Double-click (or enter) to edit

Project #1: Bigmart Sale Prediction

Initializing Packages and Importing Data

```
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")
```

Taking a peak at our data

```
train.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0
4	NCD19	8.93	Low Fat	0.000000	Household	53.8

```
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
```

```
train.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Ou
count	7060.000000	8523.000000	8523.000000	8523.000000	8
mean	12.857645	0.066132	140.992782	1997.831867	2
std	4.643456	0.051598	62.275067	8.371760	1
min	4.555000	0.000000	31.290000	1985.000000	
25%	8.773750	0.026989	93.826500	1987.000000	
50%	12.600000	0.053931	143.012800	1999.000000	1
75%	16.850000	0.094585	185.643700	2004.000000	3
max	21.350000	0.328391	266.888400	2009.000000	13

```
#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 entri
```

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

1. Exploratory Data Analysis (EDA)

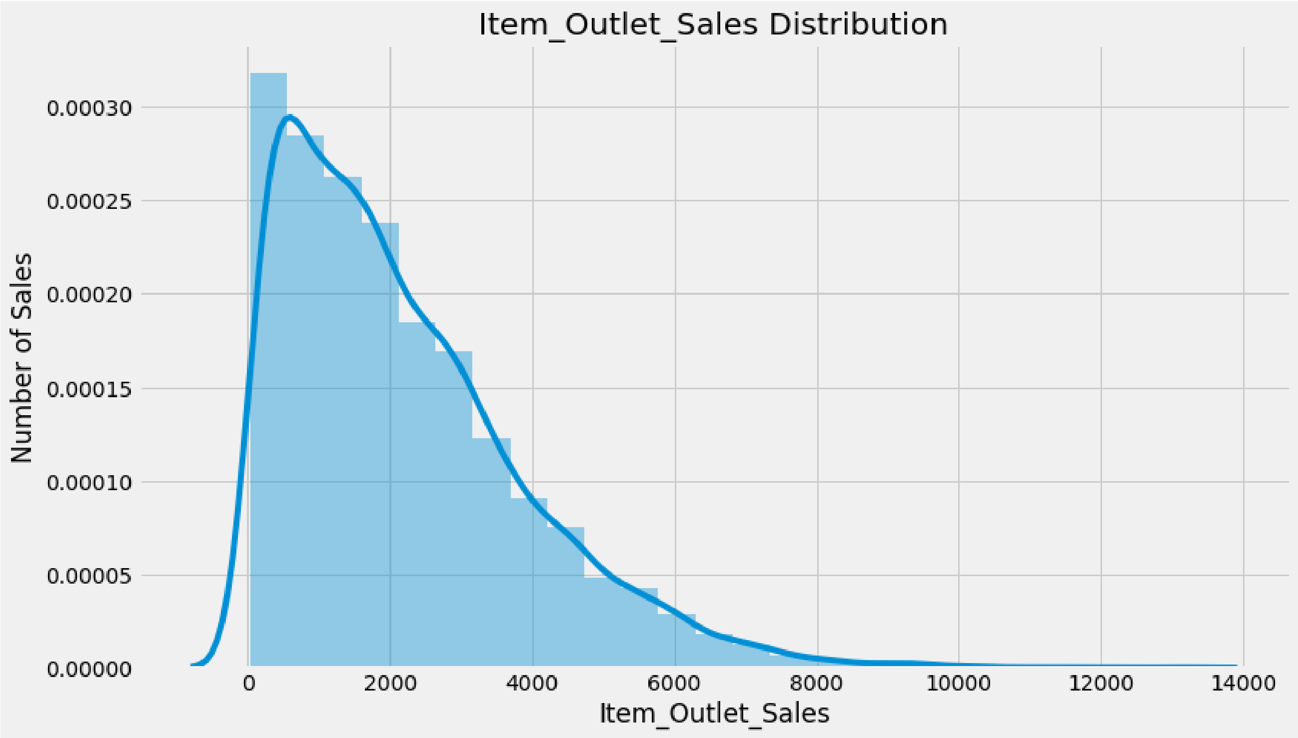
1.1. Univariate Distribution

1.1.1. Distribution of the target variable : Item_Outlet_Sales

```
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")
```

```
Text(0.5, 1.0, 'Item_Outlet_Sales Distribution')
```




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

```
Skew is: 1.1775306028542796
Kurtosis: 1.615877
```

1.1.2. Numerical Variables

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

```
Item_Weight                  float64
Item_Visibility              float64
Item_MRP                     float64
Outlet_Establishment_Year      int64
Item_Outlet_Sales            float64
dtype: object
```

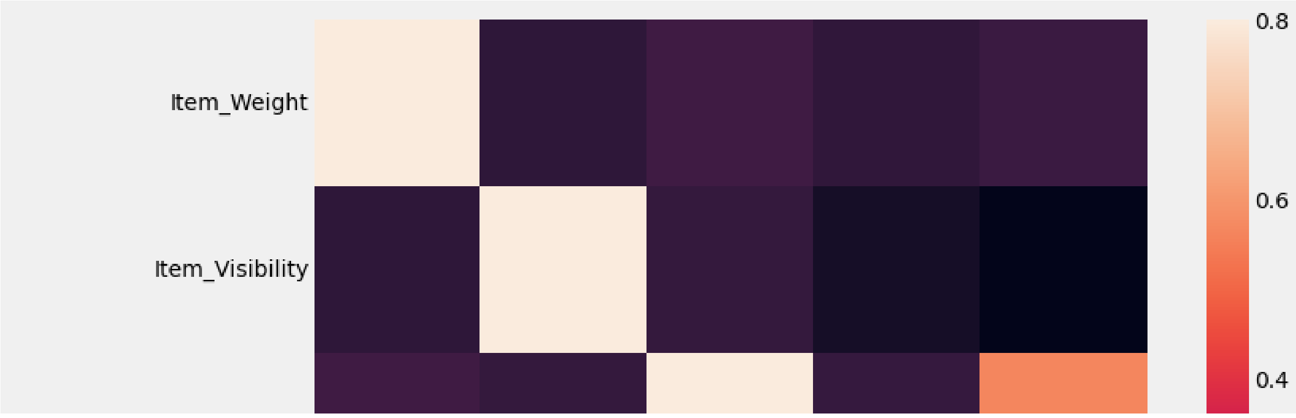
```
numeric_features.corr()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishme
Item_Weight	1.000000	-0.014048	0.027141	-(
Item_Visibility	-0.014048	1.000000	-0.001315	-(
Item_MRP	0.027141	-0.001315	1.000000	(
Outlet_Establishment_Year	-0.011588	-0.074834	0.005020	1
Item_Outlet_Sales	0.014123	-0.128625	0.567574	-(

```
corr = numeric_features.corr()

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
```

```
#correlation matrix
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corr, vmax=.8, square=True);
```



1.1.3. Categorical Variables

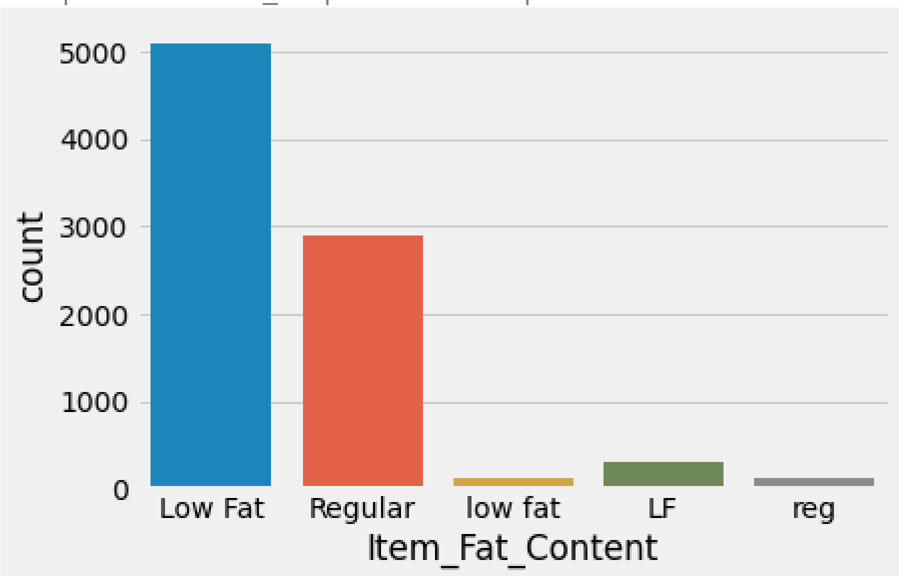
1.1.3.1. Distribution of the Item_Fat_Content


