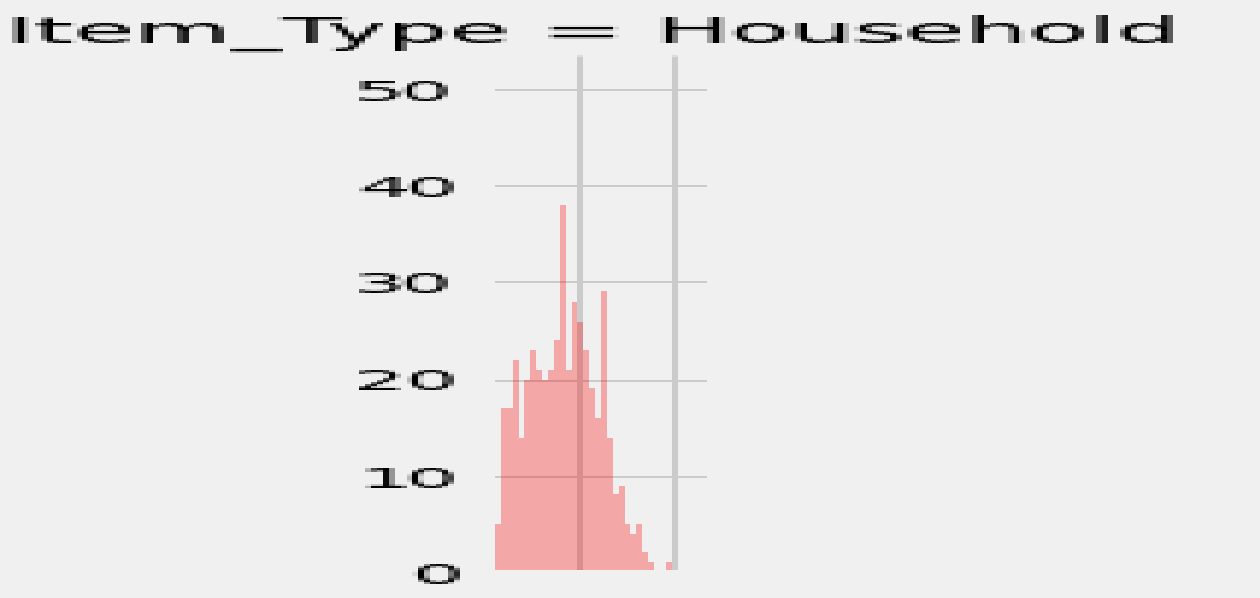
In [61]:
```
grid=sns.FacetGrid(train,col='Item_Type',col_wrap=4)
grid.map(plt.hist,'outlet_sales_sqrt',alpha=0.3,color='r',bins=70)
plt.subplots_adjust(top=1.5)
grid.fig.suptitle(' relationship between Item Type/Item Identifier and Outlet_sales_sqrt column ')
plt.show()
```

