```
import pandas as pd
import numpy as np

#Read files:
traindataset = pd.read_csv("Train.csv")
testdataset = pd.read_csv("Test.csv")
print (traindataset.head(10))
testdataset.head(10)
```

```
  Item_Identifier  Item_Weight  ...        Outlet_Type  Item_Outlet_Sales
0           FDA15        9.300  ...  Supermarket Type1          3735.1380
1           DRC01        5.920  ...  Supermarket Type2           443.4228
2           FDN15       17.500  ...  Supermarket Type1          2097.2700
3           FDX07       19.200  ...      Grocery Store           732.3800
4           NCD19        8.930  ...  Supermarket Type1           994.7052
5           FDP36       10.395  ...  Supermarket Type2           556.6088
6           FDO10       13.650  ...  Supermarket Type1           343.5528
7           FDP10          NaN  ...  Supermarket Type3          4022.7636
8           FDH17       16.200  ...  Supermarket Type1          1076.5986
9           FDU28       19.200  ...  Supermarket Type1          4710.5350

[10 rows x 12 columns]
```

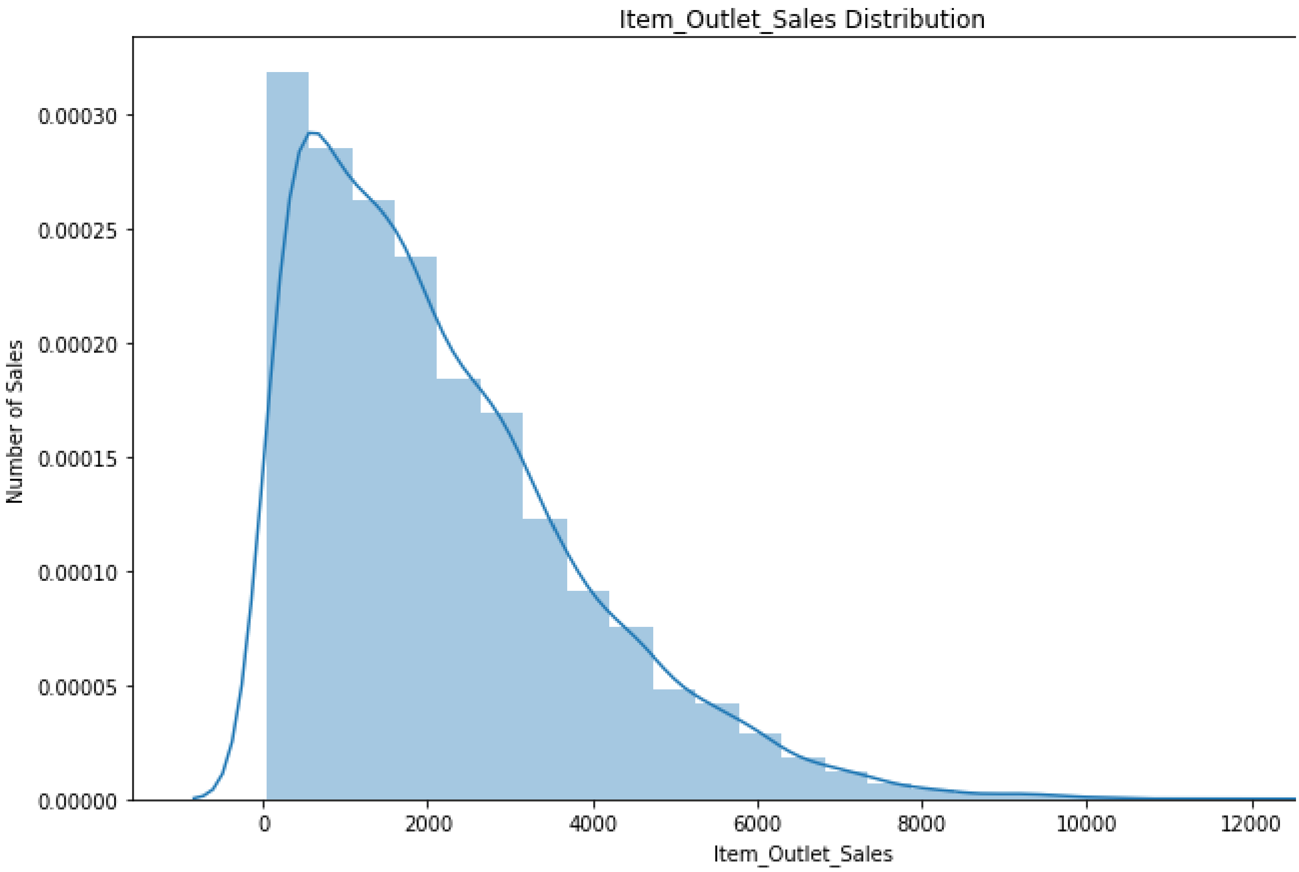
	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8
1	FDW14	8.300	reg	0.038428	Dairy	87.3
2	NCN55	14.600	Low Fat	0.099575	Others	241.7
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0
4	FDY38	NaN	Regular	0.118599	Dairy	234.2
5	FDH56	9.800	Regular	0.063817	Fruits and Vegetables	117.1
6	FDL48	19.350	Regular	0.082602	Baking Goods	50.1
7	FDC48	NaN	Low Fat	0.015782	Baking Goods	81.0
8	FDN33	6.305	Regular	0.123365	Snack Foods	95.7
9	FDA36	5.985	Low Fat	0.005698	Baking Goods	186.8