```
train.Item_Fat_Content.value_counts()
```

```
Low Fat     5089
Regular     2889
LF           316
reg          117
low fat      112
Name: Item_Fat_Content, dtype: int64
```

```
sns.countplot(train.Item_Fat_Content)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c25668a10>
```



1.1.3.2. Distribution of the Item_Type

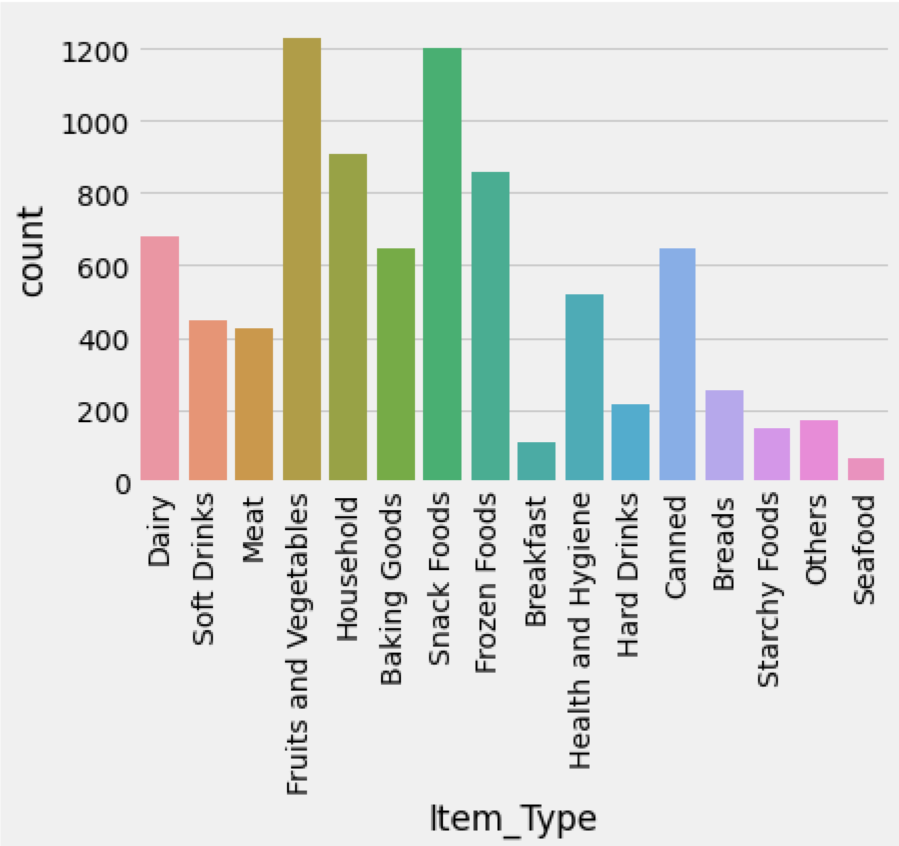
```
train.Item_Type.value_counts()
```

```
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
```

```
sns.countplot(train.Item_Type)
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
 <a list of 16 Text major ticklabel objects>)
```



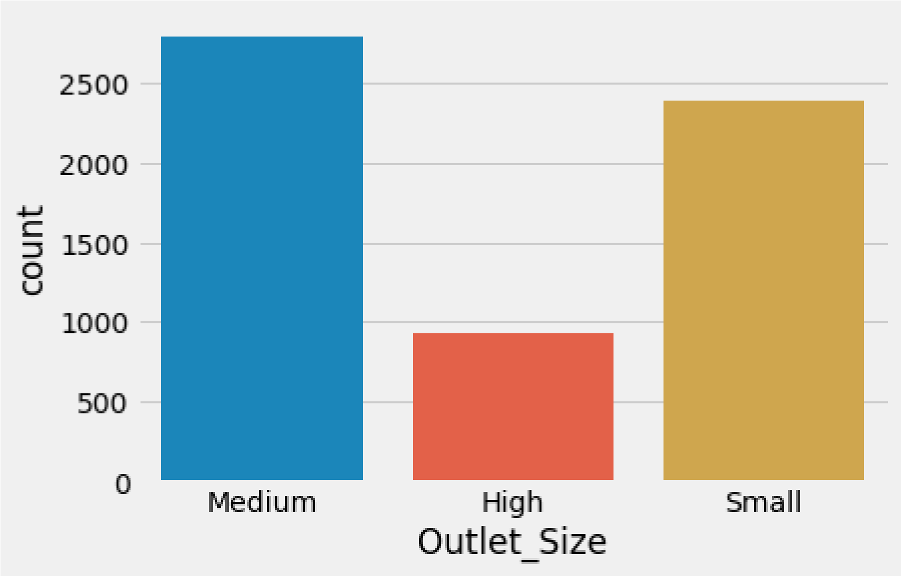
1.1.3.3. Distribution of the Outlet_Size

```
train.Outlet_Size.value_counts()
```

```
Medium    2793
Small     2388
High       932
Name: Outlet_Size, dtype: int64
```

```
sns.countplot(train.Outlet_Size)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c255e4810>
```



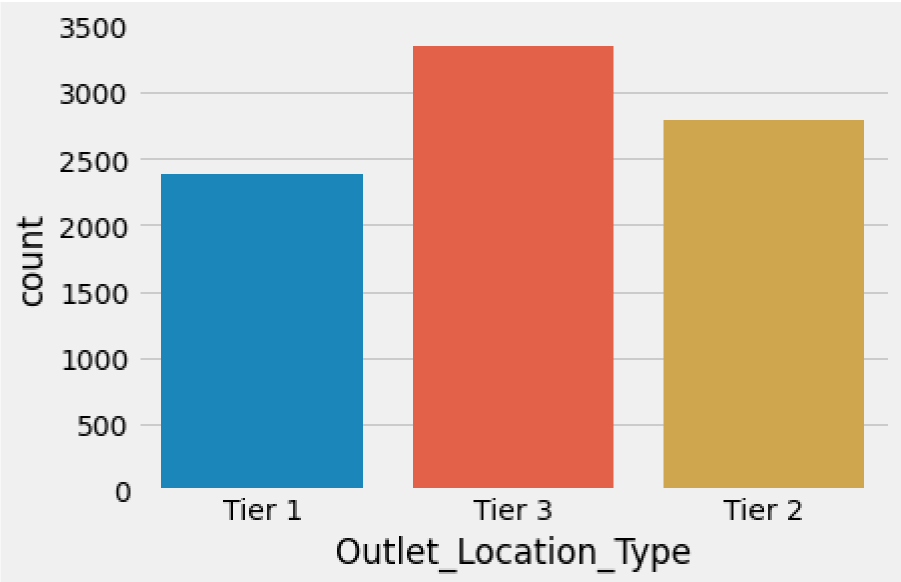
1.1.3.4. Distribution of the Outlet_Location_Type

```
train.Outlet_Location_Type.value_counts()
```

```
Tier 3    3350
Tier 2    2785
Tier 1    2388
Name: Outlet_Location_Type, dtype: int64
```