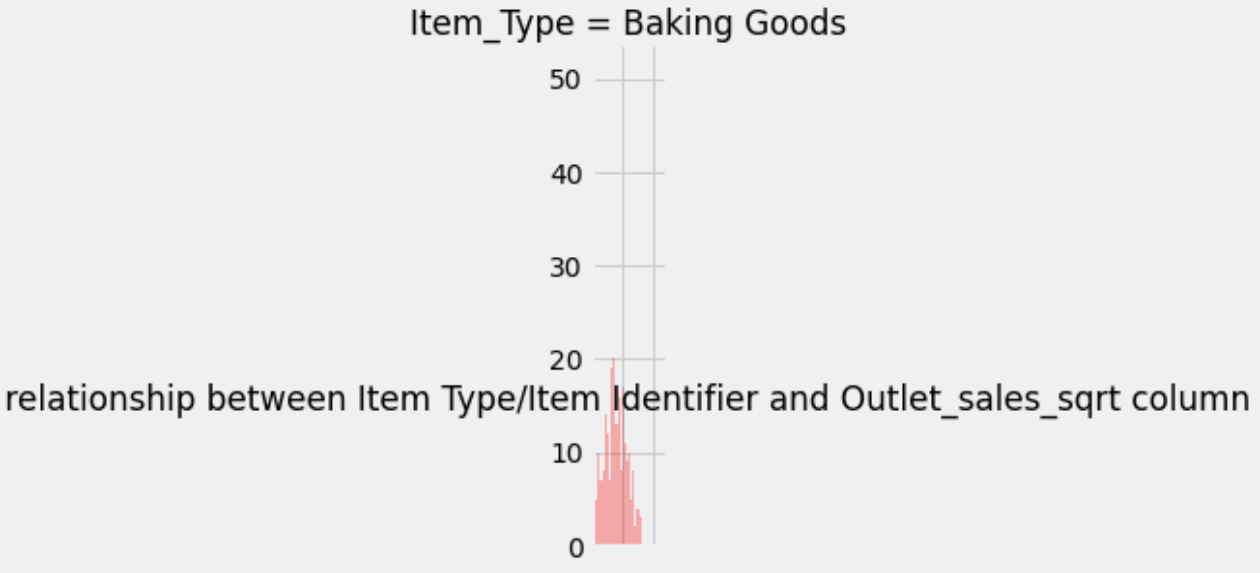

Item_Type = Dairy
50
40
30
20
10
0


Item_Type = Soft Drinks
50
40
30
20
10
0

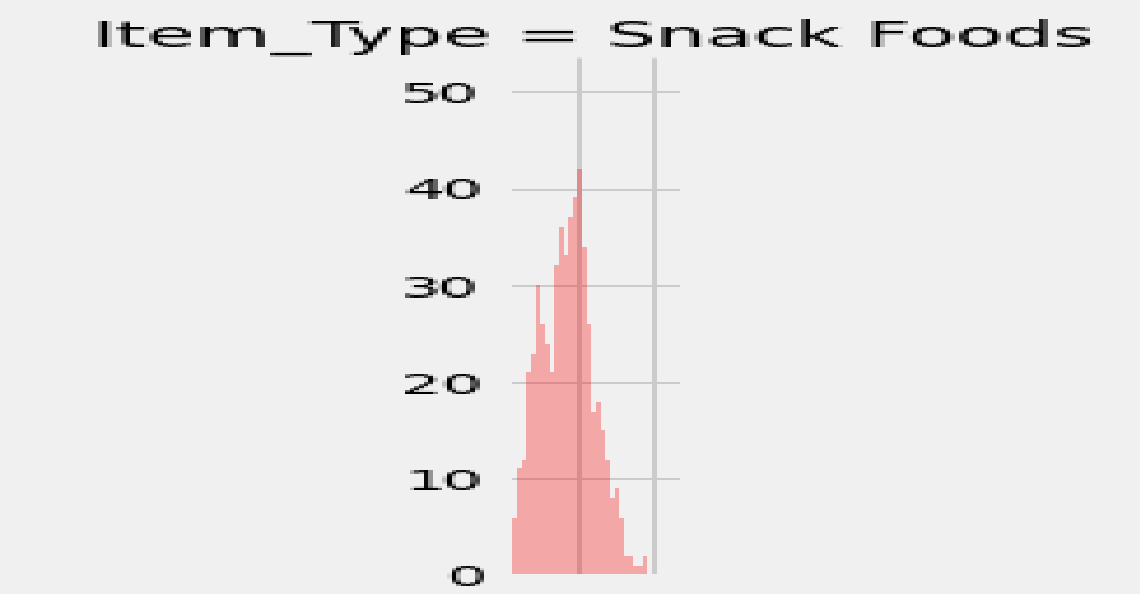

Item_Type = Meat
50
40
30
20
10
0

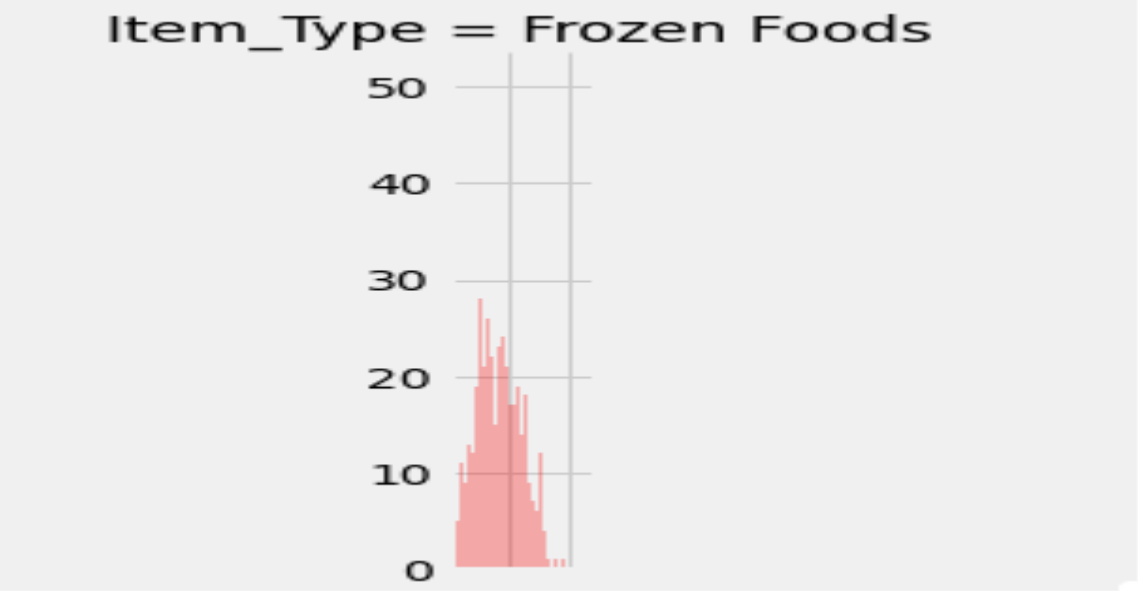

Item_Type = Fruits and Vegetables
50
40
30
20
10
0

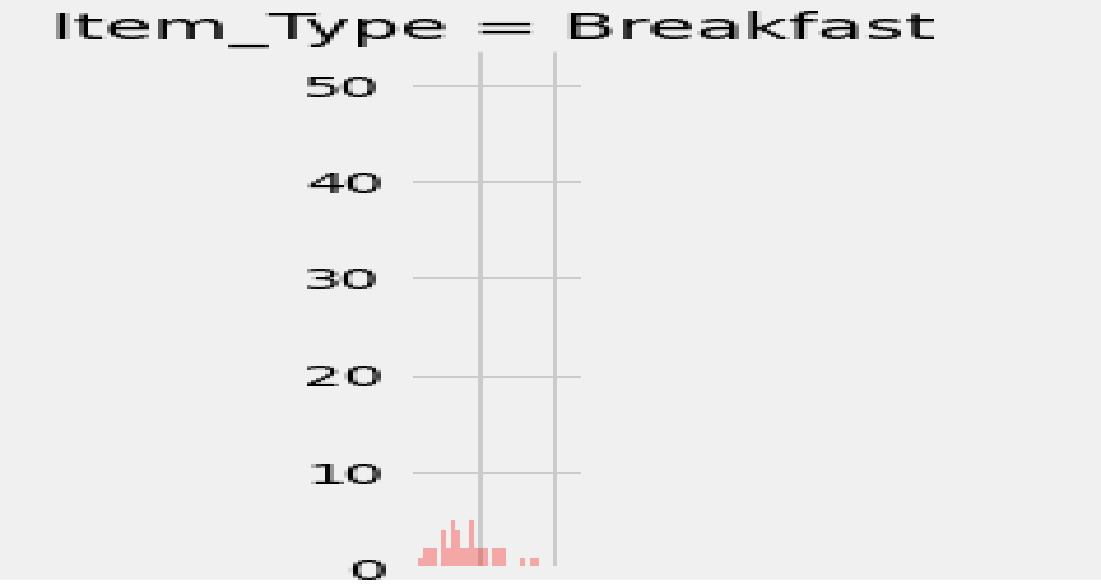


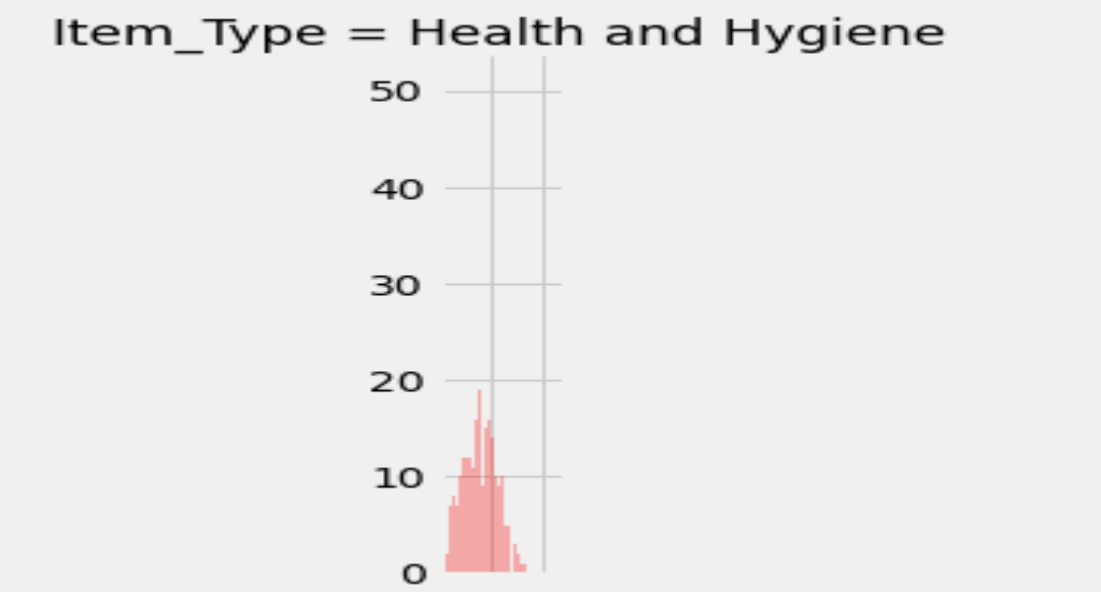


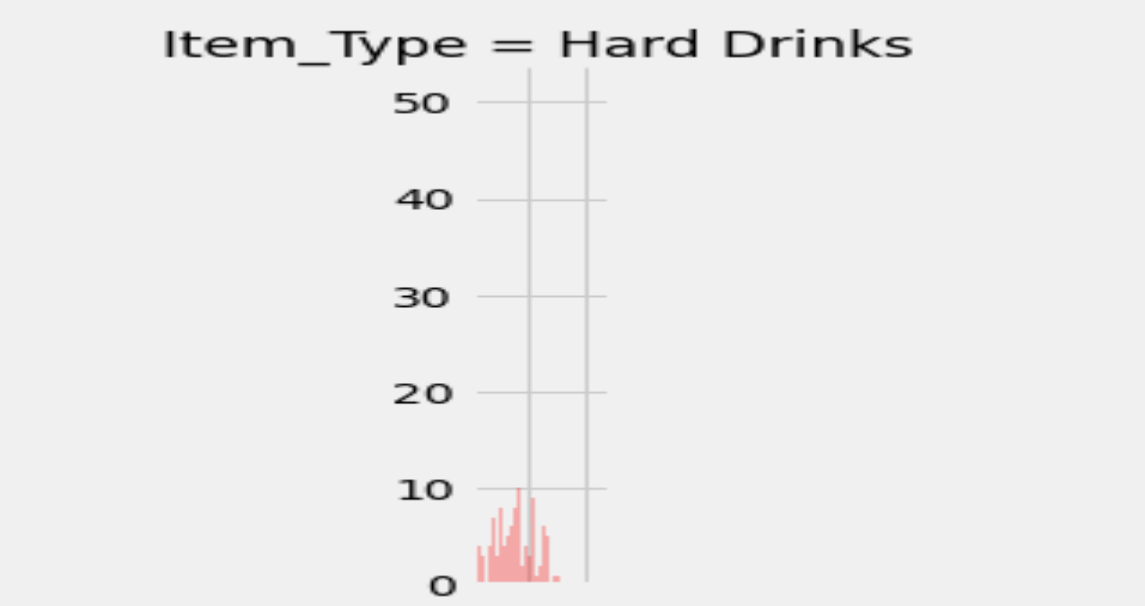
relationship between Item Type/Item Identifier and Outlet_sales_sqrt column