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

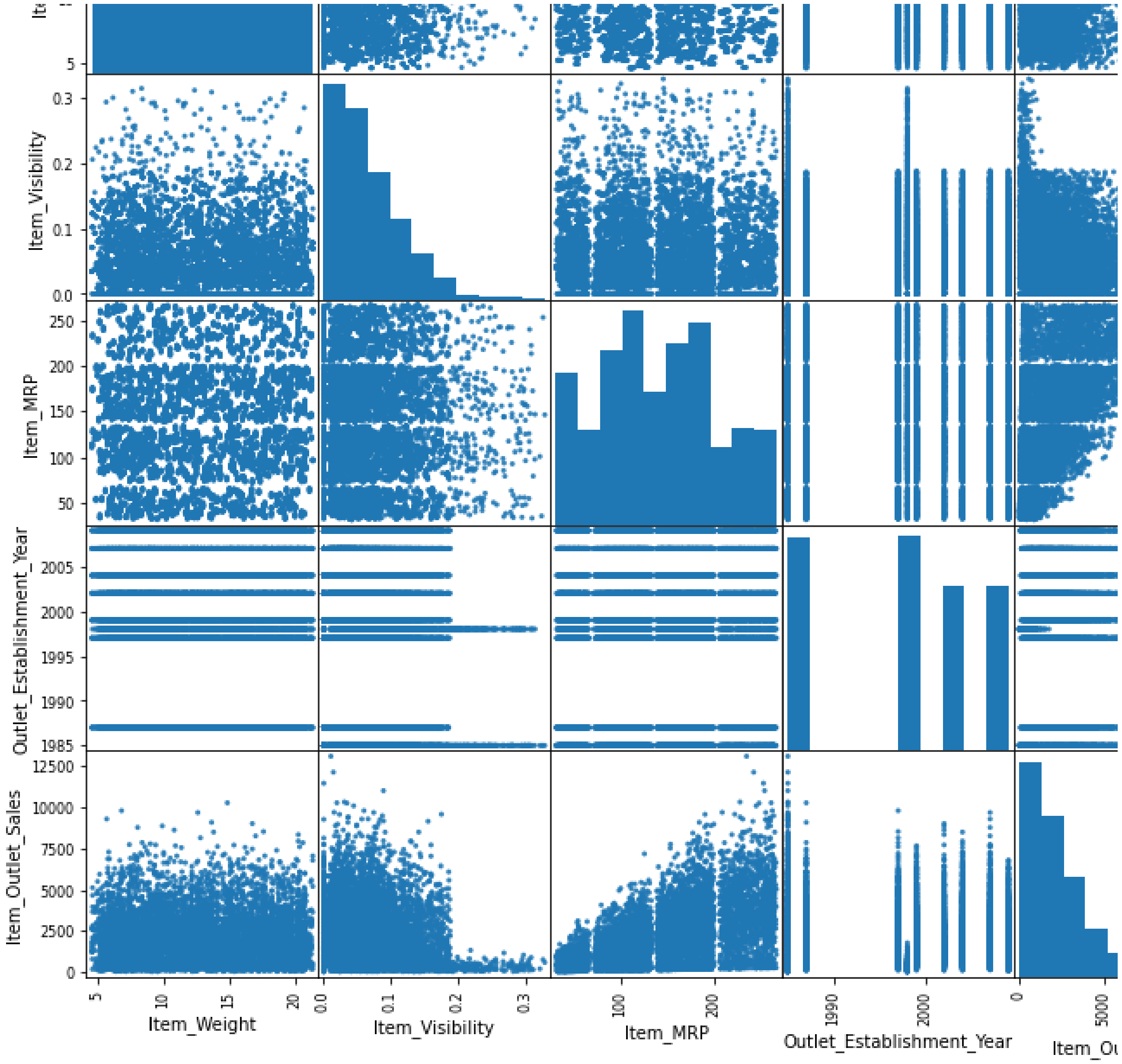
```
(8523, 13) (5681, 12) (14204, 13)
```

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12,7))
sns.distplot(traindataset.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")
```

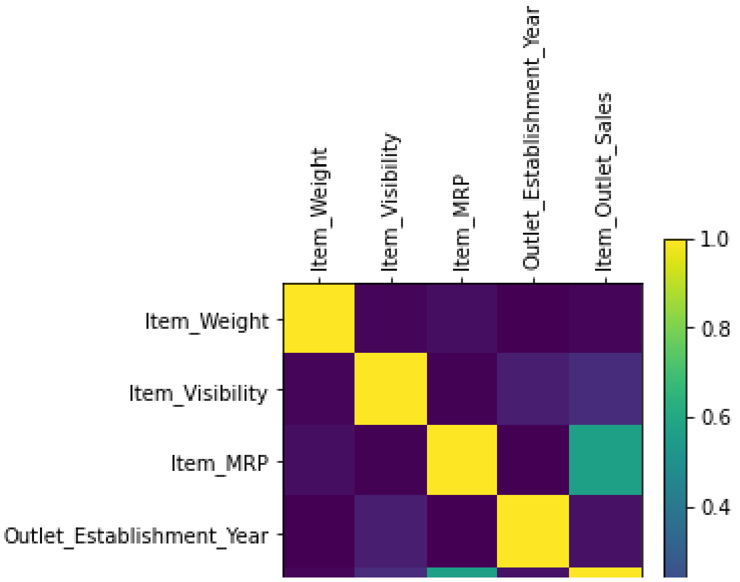
```
Text(0.5, 1.0, 'Item_Outlet_Sales Distribution')
```



```
from pandas.plotting import scatter_matrix
scatter_matrix(data, alpha=0.9, figsize=(12,12), diagonal='hist')
plt.show()
pair_corr_coeff = data.corr()
print(pair_corr_coeff)
#pair_corr_coeff.abs().style.background_gradient()