```
sns.countplot(train.Outlet_Location_Type)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c254efd50>
```



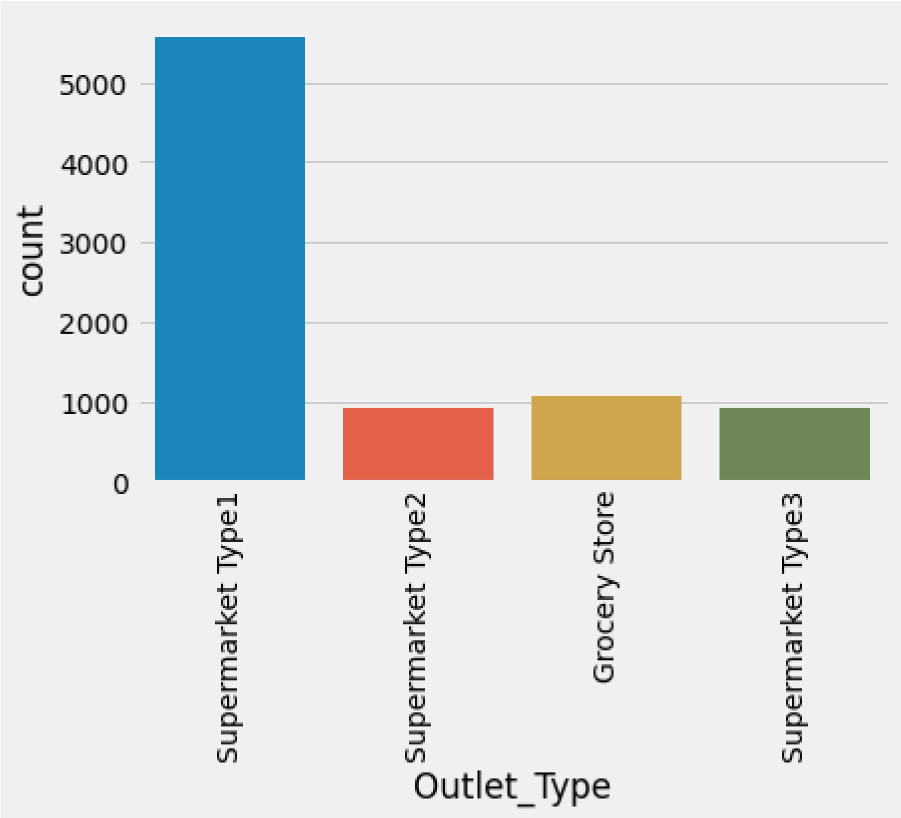
1.1.3.5. Distribution of the Outlet_Type

```
train.Outlet_Type.value_counts()
```

```
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
Name: Outlet_Type, dtype: int64
```

```
sns.countplot(train.Outlet_Type)
plt.xticks(rotation=90)
```

```
(array([0, 1, 2, 3]), <a list of 4 Text major ticklabel objects>)
```



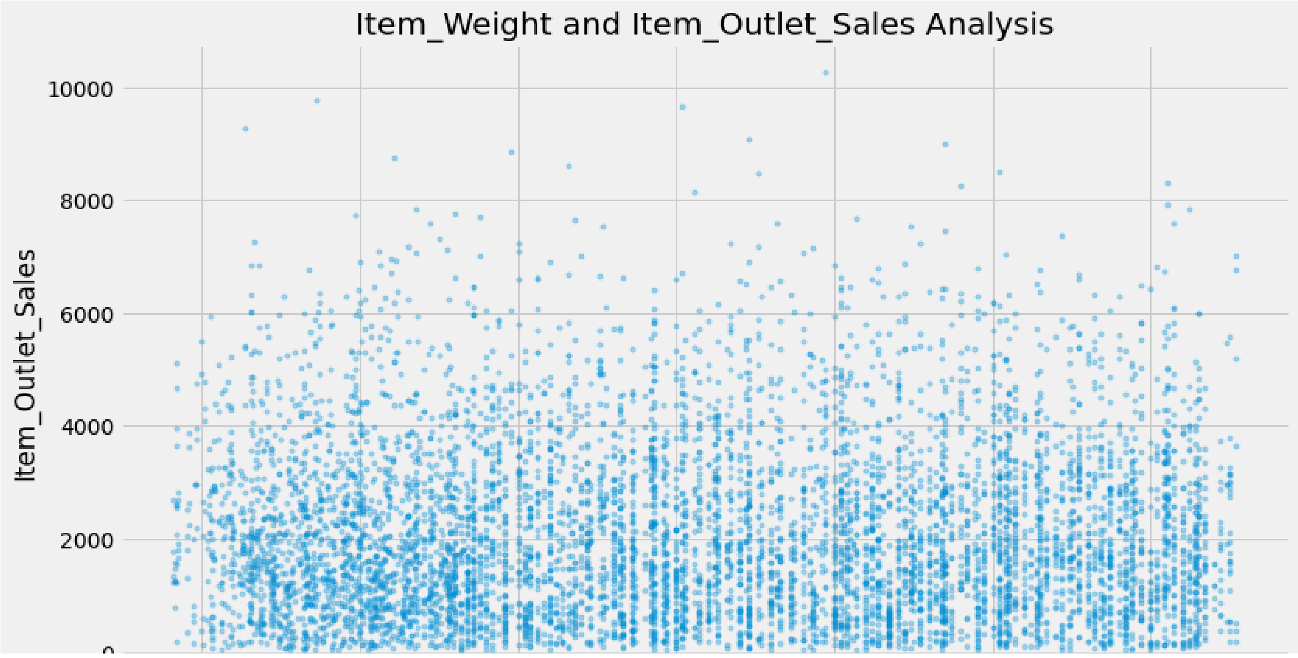
1.2. Bivariate Distribution

1.2.1. Numerical Variables

1.2.1.1. Item_Weight and Item_Outlet_Sales Analysis

```
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)
```

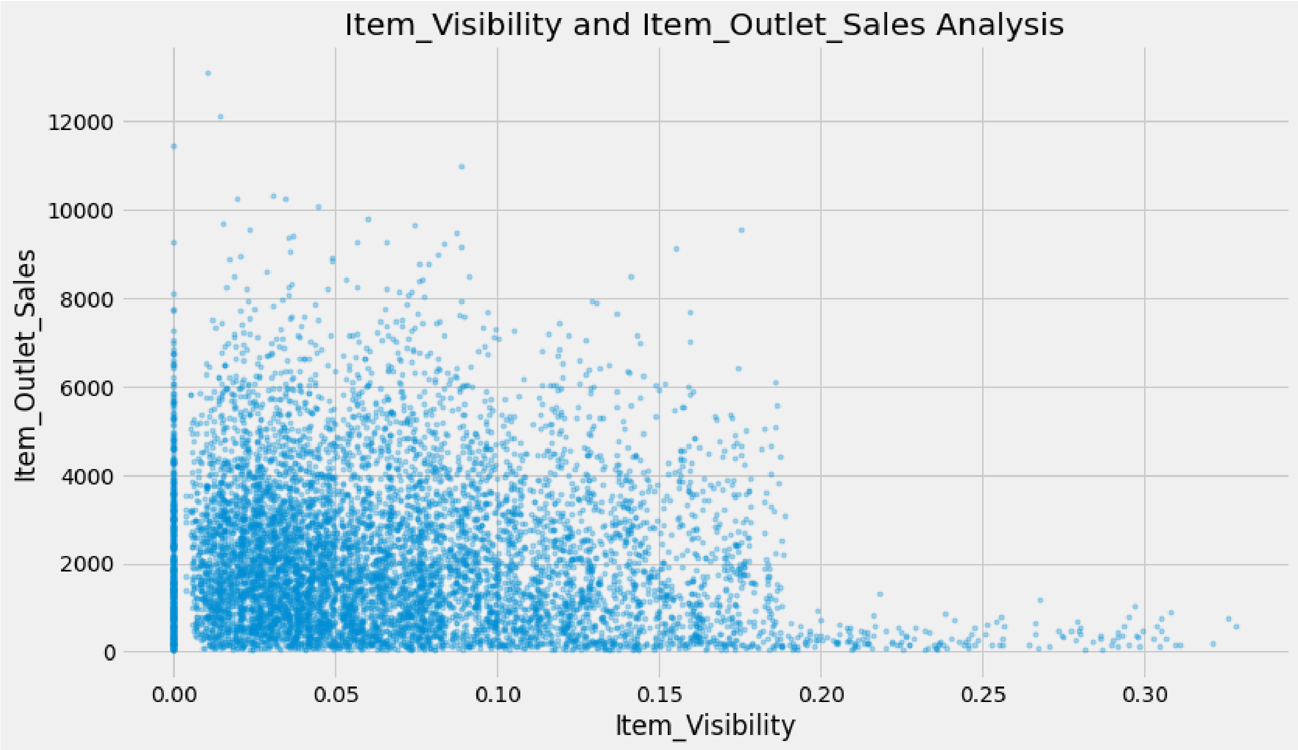
```
[<matplotlib.lines.Line2D at 0x7f1c253f1b90>]
```



1.2.1.2. Item_Visibility and Item_Outlet_Sales Analysis