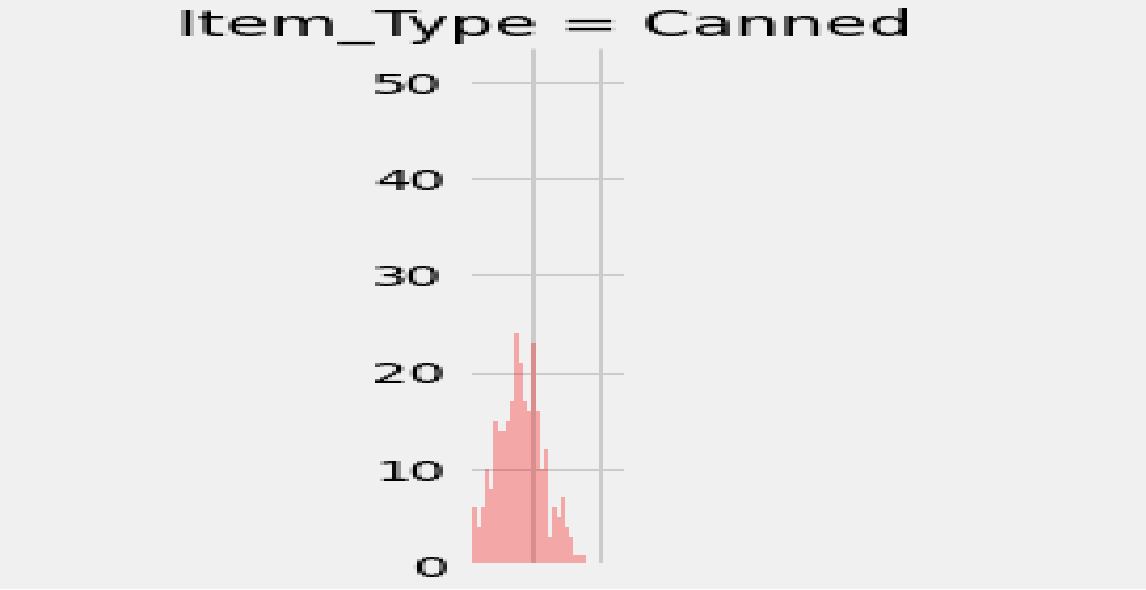

Item_Type = Snack Foods
50
40
30
20
10
0

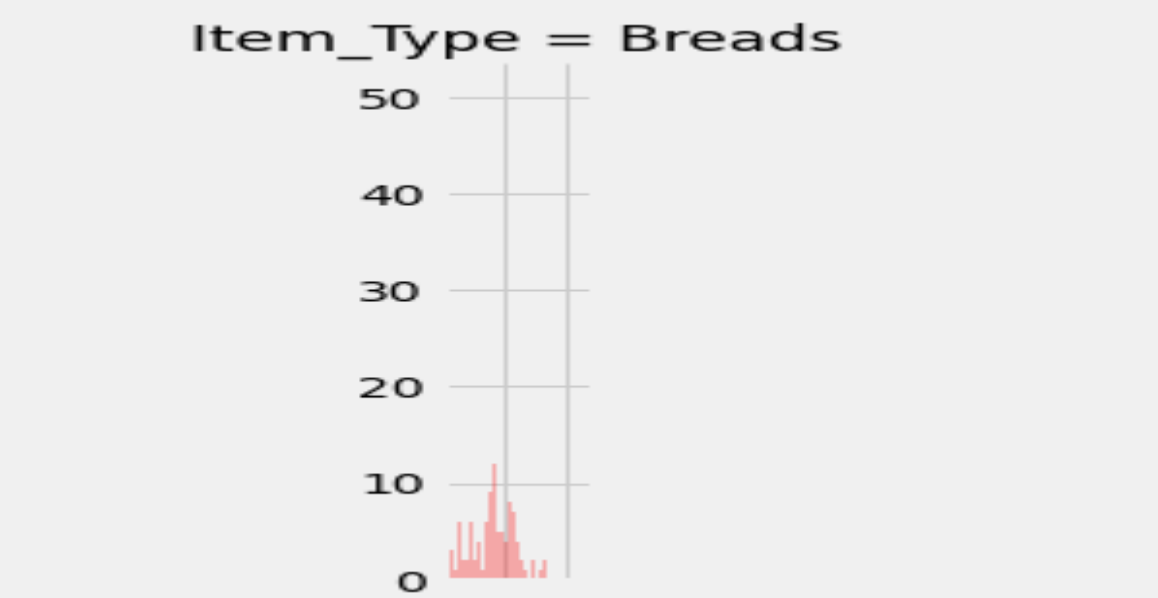

Item_Type = Frozen Foods
50
40
30
20
10
0

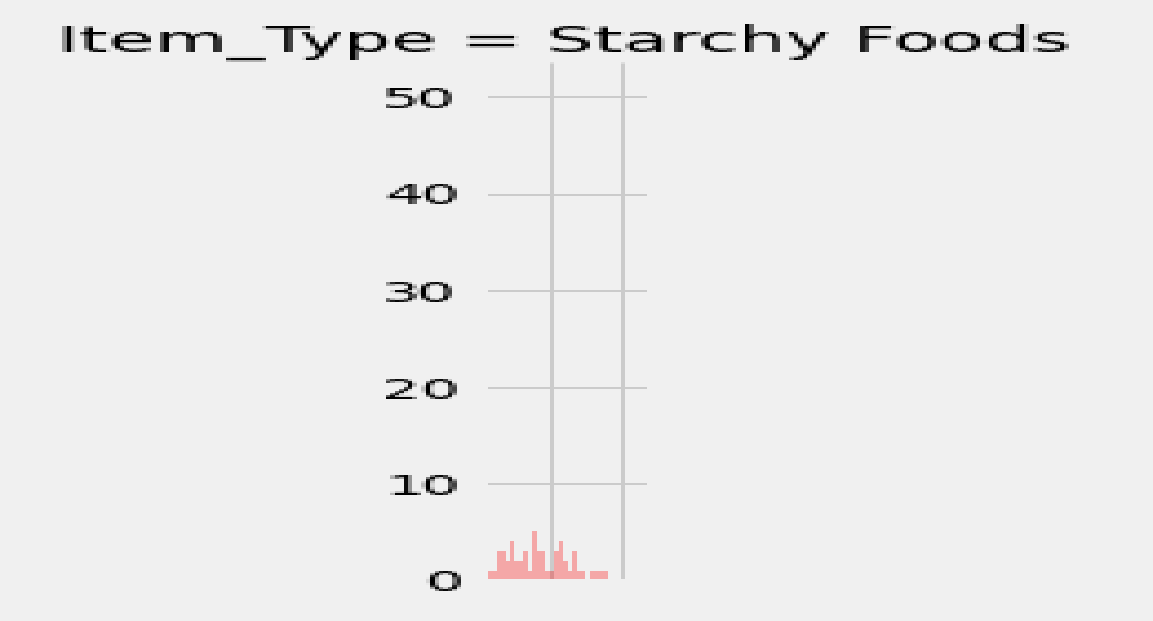

Item_Type = Breakfast
50
40
30
20
10
0

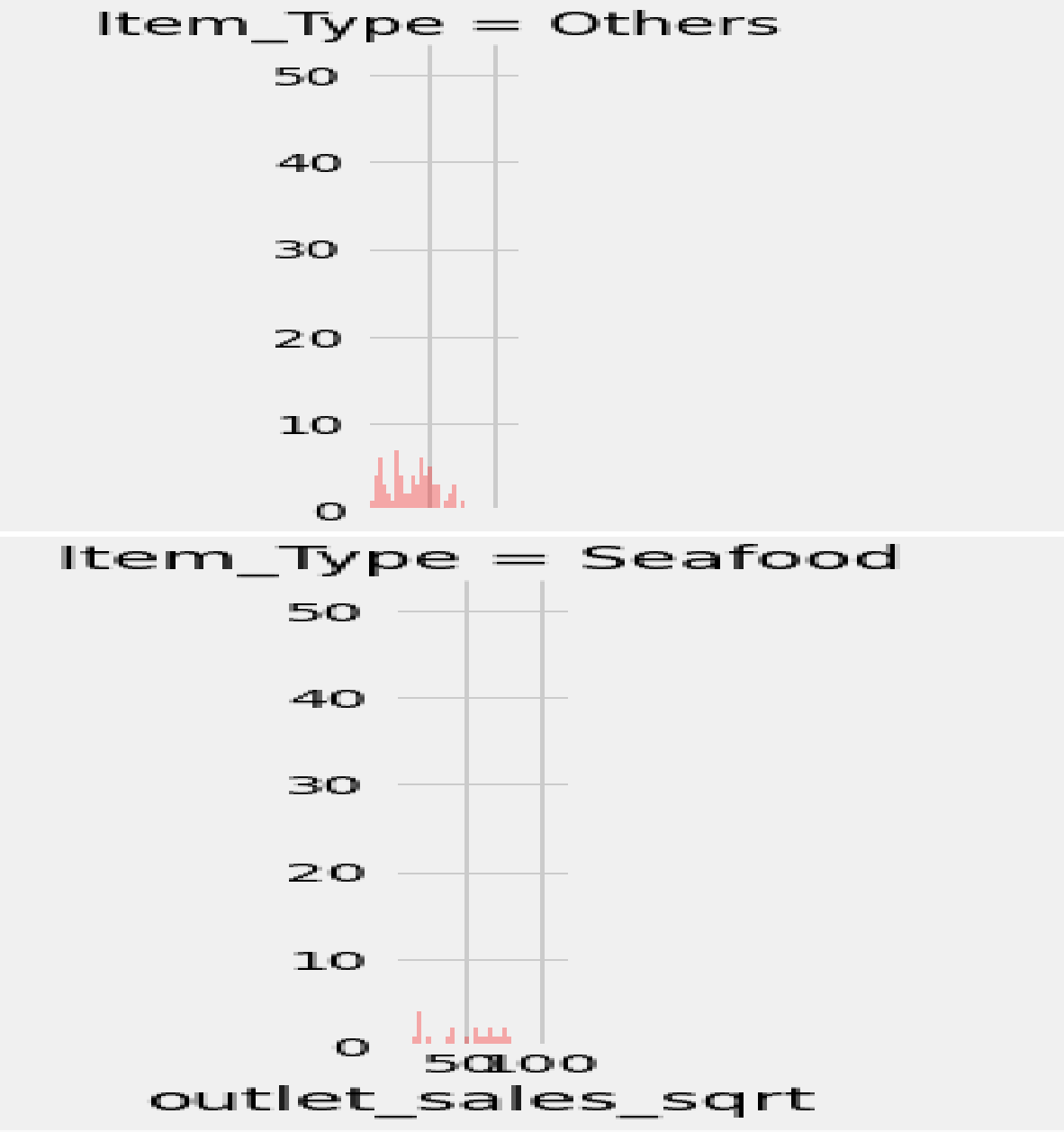

Item_Type = Health and Hygiene
50
40
30
20
10
0


Item_Type = Hard Drinks
50
40
30
20
10
0


Item_Type = Canned
50
40
30
20
10
0


Item_Type = Breads
50
40
30
20
10
0


Item_Type = Starchy Foods
50
40
30
20
10
0


Item_Type = Others
50
40
30
20
10
0
Item_Type = Seafood
50
40
30
20
10
0
50
100
outlet_sales_sqrt

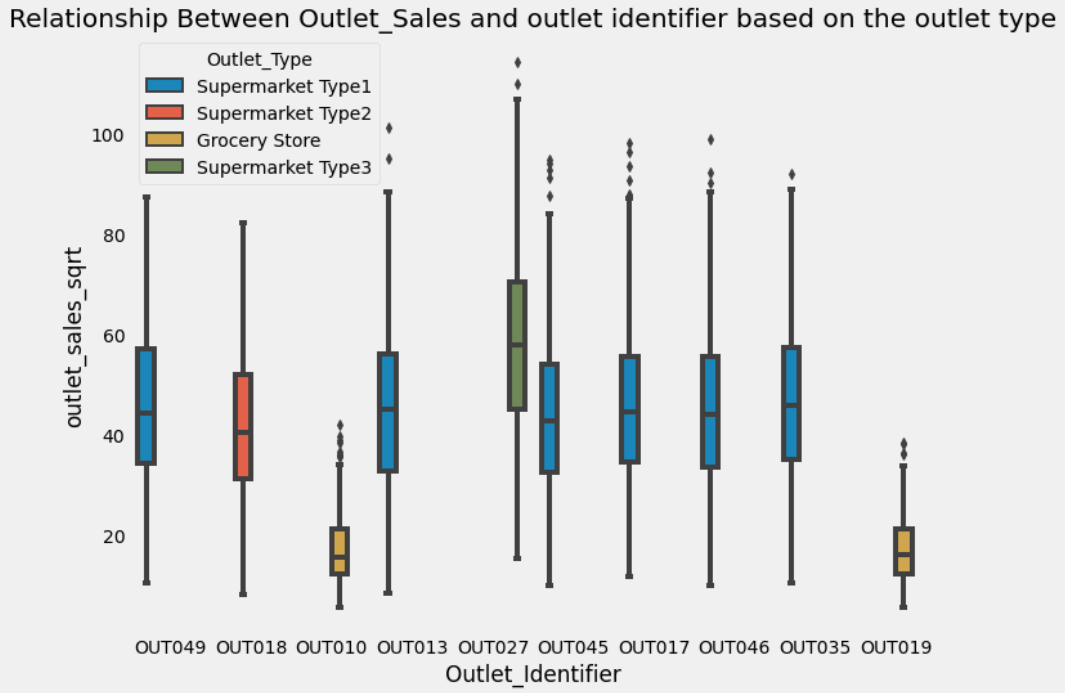
In [62]:
```
train.head(5)
```

Out[62]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	

In [63]:
```
plt.figure(figsize=(10,8))
plt.grid()
plt.title('Relationship Between Outlet_Sales and outlet identifier based on the outlet type')
sns.boxplot(x='Outlet_Identifier',y='outlet_sales_sqrt',data=train,hue='Outlet_Type')
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x1bf35680730>



Categorical Variables — One Hot Encoding

In [64]:
```
#Import library:
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