plt.matshow(np.abs(pair_corr_coeff))
plt.colorbar()
plt.xticks(range(len(pair_corr_coeff.columns)), pair_corr_coeff.columns, rotation='vertica
plt.yticks(range(len(pair_corr_coeff.columns)), pair_corr_coeff.columns)
plt.show()
```

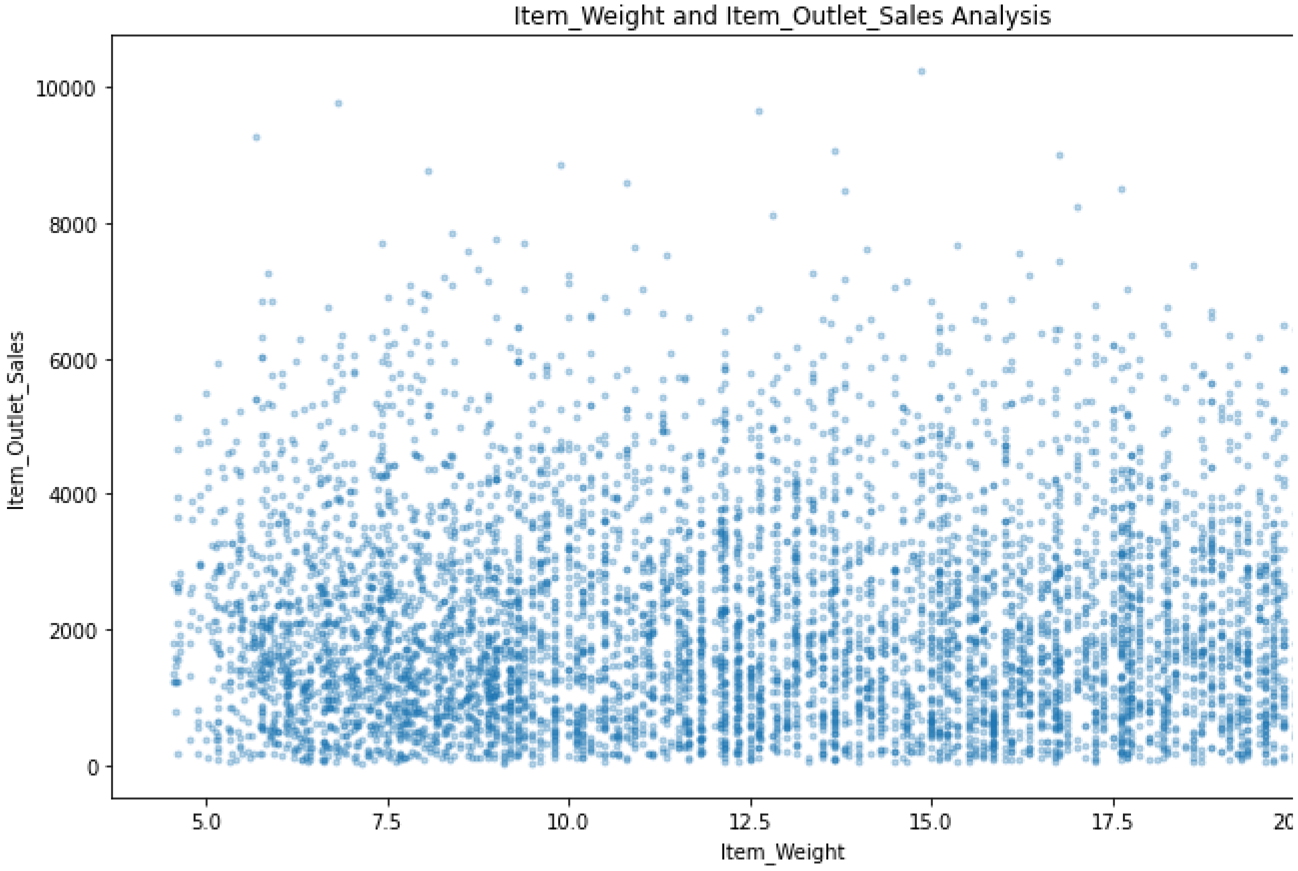


```
                           Item_Weight  ...  Item_Outlet_Sales
Item_Weight                   1.000000  ...           0.014123
Item_Visibility              -0.015901  ...          -0.128625
Item_MRP                      0.036236  ...           0.567574
Outlet_Establishment_Year     0.000645  ...          -0.049135
Item_Outlet_Sales             0.014123  ...           1.000000

[5 rows x 5 columns]
```



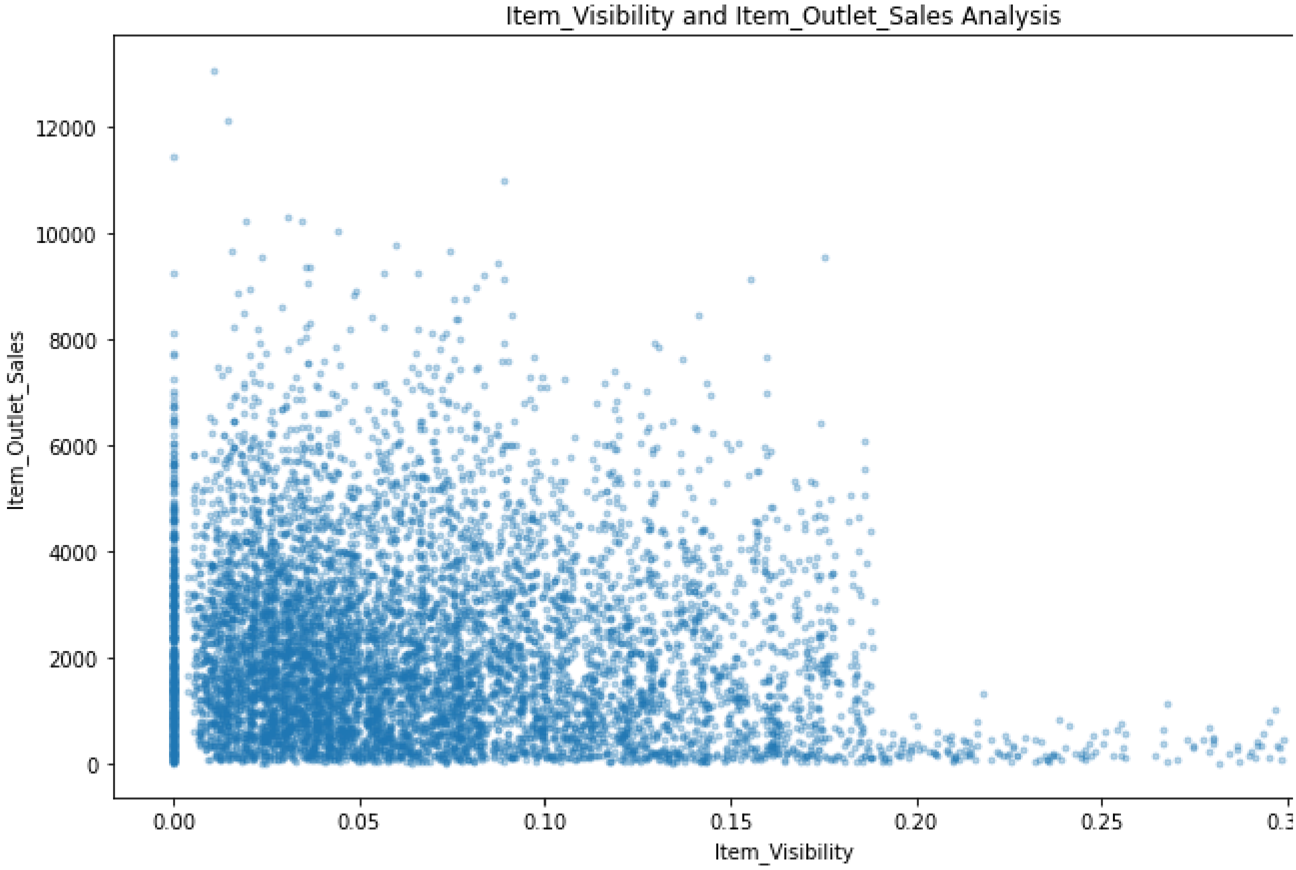
```
plt.figure(figsize=(12,7))
plt.xlabel("Item_Weight")
plt.ylabel("Item_Outlet_Sales")
plt.title("Item_Weight and Item_Outlet_Sales Analysis")
plt.plot(traindataset.Item_Weight, traindataset["Item_Outlet_Sales"],'.', alpha = 0.3)
```