```
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)
```

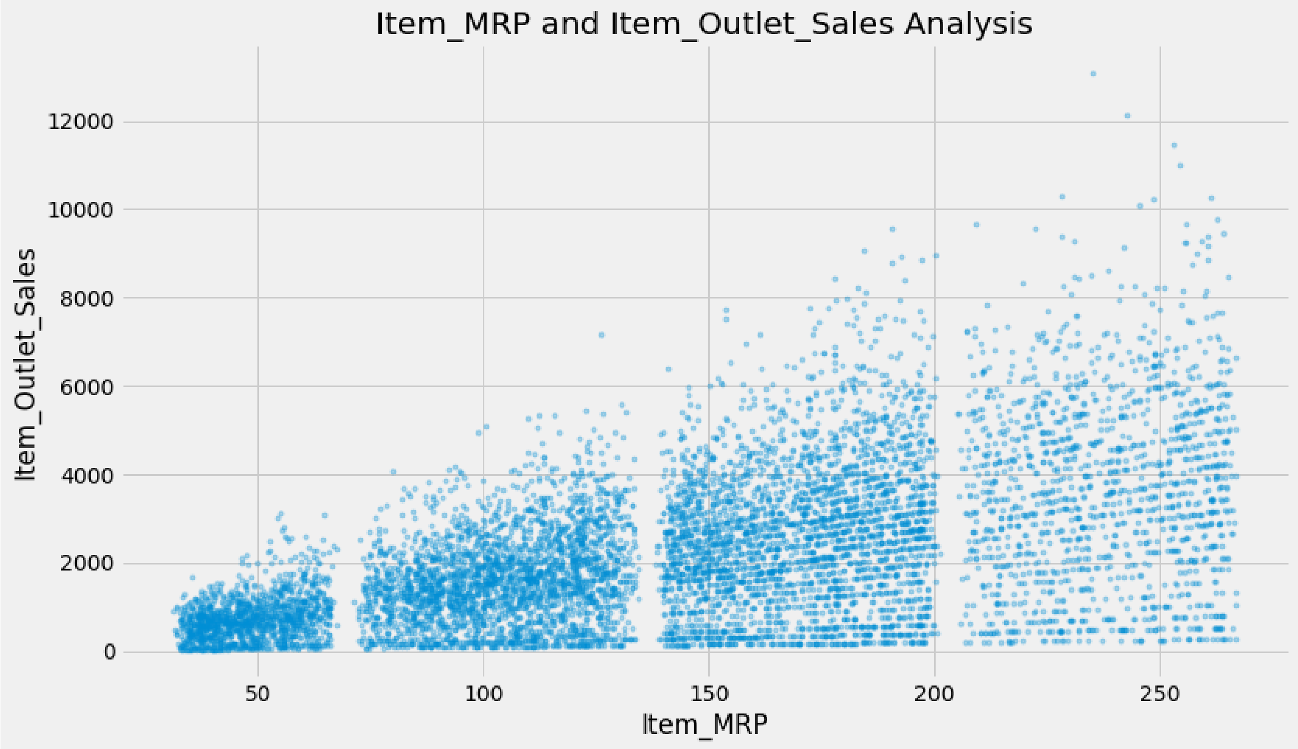
```
[<matplotlib.lines.Line2D at 0x7f1c23b68a10>]
```



1.2.1.3. Item_MRP and Item_Outlet_Sales Analysis

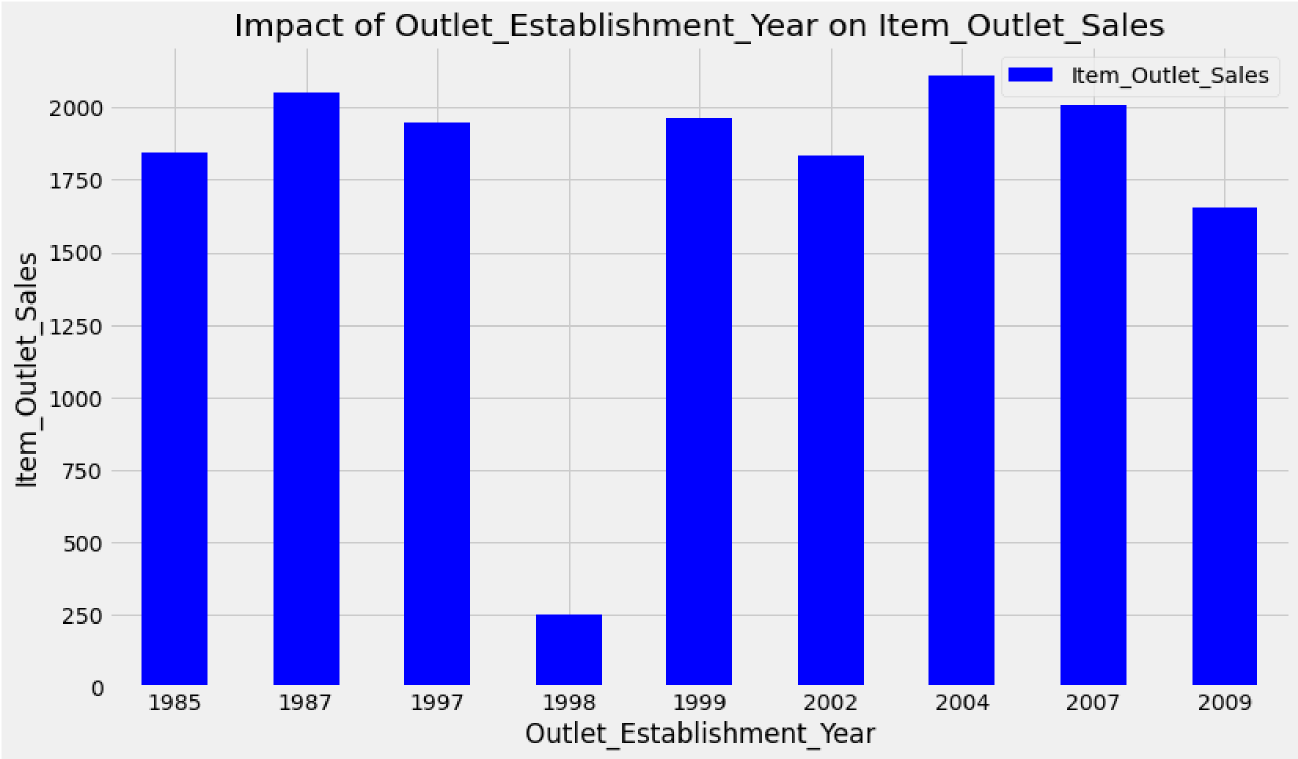
```
plt.figure(figsize=(12,7))
plt.xlabel("Item_MRP")
plt.ylabel("Item_Outlet_Sales")
plt.title("Item_MRP and Item_Outlet_Sales Analysis")
plt.plot(train.Item_MRP, train["Item_Outlet_Sales"],'.', alpha = 0.3)
```

```
[<matplotlib.lines.Line2D at 0x7f1c23ae0410>]
```



1.2.1.4. Outlet_Establishment_Year and Item_Outlet_Sales Analysis

```
Outlet_Establishment_Year_pivot = \
train.pivot_table(index='Outlet_Establishment_Year', values="Item_Outlet_Sales", aggfunc=n