#New variable for outlet
data['Outlet'] = le.fit_transform(data['Outlet_Identifier'])
var_mod = ['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Item_Type_Combined','Outlet_Type','Outlet']
for i in var_mod:
    data[i] = le.fit_transform(data[i])
```

In [65]:
```
#Dummy Variables:
data = pd.get_dummies(data, columns =['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Outlet_Type','Item_Type_Combine
d','Outlet'])
data.dtypes
```

Out[65]:

Item_Identifier	object
Item_Weight	float64
Item_Visibility	float64
Item_Type	object
Item_MRP	float64
Outlet_Identifier	object
Outlet_Establishment_Year	int64
Item_Outlet_Sales	float64
source	object
Outlet_Years	int64
Item_Visibility_MeanRatio	float64
Item_Fat_Content_0	uint8
Item_Fat_Content_1	uint8
Item_Fat_Content_2	uint8
Outlet_Location_Type_0	uint8
Outlet_Location_Type_1	uint8
Outlet_Location_Type_2	uint8
Outlet_Size_0	uint8
Outlet_Size_1	uint8
Outlet_Size_2	uint8
Outlet_Type_0	uint8
Outlet_Type_1	uint8
Outlet_Type_2	uint8
Outlet_Type_3	uint8
Item_Type_Combined_0	uint8
Item_Type_Combined_1	uint8
Item_Type_Combined_2	uint8
Outlet_0	uint8
Outlet_1	uint8
Outlet_2	uint8
Outlet_3	uint8
Outlet_4	uint8
Outlet_5	uint8
Outlet_6	uint8