```
[<matplotlib.lines.Line2D at 0x7f2c82330be0>]
```

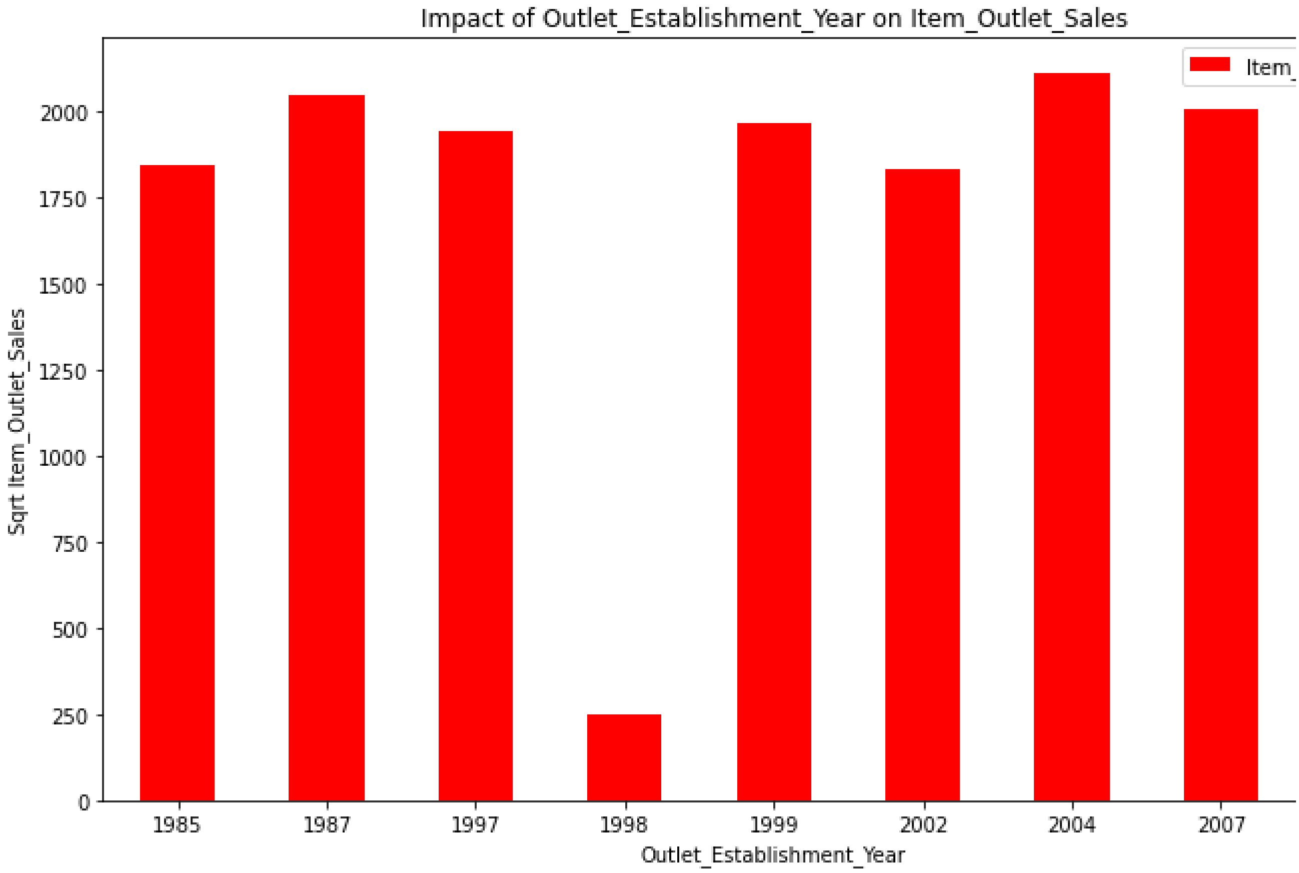