Outlet_Establishment_Year_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Outlet_Establishment_Year")
plt.ylabel("Item_Outlet_Sales")
plt.title("Impact of Outlet_Establishment_Year on Item_Outlet_Sales")
plt.xticks(rotation=0)
plt.show()
```



1.2.2. Categorial Variables

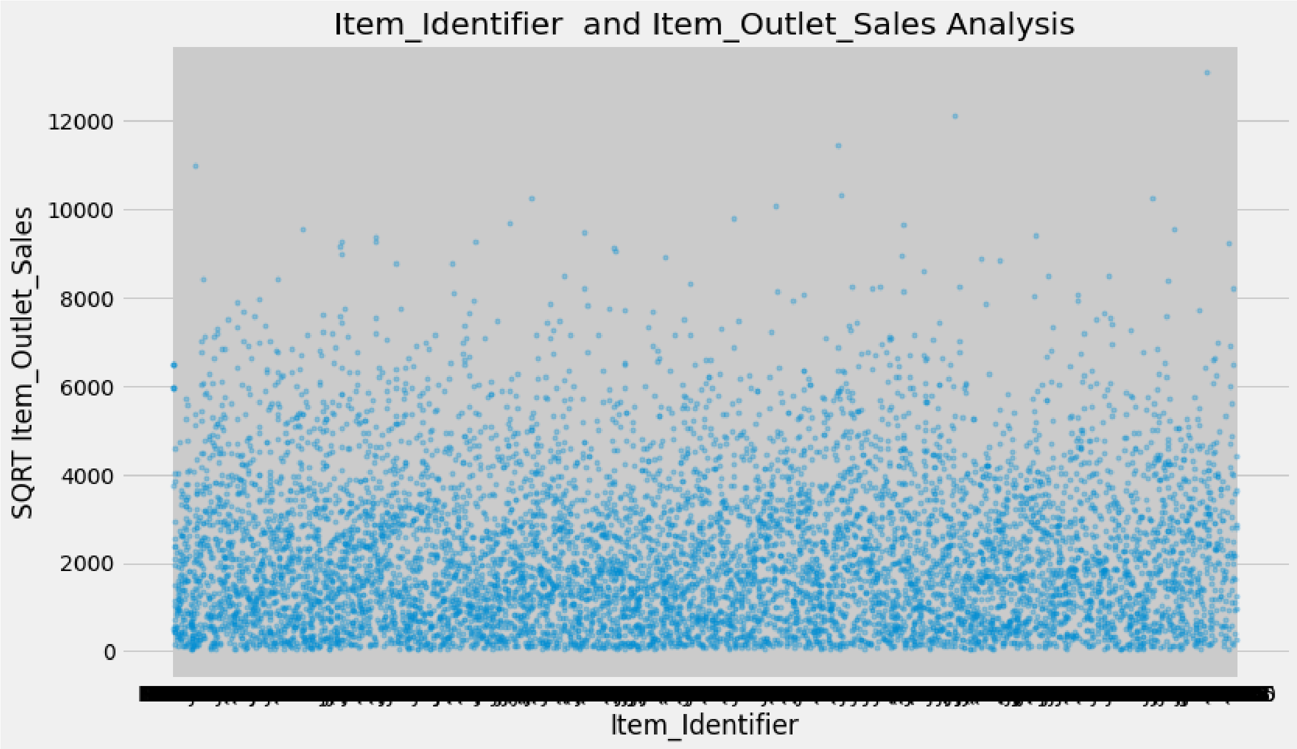
```
numeric_features = train.select_dtypes(include=[np.object])
numeric_features.dtypes
```

```
Item_Identifier         object
Item_Fat_Content        object
Item_Type               object
Outlet_Identifier       object
Outlet_Size             object
Outlet_Location_Type    object
Outlet_Type             object
dtype: object
```

1.2.2.1. Impact of Item_Identifier on Item_Outlet_Sales

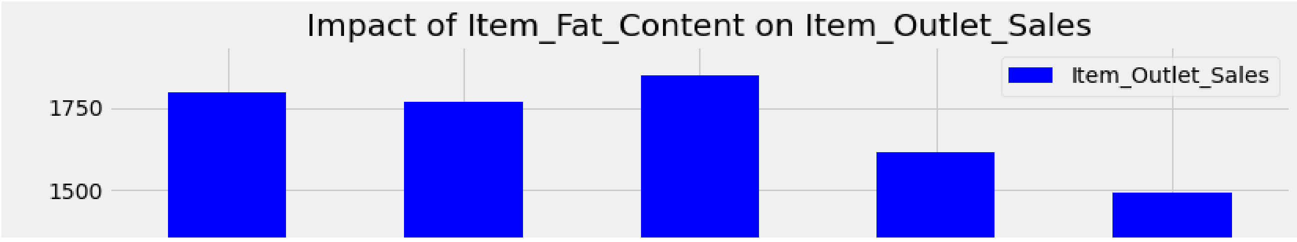
```
plt.figure(figsize=(12,7))
plt.xlabel("Item_Identifier")
plt.ylabel("SQRT Item_Outlet_Sales")
plt.title("Item_Identifier  and Item_Outlet_Sales Analysis")
plt.plot(train.Item_Identifier , train["Item_Outlet_Sales"],'.', alpha = 0.3)
```

```
[<matplotlib.lines.Line2D at 0x7f1c2567c210>]
```



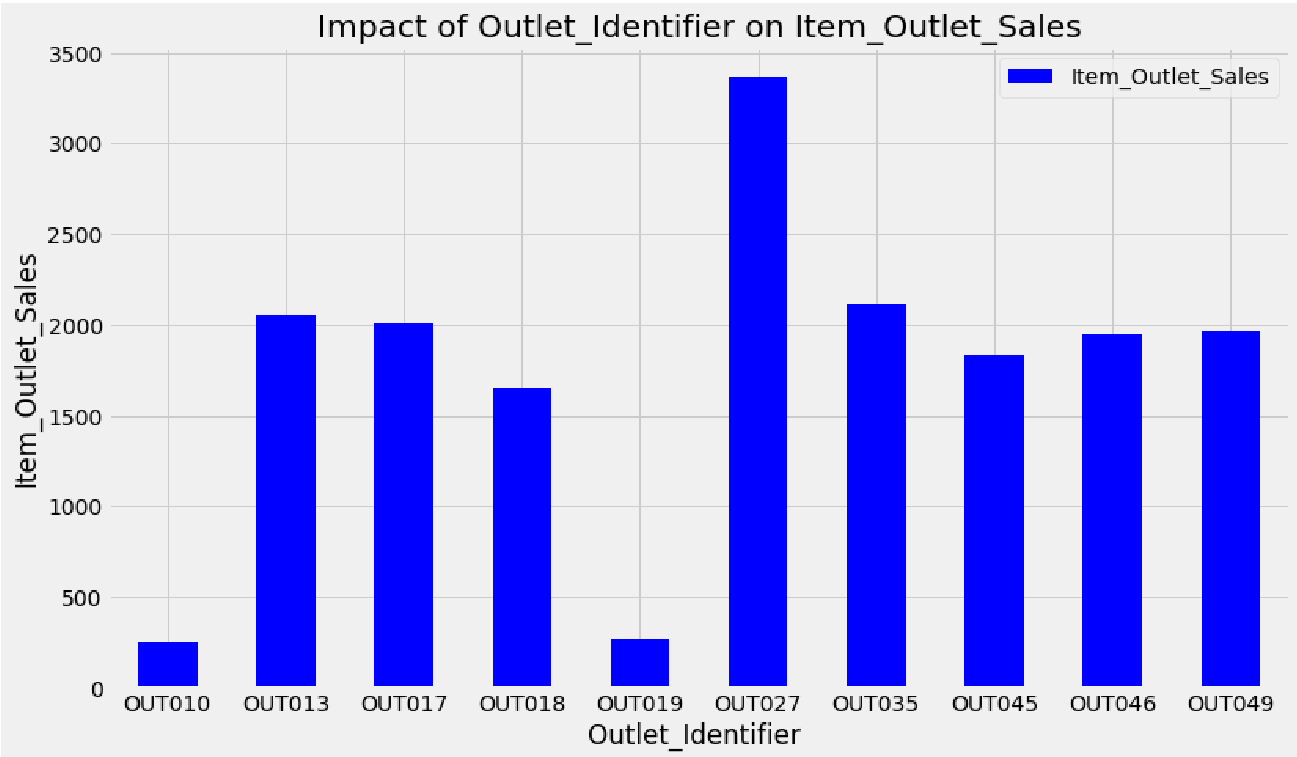
1.2.2.2. Impact of Item_Fat_Content on Item_Outlet_Sales

```
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()
```



1.2.2.3. Impact of Outlet_Identifier on Item_Outlet_Sales