```
Outlet_7                    uint8
Outlet_8                    uint8
Outlet_9                    uint8
dtype: object
```

Exporting Data

In [66]:
```
#Drop the columns which have been converted to different types:
data.drop(['Item_Type','Outlet_Establishment_Year'],axis=1,inplace=True)
#Divide into test and train:
train = data.loc[data['source']=="train"]
test = data.loc[data['source']=="test"]
#Drop unnecessary columns:
test.drop(['Item_Outlet_Sales','source'],axis=1,inplace=True)
train.drop(['source'],axis=1,inplace=True)
#Export files as modified versions:
train.to_csv("data/train_modified.csv",index=False)
test.to_csv("data/test_modified.csv",index=False)
```

Model Building

In [67]:
```
train_df = pd.read_csv('data/train_modified.csv')
test_df = pd.read_csv('data/test_modified.csv')
```

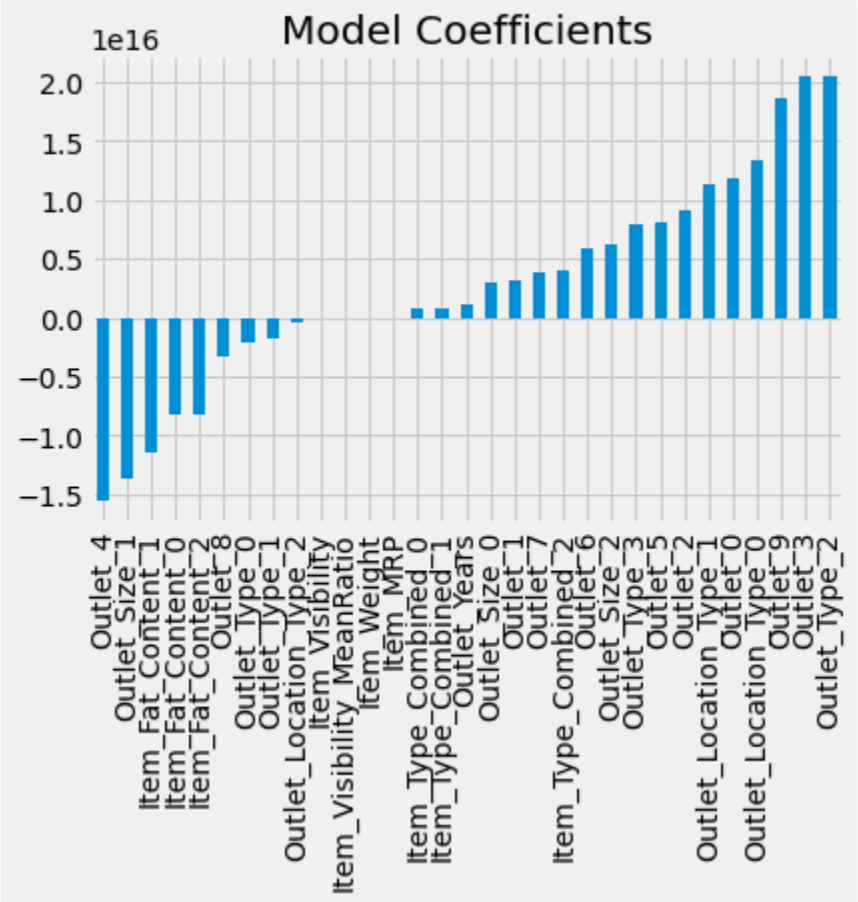
In [68]:
```
#Define target and ID columns:
target = 'Item_Outlet_Sales'
IDcol = ['Item_Identifier','Outlet_Identifier']
from sklearn import metrics
from sklearn.model_selection import cross_val_score
def modelfit(alg, dtrain, dtest, predictors, target, IDcol, filename):
    #Fit the algorithm on the data
    alg.fit(dtrain[predictors], dtrain[target])

    #Predict training set:
    dtrain_predictions = alg.predict(dtrain[predictors])

    #Perform cross-validation:
    cv_score = cross_val_score(alg, dtrain[predictors], dtrain[target], cv=20, scoring='neg_mean_squared_error')
    cv_score = np.sqrt(np.abs(cv_score))

    #Print model report:
    print ("\nModel Report")
    print ("RMSE : %.4g" % np.sqrt(metrics.mean_squared_error(dtrain[target].values, dtrain_predictions)))
    print ("CV Score : Mean - %.4g | Std - %.4g | Min - %.4g | Max - %.4g" % (np.mean(cv_score),np.std(cv_score),np.min(cv_scor
e),np.max(cv_score)))

    #Predict on testing data:
    dtest[target] = alg.predict(dtest[predictors])

    #Export submission file:
    IDcol.append(target)
    submission = pd.DataFrame({ x: dtest[x] for x in IDcol})
    submission.to_csv(filename, index=False)
```

Linear Regression Model

In [69]:
```
from sklearn.linear_model import LinearRegression
LR = LinearRegression(normalize=True)
predictors = train_df.columns.drop(['Item_Outlet_Sales','Item_Identifier','Outlet_Identifier'])
modelfit(LR, train_df, test_df, predictors, target, IDcol, 'LR.csv')
coef1 = pd.Series(LR.coef_, predictors).sort_values()
coef1.plot(kind='bar', title='Model Coefficients')
```

```
Model Report
RMSE : 1127
CV Score : Mean - 1129 | Std - 43.4 | Min - 1075 | Max - 1211
```

Out[69]: <matplotlib.axes._subplots.AxesSubplot at 0x1bf382d5700>

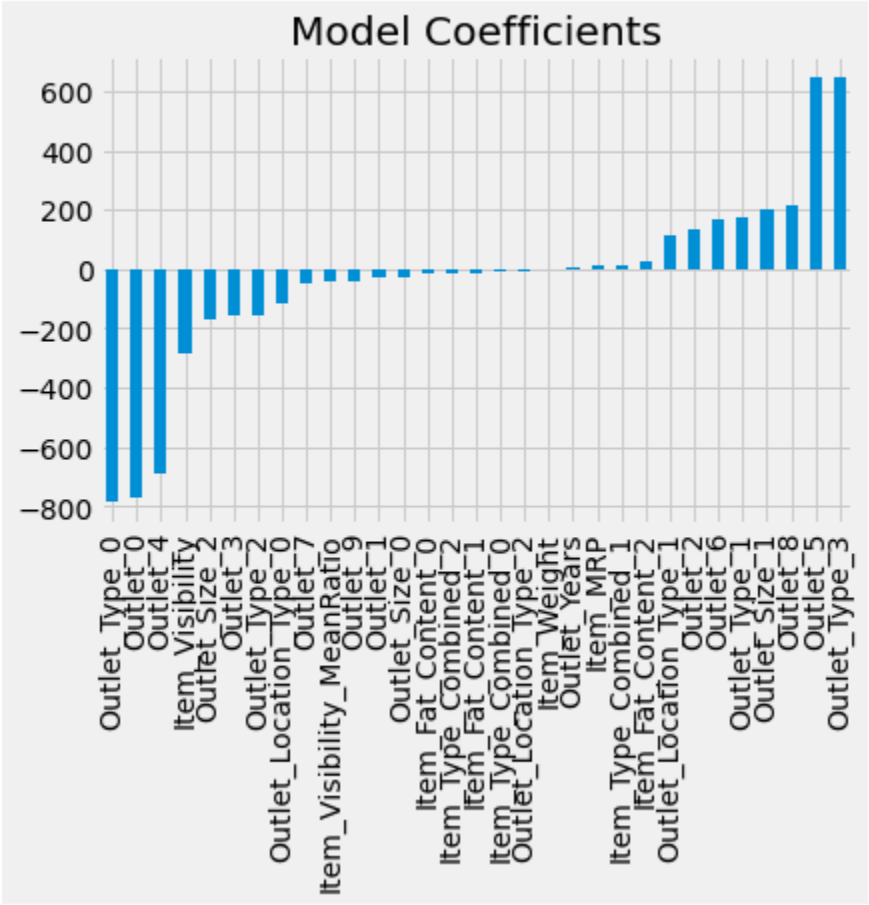


Ridge Regression Model

In [70]:
```
from sklearn.linear_model import Ridge
RR = Ridge(alpha=0.05,normalize=True)
modelfit(RR, train_df, test_df, predictors, target, IDcol, 'RR.csv')
coef2 = pd.Series(RR.coef_, predictors).sort_values()
coef2.plot(kind='bar', title='Model Coefficients')
```