```
plt.figure(figsize=(12,7))
plt.xlabel("Item_Visibility")
plt.ylabel("Item_Outlet_Sales")
plt.title("Item_Visibility and Item_Outlet_Sales Analysis")
plt.plot(traindataset.Item_Visibility, traindataset["Item_Outlet_Sales"],".", alpha = 0.3)
```

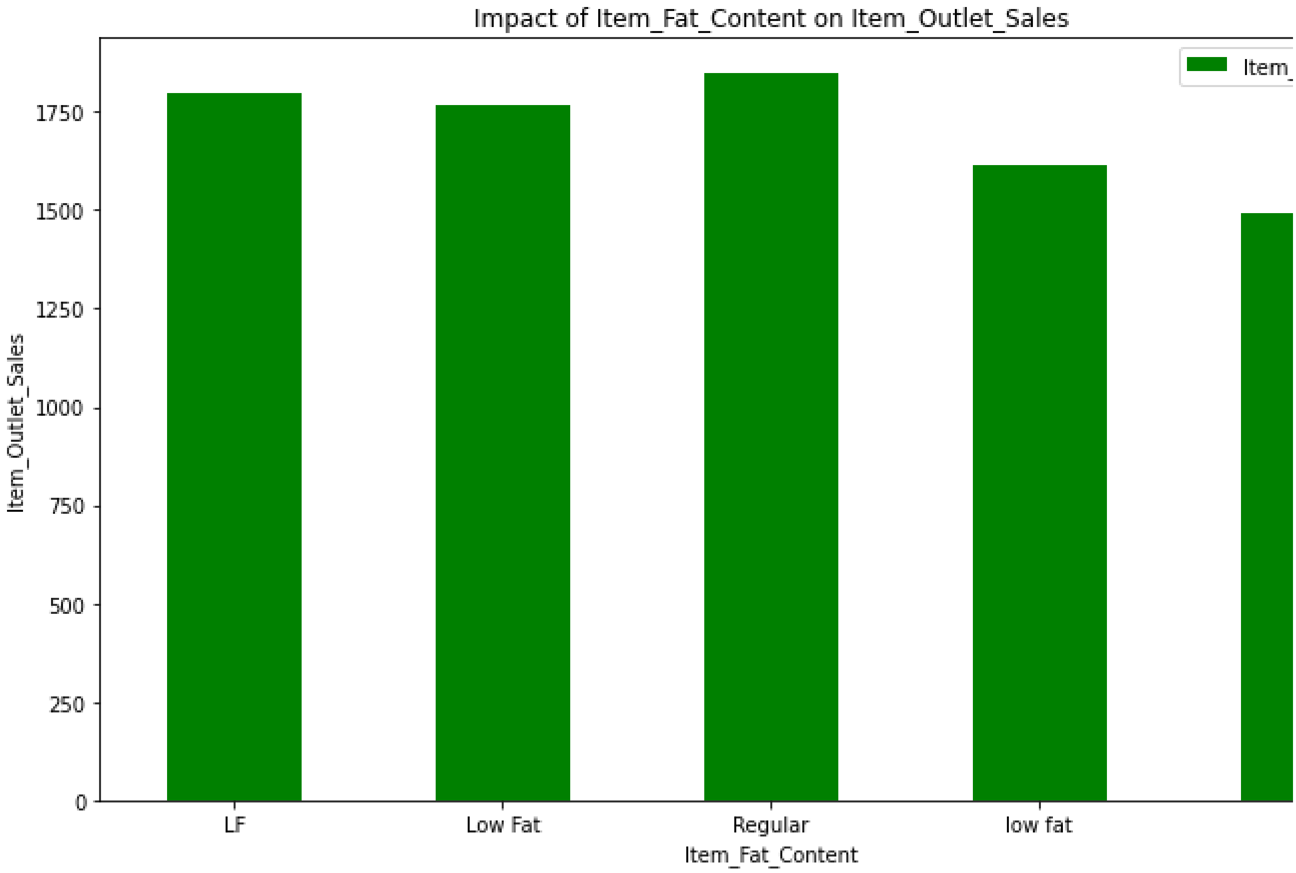
```
[<matplotlib.lines.Line2D at 0x7f2c824a6358>]
```



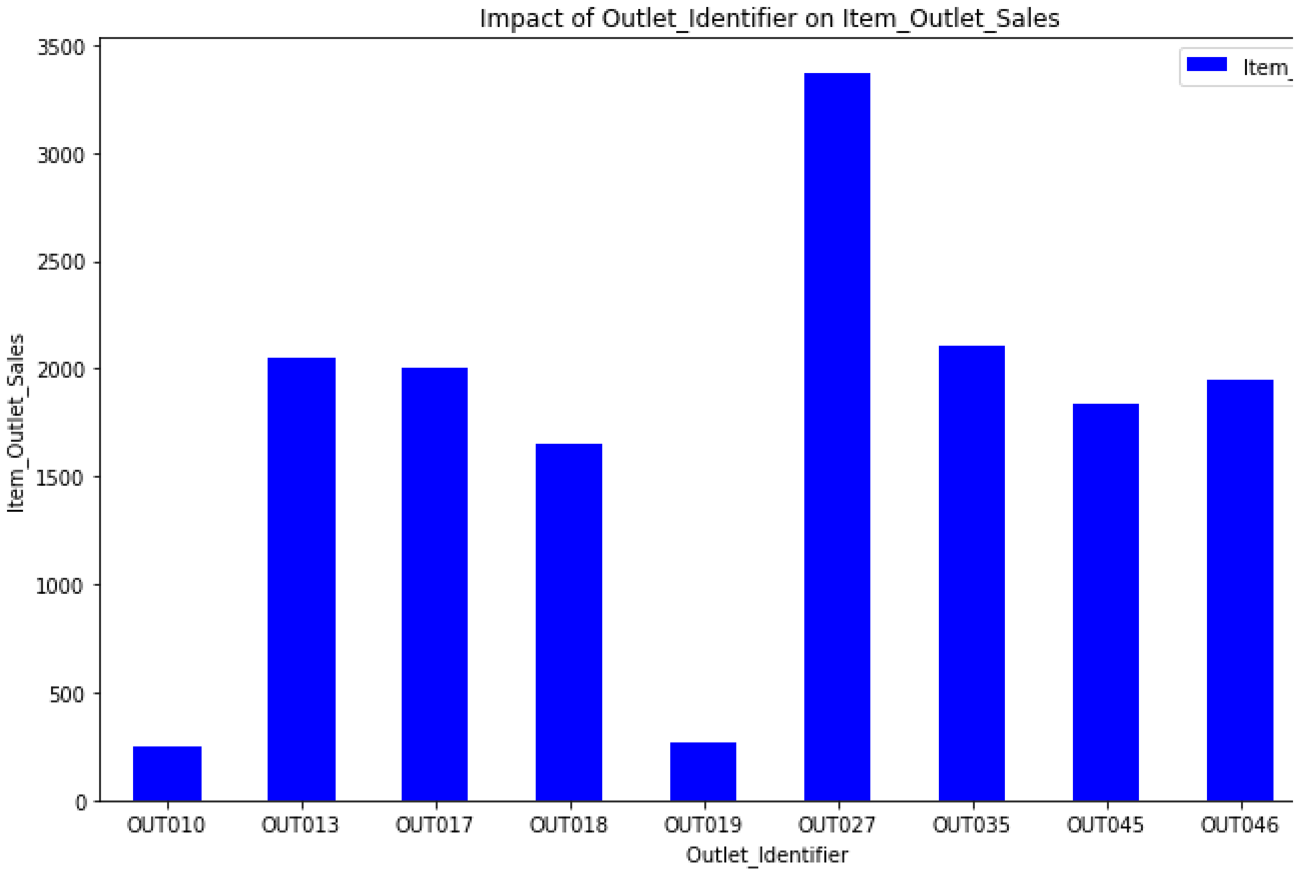
```
Outlet_Establishment_Year_pivot = traindataset.pivot_table(index='Outlet_Establishment_Yea
Outlet_Establishment_Year_pivot.plot(kind='bar', color='red',figsize=(12,7))
plt.xlabel("Outlet_Establishment_Year")
plt.ylabel("Sqrt Item_Outlet_Sales")
plt.title("Impact of Outlet_Establishment_Year on Item_Outlet_Sales")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