```
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()
```




```
train.pivot_table(values='Outlet_Type', columns='Outlet_Identifier',aggfunc=lambda x:x.mod
```

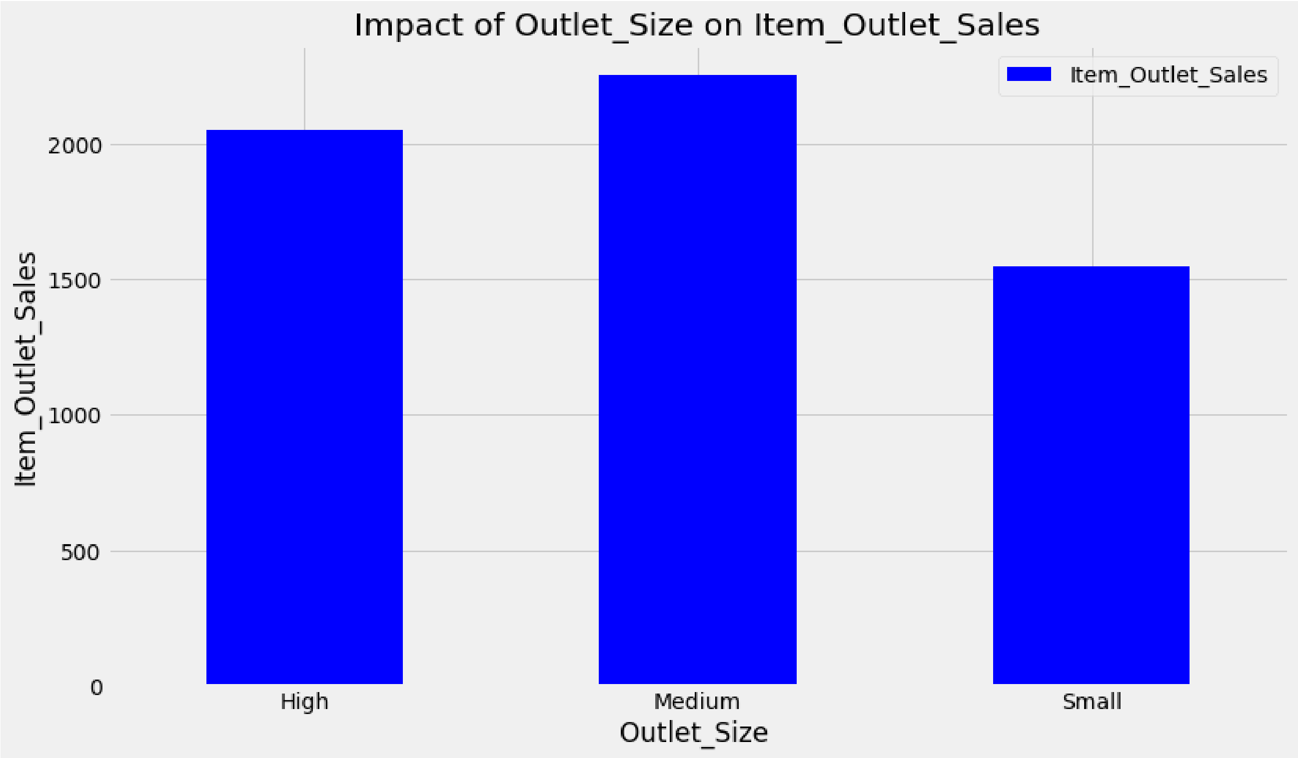
Outlet_Identifier	OUT010	OUT013	OUT017	OUT018	OUT019	OUT027
Outlet_Type	Grocery Store	Supermarket Type1	Supermarket Type1	Supermarket Type2	Grocery Store	Supermarket Type3

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

Outlet_Size	High	Medium	Small
Outlet_Type	Supermarket Type1	Supermarket Type3	Supermarket Type1

1.2.2.4. Impact of Outlet_Size on Item_Outlet_Sales

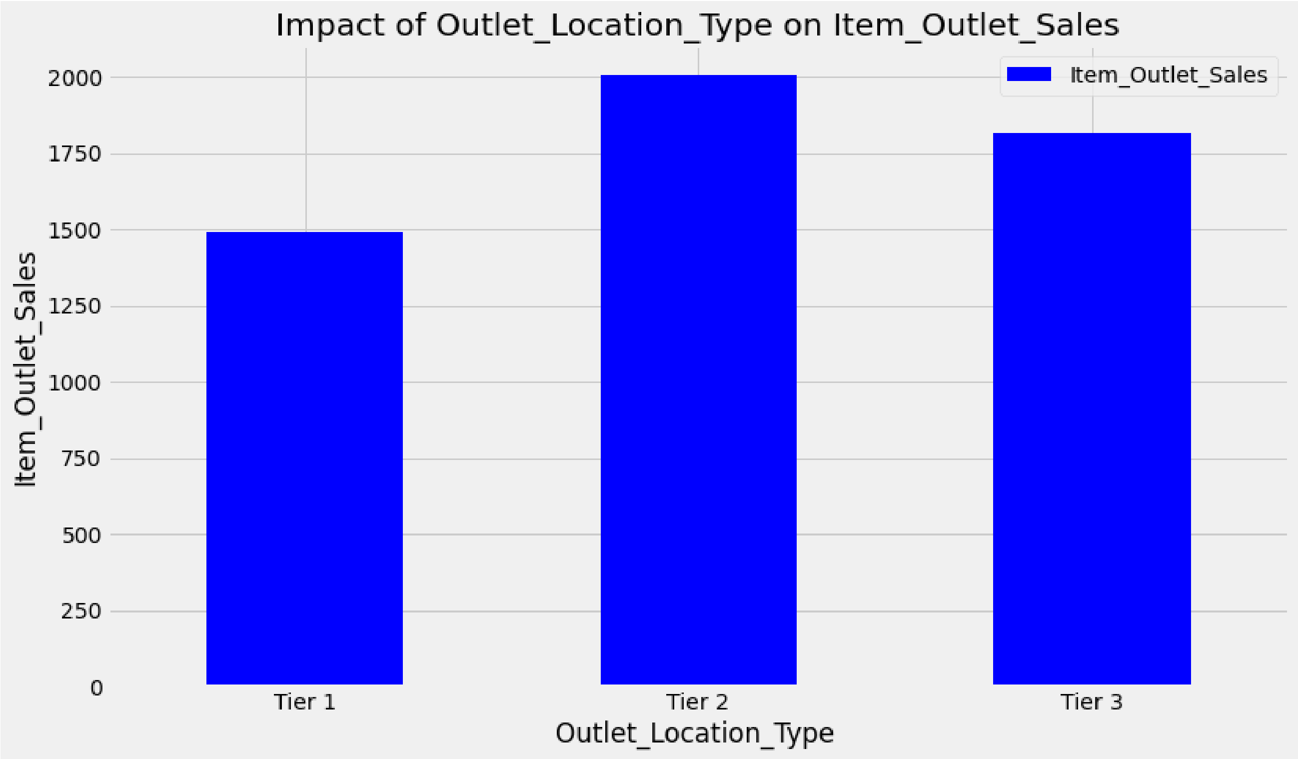
```
Outlet_Size_pivot = \
train.pivot_table(index='Outlet_Size', values="Item_Outlet_Sales", aggfunc=np.median)

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()
```



1.2.2.5. Impact of Outlet_Location_Type on Item_Outlet_Sales

```
Outlet_Location_Type_pivot = \
train.pivot_table(index='Outlet_Location_Type', values="Item_Outlet_Sales", aggfunc=np.med