```
Model Report
RMSE : 1128
CV Score : Mean - 1130 | Std - 44.76 | Min - 1076 | Max - 1217
```

Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x1bf33d7e280>

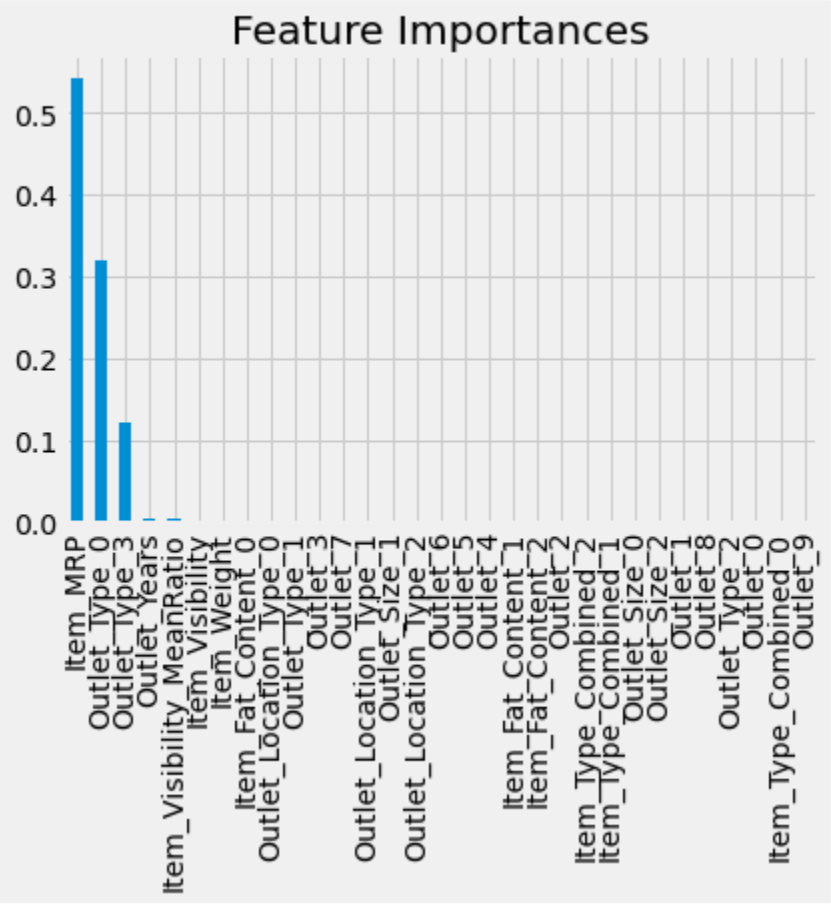


Decision Tree Model

In [71]:
```
from sklearn.tree import DecisionTreeRegressor
DT = DecisionTreeRegressor(max_depth=15, min_samples_leaf=100)
modelfit(DT, train_df, test_df, predictors, target, IDcol, 'DT.csv')
coef3 = pd.Series(DT.feature_importances_, predictors).sort_values(ascending=False)
coef3.plot(kind='bar', title='Feature Importances')
```

```
Model Report
RMSE : 1058
CV Score : Mean - 1092 | Std - 44.9 | Min - 1015 | Max - 1184
```

Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x1bf385dd8b0>

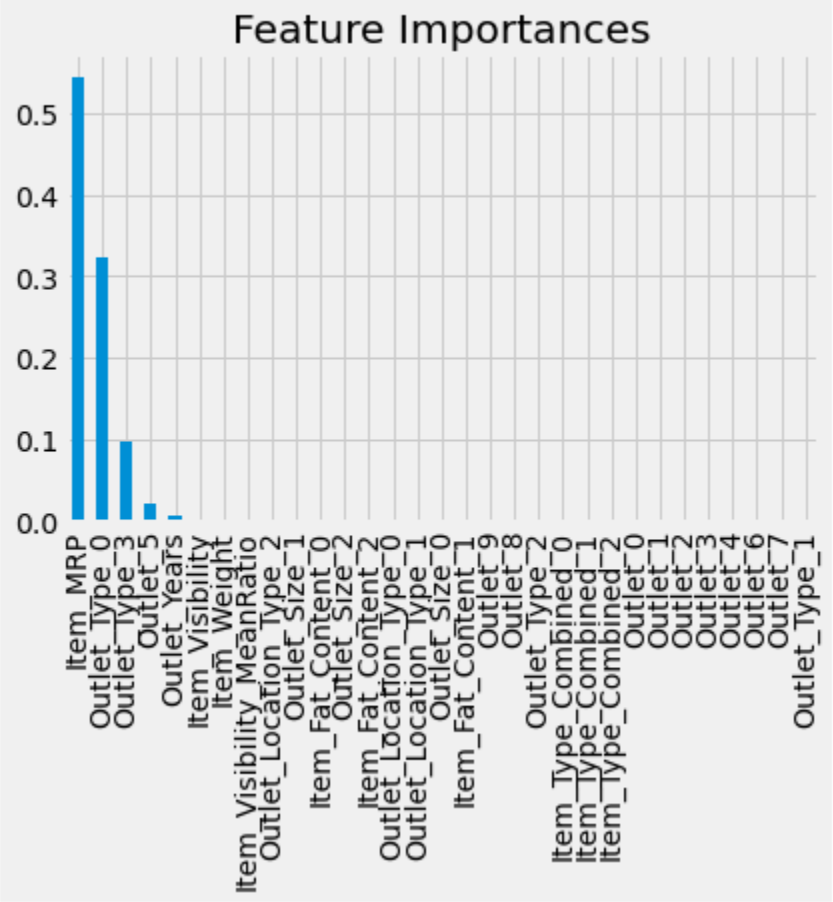


Random Forrest Model

In [72]:
```
RF = DecisionTreeRegressor(max_depth=8, min_samples_leaf=150)
modelfit(RF, train_df, test_df, predictors, target, IDcol, 'RF.csv')
coef4 = pd.Series(RF.feature_importances_, predictors).sort_values(ascending=False)
coef4.plot(kind='bar', title='Feature Importances')
```