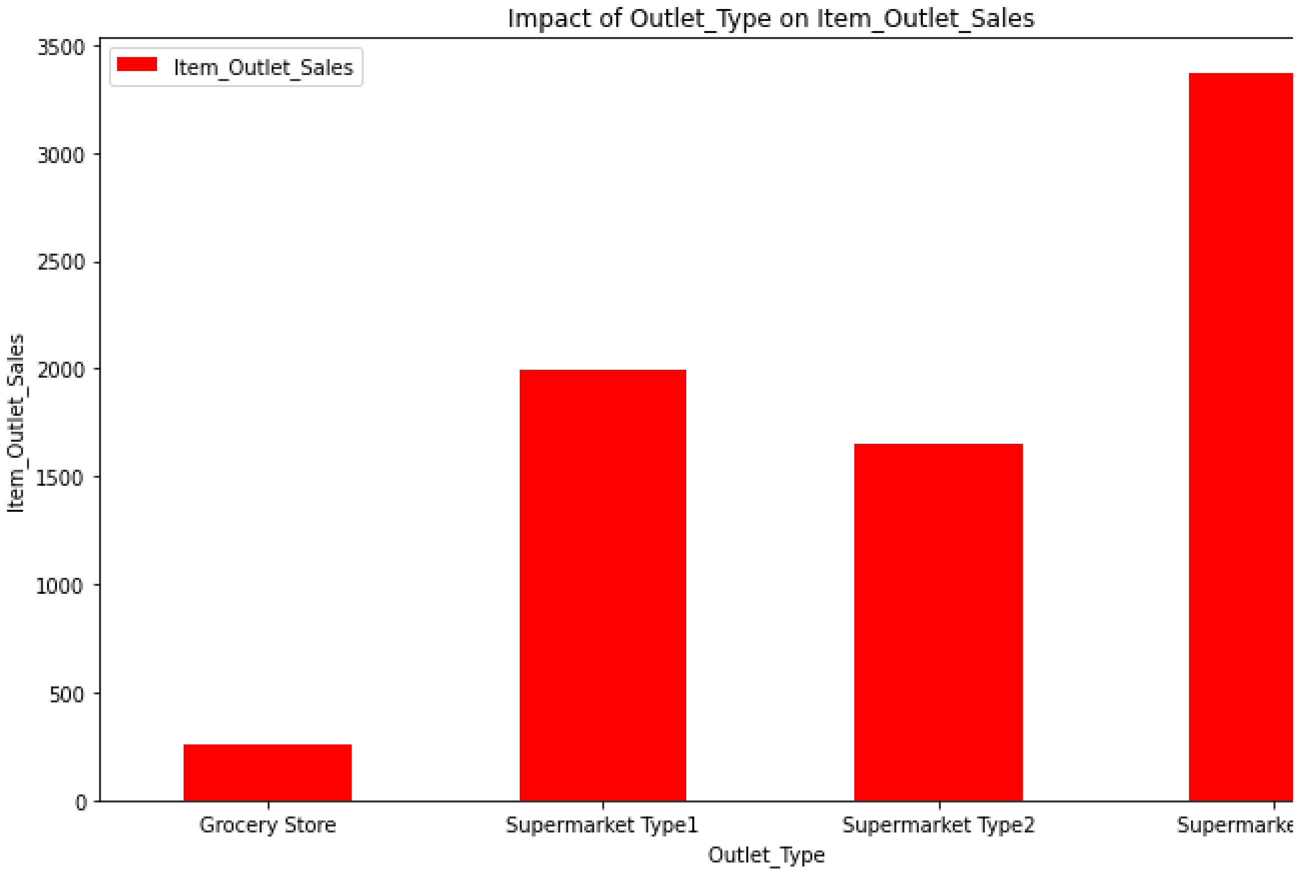
```
Item_Fat_Content_pivot =traindataset.pivot_table(index='Item_Fat_Content', values="Item_Ou
Item_Fat_Content_pivot.plot(kind='bar', color='green',figsize=(12,7))
plt.xlabel("Item_Fat_Content")
plt.ylabel("Item_Outlet_Sales")
plt.title("Impact of Item_Fat_Content on Item_Outlet_Sales")
plt.xticks(rotation=0)
plt.show()
```



```
Outlet_Identifier_pivot = traindataset.pivot_table(index="Outlet_Identifier", values="Item
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()
```

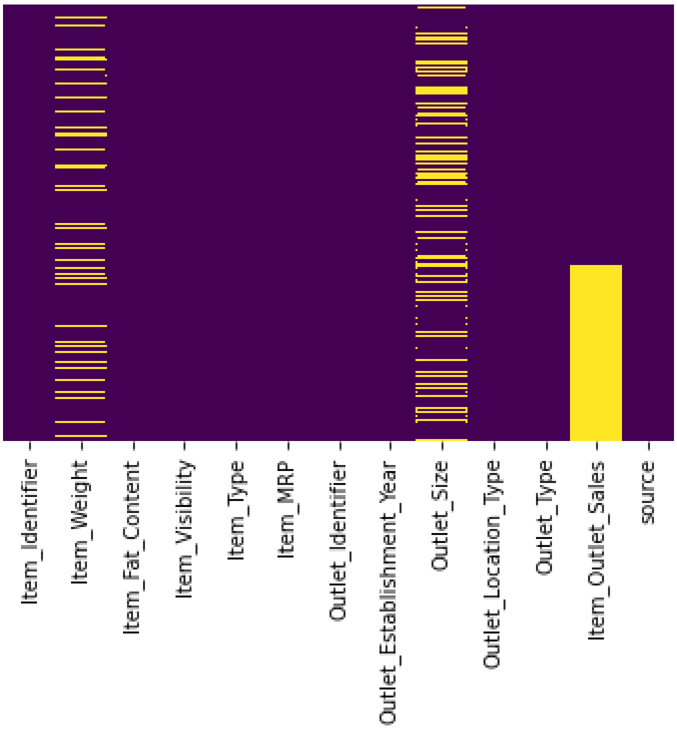


```
Outlet_Type_pivot = traindataset.pivot_table(index='Outlet_Type', values="Item_Outlet_Sale
Outlet_Type_pivot.plot(kind='bar', color='red',figsize=(12,7))
plt.xlabel("Outlet_Type ")
plt.ylabel("Item_Outlet_Sales")
plt.title("Impact of Outlet_Type on Item_Outlet_Sales")
plt.xticks(rotation=0)
plt.show()
```



```
print(data.apply(lambda x: sum(x.isnull())))
sns.heatmap(data.isnull(),cbar=False,cmap='viridis',yticklabels=False)
```

```
Item_Identifier                 0
Item_Weight                  2439
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  4016
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales            5681
source                          0
dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7f2c850d65c0>
```



```
data.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_O
count	11765.000000	14204.000000	14204.000000	14204.000000	
mean	12.792854	0.065953	141.004977	1997.830681	
std	4.652502	0.051459	62.086938	8.371664	
min	4.555000	0.000000	31.290000	1985.000000	