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()
```

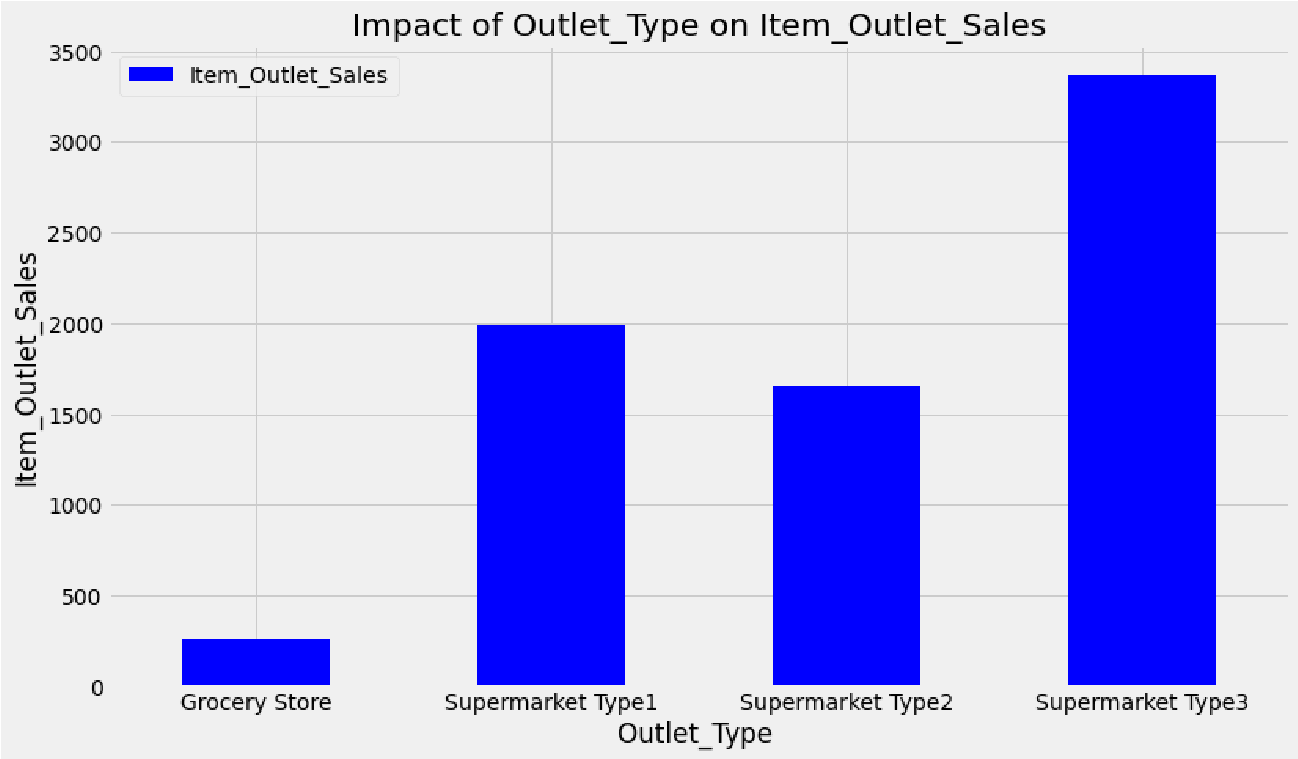



```
train.pivot_table(values='Outlet_Location_Type', columns='Outlet_Type',aggfunc=lambda x:x.
```

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

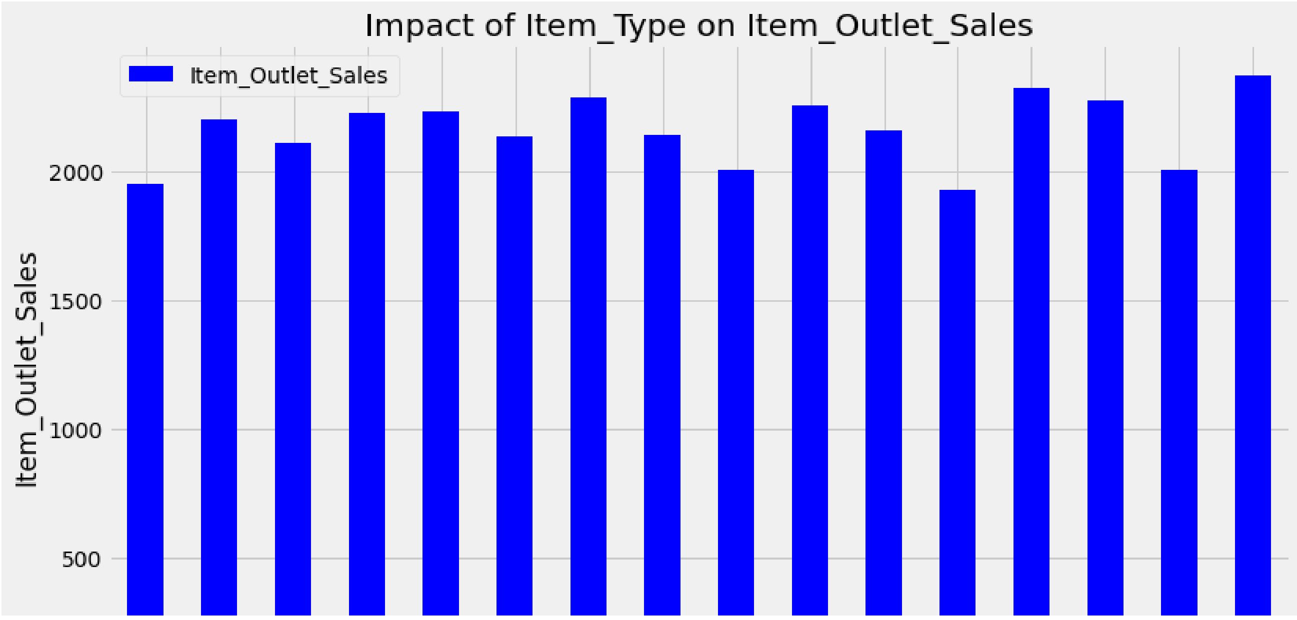
1.2.2.6. Impact of Outlet_Type on Item_Outlet_Sales

```
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()
```



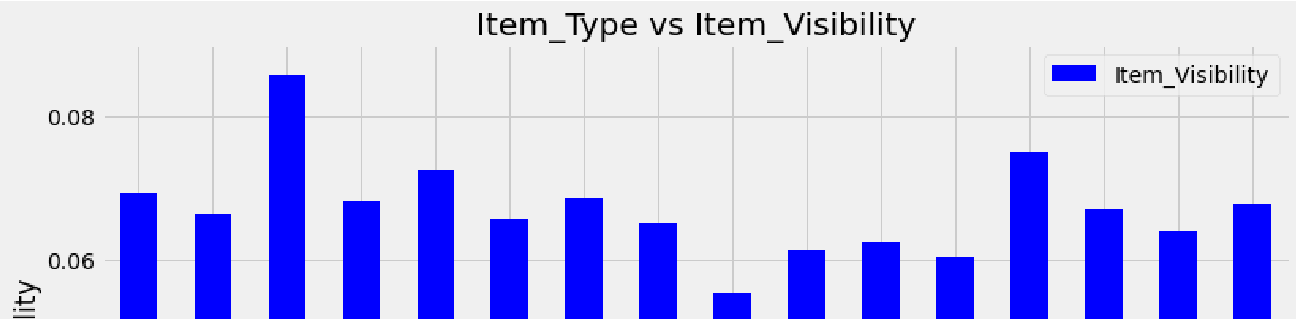
1.2.2.7. Impact of Item_Type on Item_Outlet_Sales

```
pivoTable = \
train.pivot_table(index='Item_Type', values="Item_Outlet_Sales", aggfunc=np.mean)

pivoTable.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Item_Type ")
plt.ylabel("Item_Outlet_Sales")
plt.title("Impact of Item_Type on Item_Outlet_Sales")
plt.xticks(rotation=90)
plt.show()
```



1.2.2.8. Impact of Item_Type vs Item_Visibility

```
pivoTable = \
train.pivot_table(index='Item_Type', values="Item_Visibility", aggfunc=np.mean)

pivoTable.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Item_Type ")
plt.ylabel("Item_Visibility")
plt.title("Item_Type vs Item_Visibility")
plt.xticks(rotation=90)
plt.show()
```



2. Data Pre-Processing

2.1. Looking for missing values


```
# Join Train and Test Dataset
train['source']='train'
test['source']='test'

data = pd.concat([train,test], ignore_index = True)
data.to_csv("data/data.csv",index=False)
print(train.shape, test.shape, data.shape)
```

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

2.2. Imputing Missing Values

```
 #aggfunc is mean by default! Ignores NA 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]
```

```
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).
print ('Final #missing: %d'%sum(data['Item_Weight'].isnull()))
```

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

2.3. Imputing Outlet_size with the mode

```
#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=la
outlet_size_mode
```

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

```
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
```

3. Feature Engineering

3.1. Should we combine Outlet_Type?