```
Model Report
RMSE : 1068
CV Score : Mean - 1098 | Std - 43.69 | Min - 1034 | Max - 1182
```

Out[72]: `<matplotlib.axes._subplots.AxesSubplot at 0x1bf3844ca30>`



XGBoost Model

In [73]:
```
!pip install xgboost
from xgboost import XGBRegressor
my_model = XGBRegressor(n_estimators=1000, learning_rate=0.05)
my_model.fit(train_df[predictors], train_df[target], early_stopping_rounds=5,
             eval_set=[(test_df[predictors], test_df[target])], verbose=False)
```

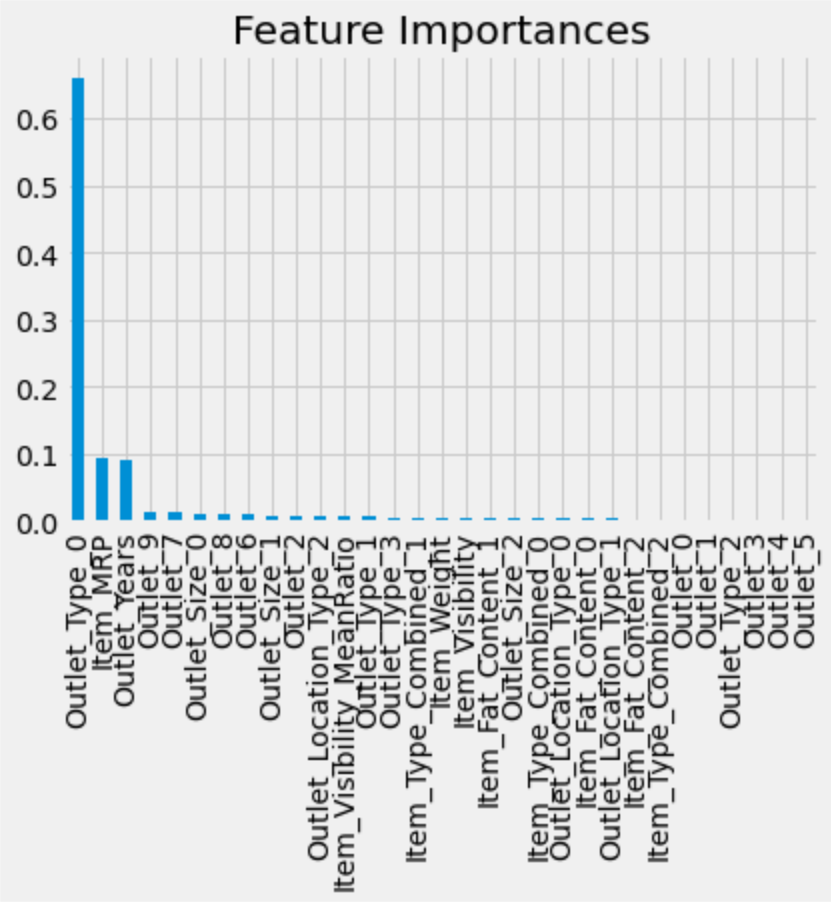
```
Requirement already satisfied: xgboost in c:\users\hp\anaconda3\lib\site-packages (1.6.1)
Requirement already satisfied: numpy in c:\users\hp\anaconda3\lib\site-packages (from xgboost) (1.18.5)
Requirement already satisfied: scipy in c:\users\hp\anaconda3\lib\site-packages (from xgboost) (1.5.0)
```

Out[73]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.05, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=1000,
             n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
             reg_alpha=0, reg_lambda=1, ...)
```

In [74]:
```
#Predict training set:
train_df_predictions = my_model.predict(train_df[predictors])
# make predictions
predictions = my_model.predict(test_df[predictors])
from sklearn.metrics import mean_absolute_error
print("Mean Absolute Error : " + str(mean_absolute_error(predictions, test_df[target])))
print("RMSE : %.4g" % np.sqrt(metrics.mean_squared_error((train_df[target]).values, train_df_predictions)))
IDcol.append(target)
submission = pd.DataFrame({ x: test_df[x] for x in IDcol})
submission.to_csv("print.csv", index=False)
coef5 = pd.Series(my_model.feature_importances_, predictors).sort_values(ascending=False)
coef5.plot(kind='bar', title='Feature Importances')
```

```
Mean Absolute Error : 140.9955906647652
RMSE : 994.5
```

Out[74]: <matplotlib.axes._subplots.AxesSubplot at 0x1bf344f2520>



CHAPTER

RESULT ANALYSIS

Discussion & Analysis

Test Results

Chapter 3

Result Analysis

Discussion & Analysis

The smallest sales were made in the smallest locales, it was discovered. However, in some circumstances, despite being type-3, it was shown that medium-sized locations achieved the most sales (there are three types of supermarkets e.g. supermarket type-1, type-2, and type-3) To improve Big mart product sales in a specific outlet, additional locations should be converted to Type 3 Supermarkets rather than the largest size site. However, the suggested model outperforms previous models in terms of future sales projections at all locations.

Observations and Data Modeling

Correlation is a statistical technique for determining the relationship between a target variable and its predictions. The target variable in this study is item sales, and its association with other variables is investigated.

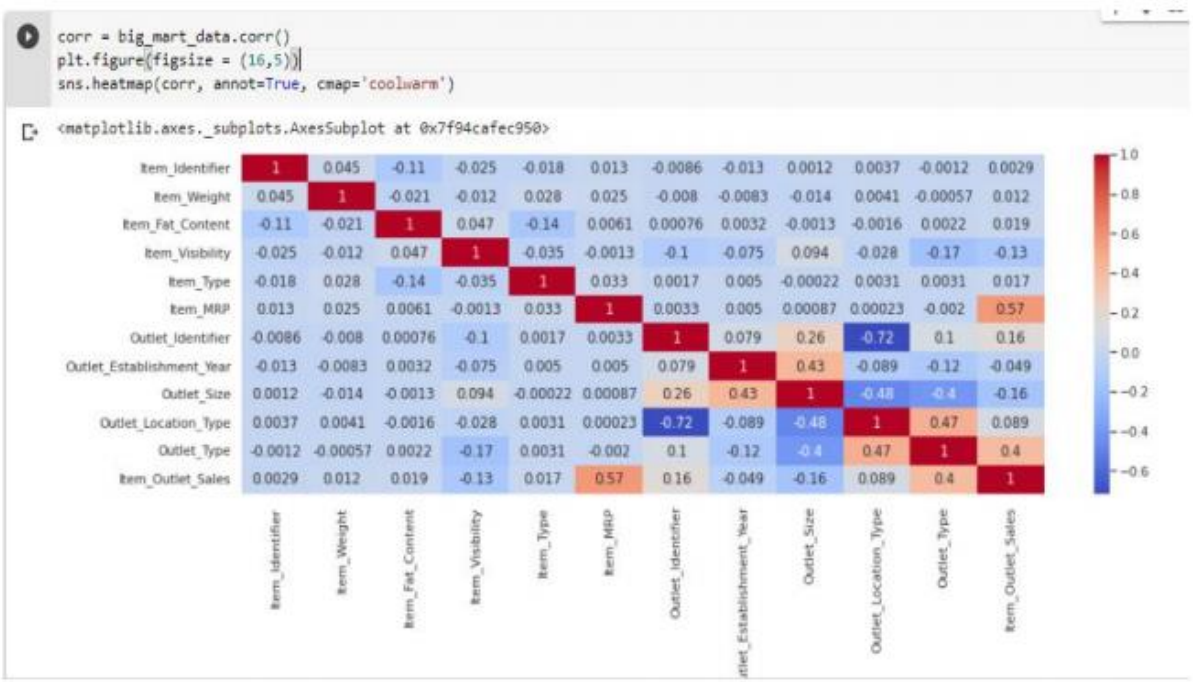


Fig 9 - Correlation among various attributes

The association between various dependent and independent variables is investigated in order to determine the next steps. After data pre-processing, variables are collected, and the following are some key observations concerning some of the variables:

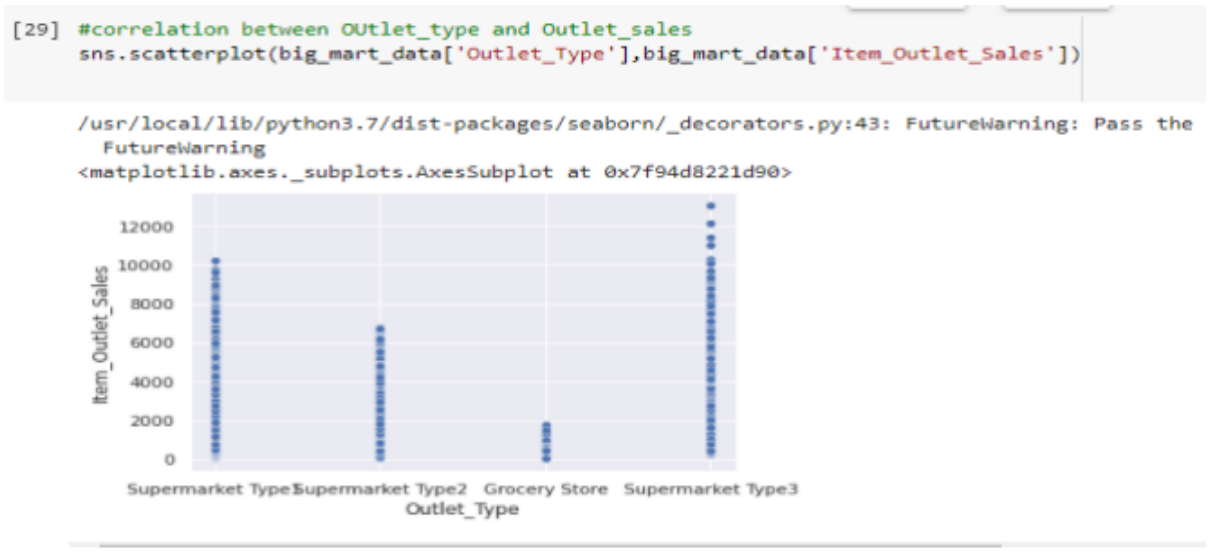


Fig 10 - Correlation among outlet type & sales

To improve the sales of products at Big Mart, more sites should be switched or changed to Supermarket Type3. Any one-stop-shopping facility, such as Big Mart, can benefit from this model by forecasting future sales at various locations.

The largest location did not provide the most revenue. The location that generated the most sales was Supermarket Type3, which had a medium size in our dataset. This outlet performed significantly better than any other outlet location in the sample, regardless of size. The XGBoost model has higher R-squared values than the average. As a result, the used model fits better and is more accurate.

```
Machine Learning Model Training

XGBoost Regressor

[41] regressor = XGBRegressor()

[42] regressor.fit(X_train, Y_train)

[56] # prediction on test data
     test_data_prediction = regressor.predict(X_test)

[57] # R squared Value
     r2_test = metrics.r2_score(Y_test, test_data_prediction)

[58] print('R Squared value = ', r2_test)

     R Squared value =  0.5806913419800449
```

Fig 11 - code showing model score of XGBoost Regressor

Check out the methods Linear Regression, Ridge Regression, Decision Tree, Random Forest, and XGBoost using sales data. Because it uses and requires real measurements at each projected data point, RMSE is frequently used in supervised learning applications when compared to other techniques.

Furthermore, modifications at a critical stage of regression model construction can boost the flexibility of the suggested approach. Experiments are also required for adequate evaluations of accuracy and resource efficiency in order to analyze and optimize correctly.

METRICS OF PERFORMANCE

MEAN ABSOLUTE ERROR

The simplest regression error statistic is mean absolute error (MAE). MAE effectively describes the typical residual magnitude.

The MAE does not reflect underperformance or overperformance of the model (whether or not the model undershoots or overshoots actual data) because we utilise the absolute value of the residual.

MEAN SQUARE ERROR

The existence of outliers in our data highlights the effect of the square term in the MSE calculation. In MAE, each residual adds proportionally to the total error, but in MSE, the error accumulates quadratically.

R2 SCORE

The R2 coefficient of determination is a statistical measure of how well regression predictions approximate real data points in regression. With a R2 of 1, the regression predictions precisely match the data.

MEDIAN ABSOLUTE ERROR

The median absolute error is particularly interesting because it is unaffected by outliers. By taking the median of all absolute discrepancies between the target and the prediction, the loss is computed.

Multiple instance parameters and various other factors can also be employed to more creatively and successfully anticipate sales.

When the parameters employed are enhanced, the accuracy of prediction systems can be greatly improved. The way the sub-models work can also help improve the system's productivity.

Because profit is directly linked to the accuracy of sales estimates, the Big Marts strive for accuracy so that the company does not lose money. In this study, we used the Xgboost algorithm, linear regression and random forest to create a model for sales prediction of a specific outlet's product, which we tested on the Big Mart 2013 dataset. Experiments show that our method produces more accurate predictions than other available methods such as decision trees and ridge regression.

CHAPTER

CONCLUSION & FUTURE WORK

Conclusion

Scope for Future Work

Chapter 4

Conclusion and Future Work

Conclusion

Every shopping mall in today's digitally linked world wants to anticipate client requests in order to avoid seasonal sales item shortages. Companies and shopping malls are getting better at anticipating product sales and consumer requests on a daily basis.

It is concluded that the Xgboost technique works better than existing models with lower MAE and RMSE and that many characteristics and elements can be used to make this sales forecast more original and successful. As the number of factors employed grows, accuracy, which is important in prediction-based systems, can grow dramatically. In addition, learning how the sub-models work might help the system become more productive.

Scope of Future Work

- The prediction's reach could be expanded in the future to include even greater scale data.
- To do so, you'll need to know the required parameters, their ideal values, and accurate sales forecasts.
- This concept can be used as a stepping stone for predicting outlet item sales in huge stores using deep learning, and it could and should be expanded upon in future studies.
- Multiple instance parameters and various other factors can also be employed to more creatively and successfully anticipate sales.
- When the parameters employed are raised, accuracy, which is important in prediction systems, can be considerably improved.
- The way the sub-models work can also improve the system's productivity.
- New methodologies will be employed to assess consumer demands better and build marketing plans.
- Hybrid models can also be constructed to improve the system's accuracy.
- Then their precision can be assessed in the same way that we did here.
- Both sellers and consumers would benefit from this system.
- An effective recommendation system can be constructed using transactional data, and customers with similar preferences will be recommended products in the store.

BIBLIOGRAPHY/REFERENCES

[1] Gopal Behera, Neeta Nain, "A Comparative Study of Big Mart Sales Prediction", September 2019.

[2] Nikita Mallick, Karan Singh, “SALES PREDICTION MODEL FOR BIG MART”, July 2020.

[3] AYESHA SYED, ASHA JYOTHI KALLURI, VENKATESWARA REDDY POCHA, VENKATA ARUN KUMAR DASARI, B.RAMASUBBAIAH, “BIG MART SALES USING MACHINE LEARNING WITH DATA ANALYSIS”, February 2020.

[4] Dr.P.Prem Delphy, Dr.S.Suresh Kumar, Mr.K.Prabhakar, “A Predictive Analysis for Sales Forecasting in Imbalanced Grocery Retail Datasets Using Various Machine Learning Algorithms”, April 2020.

[5] Nimit Jain “Big Mart Sales Prediction Using Machine Learning”, December 2021.

big mart

ORIGINALITY REPORT

19%	7%	4%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of North Texas Student Paper	5%
2	ijisrt.com Internet Source	4%
3	Submitted to Coventry University Student Paper	2%
4	"Computer Vision and Image Processing", Springer Science and Business Media LLC, 2020 Publication	1%
5	Submitted to National College of Ireland Student Paper	1%
6	Submitted to Educational Service District 105 Student Paper	1%
7	en.wikipedia.org Internet Source	1%
8	Submitted to RDI Distance Learning Student Paper	<1%
	Submitted to University of Witwatersrand	

9	Student Paper	<1%
10	doczz.net Internet Source	<1%
11	Submitted to University of Bradford Student Paper	<1%
12	es.scribd.com Internet Source	<1%
13	Submitted to Leeds Beckett University Student Paper	<1%
14	N.M Saravana Kumar, K Hariprasath, N Kaviyavarshini, A Kavinya. "A study on forecasting bigmart sales using optimized machine learning techniques", Science in Information Technology Letters, 2020 Publication	<1%
15	Submitted to University of Dammam Student Paper	<1%
16	www.coursehero.com Internet Source	<1%
17	tdx.cat Internet Source	<1%
18	journal.esrgroups.org Internet Source	<1%
	id.123dok.com	

19	Internet Source	<1%
20	www.igi-global.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		