```
data.apply(lambda x: len(x.unique()))
```

```
Item_Identifier               1559
Item_Weight                    416
Item_Fat_Content                 5
Item_Visibility              13006
Item_Type                       16
Item_MRP                      8052
Outlet_Identifier               10
Outlet_Establishment_Year        9
Outlet_Size                      4
Outlet_Location_Type             3
Outlet_Type                      4
Item_Outlet_Sales             3494
source                           2
dtype: int64
```

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

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

Frequency of Categories for varible Item_Type
Fruits and Vegetables    2013
Snack Foods              1989
Household                1548
Frozen Foods             1426
Dairy                    1136
Baking Goods             1086
Canned                   1084
Health and Hygiene        858
Meat                      736
Soft Drinks               726
Breads                    416
Hard Drinks               362
Others                    280
Starchy Foods             269
Breakfast                 186
Seafood                    89
Name: Item_Type, dtype: int64

Frequency of Categories for varible Outlet_Size
Medium    4655
Small     3980
High      1553
Name: Outlet_Size, dtype: int64

Frequency of Categories for varible Outlet_Location_Type
Tier 3    5583
Tier 2    4641
Tier 1    3980
Name: Outlet_Location_Type, dtype: int64

Frequency of Categories for varible Outlet_Type
Supermarket Type1    9294
Grocery Store        1805
Supermarket Type3    1559
Supermarket Type2    1546
Name: Outlet_Type, dtype: int64
```

```
item_avg_weight = data.pivot_table(values='Item_Weight', index='Item_Identifier')
item_avg_weight = data.pivot_table(values='Item_Weight', index='Item_Identifier')
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
data['Item_Weight'] = data[['Item_Weight','Item_Identifier']].apply(impute_weight,axis=1).
print ('Final missing: %d'%sum(data['Item_Weight'].isnull()))
```

```
Final missing: 0
```

```
#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=(l
print ('Mode for each Outlet_Type:')
print (outlet_size_mode)

#Get a boolean variable specifying missing Item_Weight values
miss_bool = data['Outlet_Size'].isnull()

#Impute data and check #missing values before and after imputation to confirm

data.loc[miss_bool,'Outlet_Size'] = data.loc[miss_bool,'Outlet_Type'].apply(lambda x: outl
print("final values that are missing are")
print (sum(data['Outlet_Size'].isnull()))
```

```
Mode for each Outlet_Type:
Outlet_Type Grocery Store Supermarket Type1 Supermarket Type2 Supermarket Type3
Outlet_Size         Small             Small            Medium            Medium
final values that are missing are
0
```

```
sns.heatmap(data.isnull(),cbar=False,cmap='viridis',yticklabels=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2c85147320>
```

```
data.pivot_table(values='Item_Outlet_Sales',index='Outlet_Type')
```

	Item_Outlet_Sales
Outlet_Type	
Grocery Store	339.828500
Supermarket Type1	2316.181148
Supermarket Type2	1995.498739
Supermarket Type3	3694.038558

```
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
```

```
data['Item_Visibility_MeanRatio'] = data.apply(lambda x: x['Item_Visibility']/visibility_i
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
```

```
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
```

```
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
```

```
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
```

```
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
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
#New variable for outlet
data['Outlet'] = le.fit_transform(data['Outlet_Identifier'])
```

```
data['Outlet'] = le.fit_transform(data['Outlet_Identifier'])
var_mod = ['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Item_Type_Combined','(
le = LabelEncoder()
for i in var_mod:
    data[i] = le.fit_transform(data[i])
```

```
s(data, columns=['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Outlet_Type','It
```

```
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
Item_Visibility_MeanRatio     float64
Outlet_Years                    int64
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
```

```
data[['Item_Fat_Content_0','Item_Fat_Content_1','Item_Fat_Content_2']].head(10)
```

	Item_Fat_Content_0	Item_Fat_Content_1	Item_Fat_Content_2
0	1	0	0
1	0	0	1
2	1	0	0
3	0	0	1
4	0	1	0
5	0	0	1
6	0	0	1
7	1	0	0
8	0	0	1
9	0	0	1

```
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("train_modified.csv",index=False)
test.to_csv("test_modified.csv",index=False)
```

```
/usr/local/lib/python3.6/dist-packages/pandas/core/frame.py:3997: SettingWithCopyWarr
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  errors=errors,
```

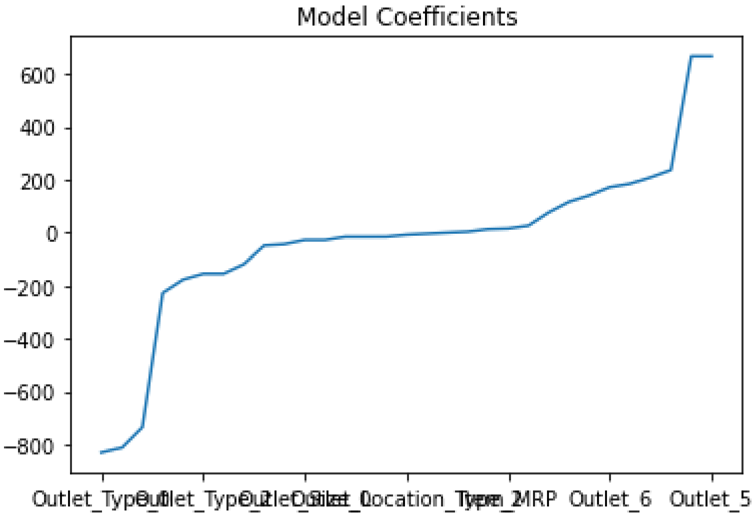
```
mean_sales = train['Item_Outlet_Sales'].mean()

#Define a dataframe with IDs for submission:
```

```
#Define a dataframe with IDs for submission:
base1 = test[['Item_Identifier','Outlet_Identifier']]
base1['Item_Outlet_Sales'] = mean_sales

#Export submission file
base1.to_csv("alg0.csv",index=False)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  """
```