```
#Creates pivot table with Outlet_Type and the mean of Item_Outlet_Sales. Agg function is b
data.pivot_table(values='Item_Outlet_Sales', columns='Outlet_Type')
```

Outlet_Type	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3

3.2. Item_Visibility minimum value 0

```
#Get all Item_Visibility mean values for respective Item_Identifier
visibility_item_avg = data.pivot_table(values='Item_Visibility',index='Item_Identifier')


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'] = data[['Item_Visibility','Item_Identifier']].apply(impute_visibil
print ('Final #zeros: %d'%sum(data['Item_Visibility'] == 0))
```

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

3.3. Determine the years of operation of a store

```
#Years:
data['Outlet_Years'] = 2013 - data['Outlet_Establishment_Year']
data['Outlet_Years'].describe()
```

```
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
```

3.4. Create a broad category of Type of Item

```
#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()
```

```
Food              10201
```

```
Non-Consumable     2686
Drinks             1317
Name: Item_Type_Combined, dtype: int64
```

3.5. Modify categories of Item_Fat_Content

```
#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
```

```
#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()
```

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

4. Feature Transformations

4.1. Creating variable Item_Visibility_MeanRatio

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

```
count    14204.000000
mean         1.061884
std          0.235907
```

```
min          0.844563
25%          0.925131
50%          0.999070
75%          1.042007
max          3.010094
Name: Item_Visibility_MeanRatio, dtype: float64
```

4.2. Numerical and Categorical Variables – Dummy variables

```
#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','O
le = LabelEncoder()
for i in var_mod:
    data[i] = le.fit_transform(data[i])


#Dummy Variables:
data = pd.get_dummies(data, columns=['Item_Fat_Content','Outlet_Location_Type','Outlet_Siz
                              'Item_Type_Combined','Outlet'])

data.dtypes
```

```
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
```

4.3. Exporting Data

```
#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)
```

5. Model, predict and solve the problem

```
train_df = pd.read_csv('data/train_modified.csv')
test_df = pd.read_csv('data/test_modified.csv')
```

```
#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='
    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, dtra
    print("CV Score : Mean - %.4g | Std - %.4g | Min - %.4g | Max - %.4g" % (np.mean(cv_sc

    #Print r2 score:
    print("r2 score : %.4g" % metrics.r2_score((dtrain[target]).values, dtrain_predictions

    #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

```
from sklearn.linear_model import LinearRegression
LR = LinearRegression(normalize=True)

predictors = train_df.columns.drop(['Item_Outlet_Sales','Item_Identifier','Outlet_Identifi
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')
```

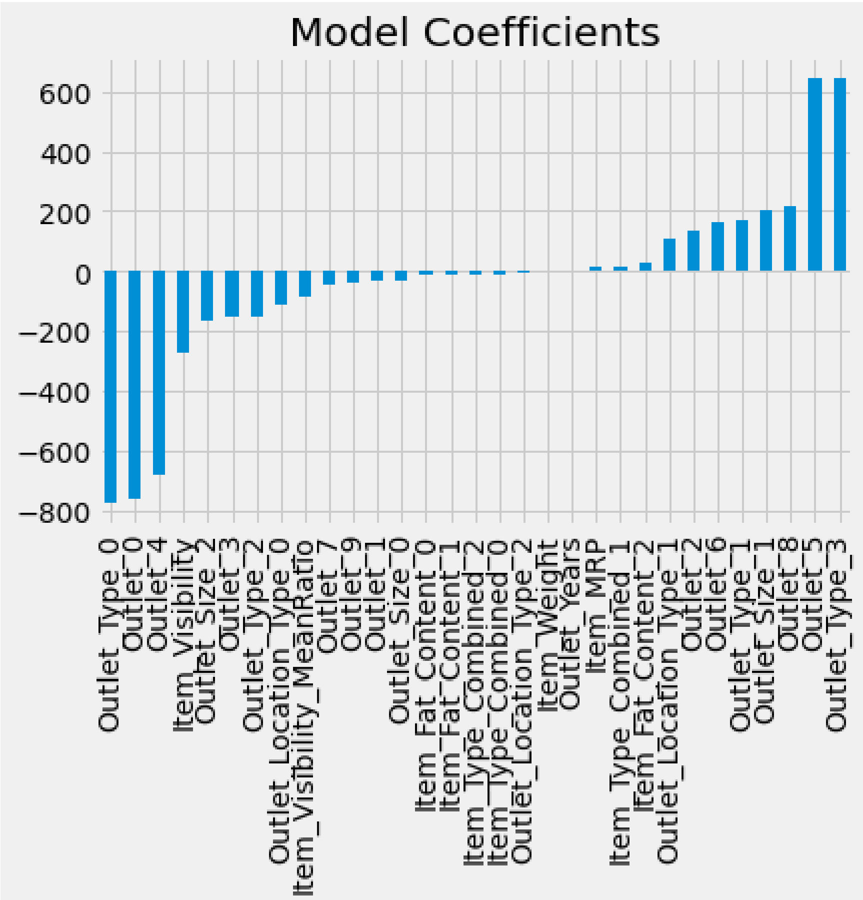
```
Model Report
```

Ridge Regression Model

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c15df8c50>
```

```
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')
```

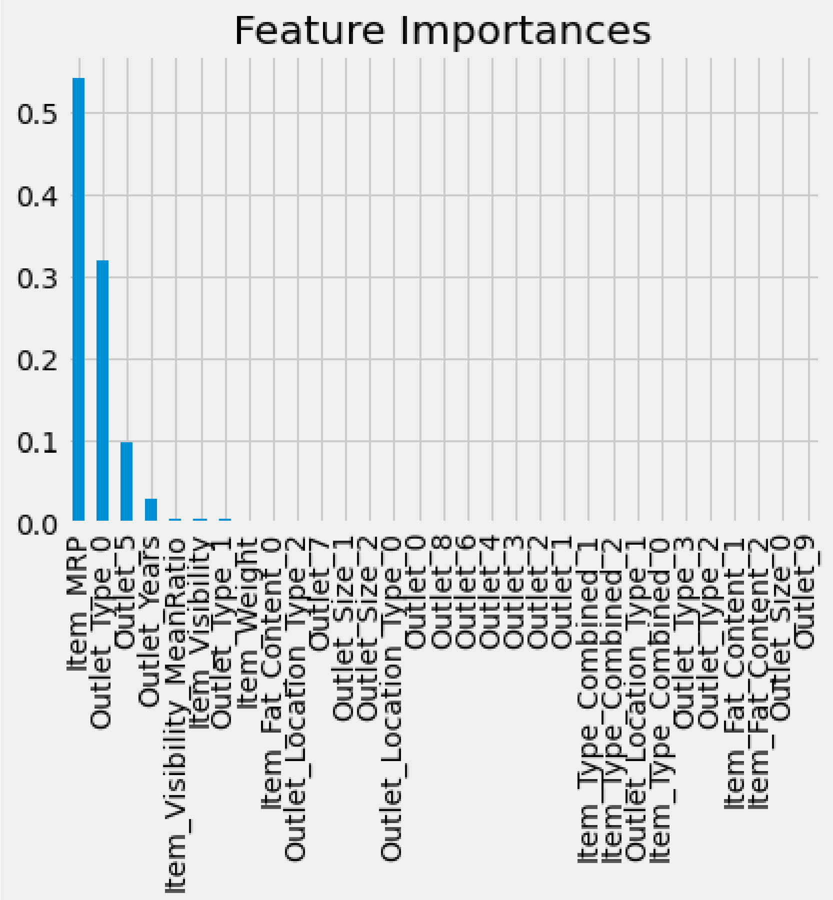
```
Model Report
RMSE : 1129
CV Score : Mean - 1130 | Std - 44.6 | Min - 1076 | Max - 1217
r2 score : 0.5625
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c15913f10>
```



Decision Tree Model

```
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 - 1091 | Std - 45.42 | Min - 1003 | Max - 1186
r2 score : 0.6158
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c15cb4750>
```



Random Forest Model

```
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 : 1069
CV Score : Mean - 1097 | Std - 43.41 | Min - 1028 | Max - 1180
r2 score : 0.6077
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c15b15150>
```

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)
```

```
[08:42:53] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now
XGBRegressor(learning_rate=0.05, n_estimators=1000)
```

```
#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_
print("r2 score : %.4g" % metrics.r2_score((train_df[target]).values, train_df_predictions
```

```
Mean Absolute Error : 220.26657130725278
RMSE : 1052
r2 score : 0.6197
```

```
IDcol.append(target)
submission = pd.DataFrame({ x: test_df[x] for x in IDcol})
submission.to_csv("XGboost.csv", index=False)
```