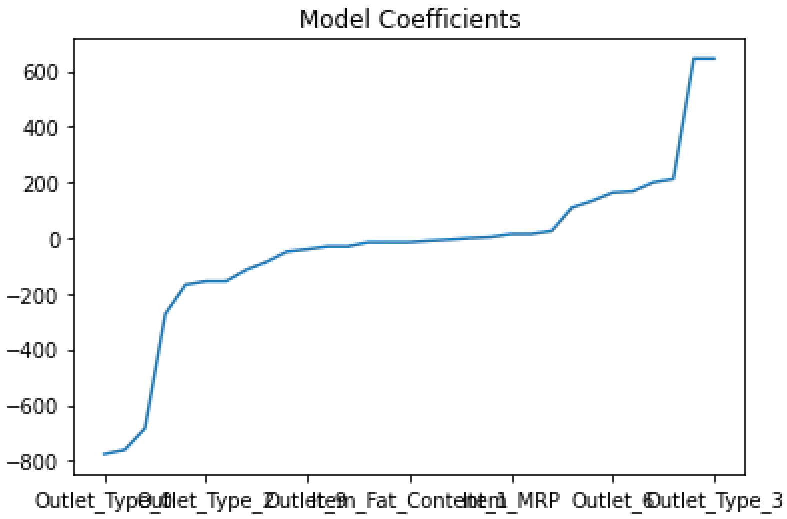
```
target = 'Item_Outlet_Sales'
IDcol = ['Item_Identifier','Outlet_Identifier']
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.linear_model import LinearRegression, Ridge, Lasso
predictors = [x for x in train.columns if x not in [target]+IDcol]
# print predictors
alg1 = LinearRegression(normalize=True)

    #Fit the algorithm on the data
alg1.fit(train[predictors], train[target])

    #Predict training set:
train_predictions = alg1.predict(train[predictors])

    #Perform cross-validation:
cv_score = cross_val_score(alg1, train[predictors], train[target], cv=20)
cv_score = np.sqrt(np.abs(cv_score))

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

    #Predict on testing data:
test[target] = alg1.predict(test[predictors])

    #Export submission file:
IDcol.append(target)
submission = pd.DataFrame({ x: test[x] for x in IDcol})
submission.to_csv('alg1.csv', index=False)
coef1 = pd.Series(alg1.coef_, predictors).sort_values()
coef1.plot(title='Model Coefficients')
print("Accuracy is :")
print(alg1.score(train[predictors],train[target])*100)
```

```
Model Report
RMSE : 1127
CV Score : Mean - 0.7475 | Std - 0.0194 | Min - 0.6984 | Max - 0.7847
Accuracy is :
56.350509261468716
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:27: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```



```
target = 'Item_Outlet_Sales'
IDcol = ['Item_Identifier','Outlet_Identifier']
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.linear_model import LinearRegression, Ridge, Lasso
predictors = [x for x in train.columns if x not in [target]+IDcol]
alg2 = Ridge(alpha=0.05,normalize=True)

    #Fit the algorithm on the data
alg2.fit(train[predictors], train[target])

    #Predict training set:
train_predictions = alg2.predict(train[predictors])

    #Perform cross-validation:
cv_score = cross_val_score(alg2, train[predictors], train[target], cv=20)
cv_score = cross_val_score(alg2, train[predictors], train[target], cv=20)
cv_score = np.sqrt(np.abs(cv_score))

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

    #Predict on testing data:
test[target] = alg2.predict(test[predictors])

    #Export submission file:
IDcol.append(target)
submission = pd.DataFrame({ x: test[x] for x in IDcol})
submission.to_csv('alg2.csv', index=False)
coef2 = pd.Series(alg2.coef_, predictors).sort_values()
coef2.plot( title='Model Coefficients')
print("Accuracy is :")
print(alg2.score(train[predictors],train[target])*100)
```

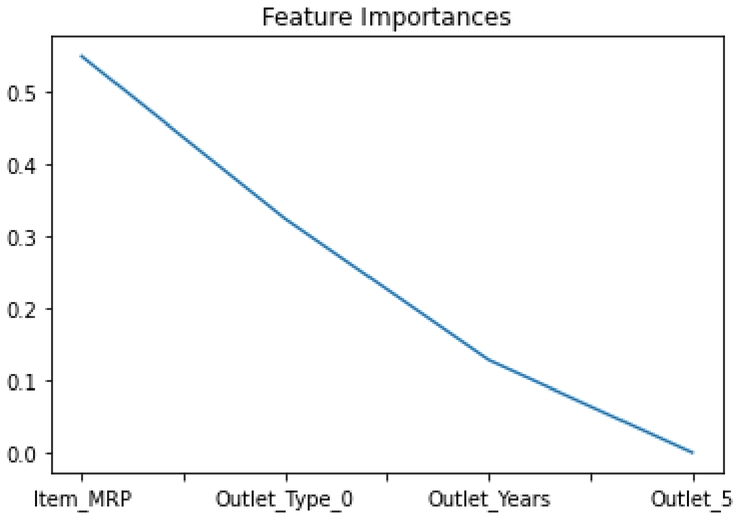
```
Model Report
RMSE : 1129
CV Score : Mean - 0.7471 | Std - 0.01819 | Min - 0.7032 | Max - 0.7828
Accuracy is :
56.25404650408207
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:26: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```



```
target = 'Item_Outlet_Sales'
IDcol = ['Item_Identifier','Outlet_Identifier']
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor

predictors = ['Item_MRP','Outlet_Type_0','Outlet_5','Outlet_Years']
alg3 = DecisionTreeRegressor(max_depth=8, min_samples_leaf=150)
```

```
alg3 = DecisionTreeRegressor(max_depth=8, min_samples_leaf=150)

    #Fit the algorithm on the data
alg3.fit(train[predictors], train[target])

    #Predict training set:
train_predictions = alg3.predict(train[predictors])

    #Perform cross-validation:
cv_score = cross_val_score(alg3, train[predictors], train[target], cv=20)
cv_score = np.sqrt(np.abs(cv_score))

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

    #Predict on testing data:
test[target] = alg3.predict(test[predictors])

    #Export submission file:
IDcol.append(target)
submission = pd.DataFrame({ x: test[x] for x in IDcol})
submission.to_csv('alg3.csv', index=False)
coef3 = pd.Series(alg3.feature_importances_, predictors).sort_values(ascending=False)
coef3.plot(title='Feature Importances')
print("Accuracy is : ")
print(alg3.score(train[predictors],train[target])*100)
```

```
Model Report
RMSE : 1071
CV Score : Mean - 0.7635 | Std - 0.02464 | Min - 0.7032 | Max - 0.8087
Accuracy is :
60.58974644993484
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:28: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```


Feature Importances
Item_MRP
Outlet_Type_0
Outlet_Years
Outlet_5