MVP

Analysis of Big mart sales prediction

the goal of these model is to find out the sales of each product at a particular store and the sales of the different stores of Big Mart.

Steps of work:

1- Importing the libraries: numpy, pandas, seaborn, matplotlib and sklearn
2- Importing the data: I used Big Mart dataset it contains two sets: train (8523,12) and test (5681,11)
3- Exploratory Data Analysis (EDA): I explore the data to understand it features and data type I am going to work on and know if it has null values

```
In [6]: #test dataset
        test_bigmart_df.info()
```

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

```
In [7]: #From .info() we can see that:
        # Item_Weight and Outlet_Size in test and train data have non values.
        #to solve this problem :
        #Item_Weight: sense it is numaric I will replace it by the mean value.
        #Outlet_Size: sense it is catagoric I will replace it by the mode value.
```

```
#IN categorical variables we will see the number of unique values in each of them
train_bigmart_df.apply(lambda x: len(x.unique()))
```

```
Item_Identifier               1559
Item_Weight                    416
Item_Fat_Content                 5
Item_Visibility               7880
Item_Type                       16
Item_MRP                      5938
Outlet_Identifier               10
Outlet_Establishment_Year        9
Outlet_Size                      4
Outlet_Location_Type             3
Outlet_Type                      4
Item_Outlet_Sales             3493
dtype: int64
```

```
# We can see that there is 1559 products and 10 outlets.
```

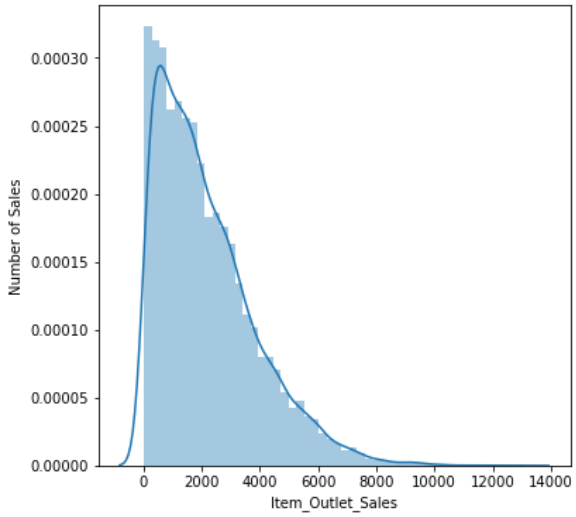
```
train_bigmart_df['Item_Fat_Content'].value_counts()
```

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

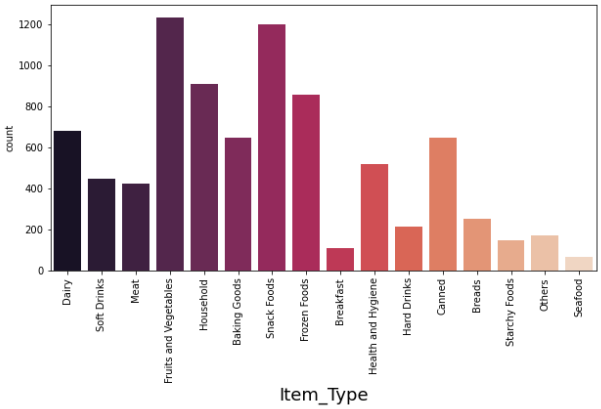
4-handling missing data

- The Item_Weight and Outlet_Size have missing data so I replace Item_Weight with the mean value, and the Outlet_Size with the mode of the Outlet_Size for the particular type of outlet.
- replacing the 0 values in Item_Visibility with the mean.

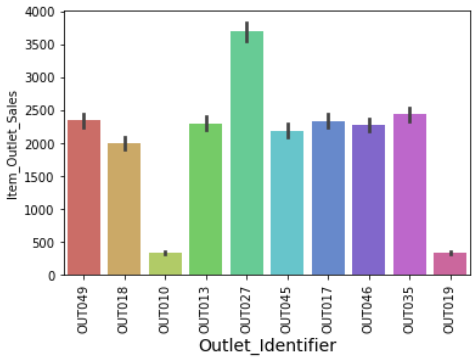
5- Data Visualization:



We can see that our target variable is skewed towards the right



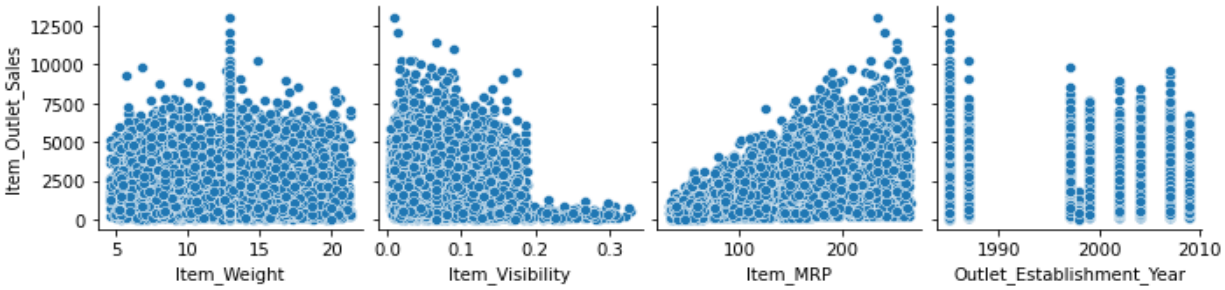
from the plot we can see that Fruits and Vegetables, Snack Foods and Household .are more sold than the other items



We can see that outlet 27 has the top sale

.

Plot shows the relationship between the target variable and the numirecal variable



-Item_Weight - The data is very spread, no specific pattern.

-Item_Visibility - Appears to be spread as well but some concentration around the (0,0) indicate small visibility items are not selling well is some cases.

-Item_MRP - Items with higher MRP soled better in most cases.

6-Data Preprocessing:

I convert **categorical** columns to **numerical** values using LabelEncoder()

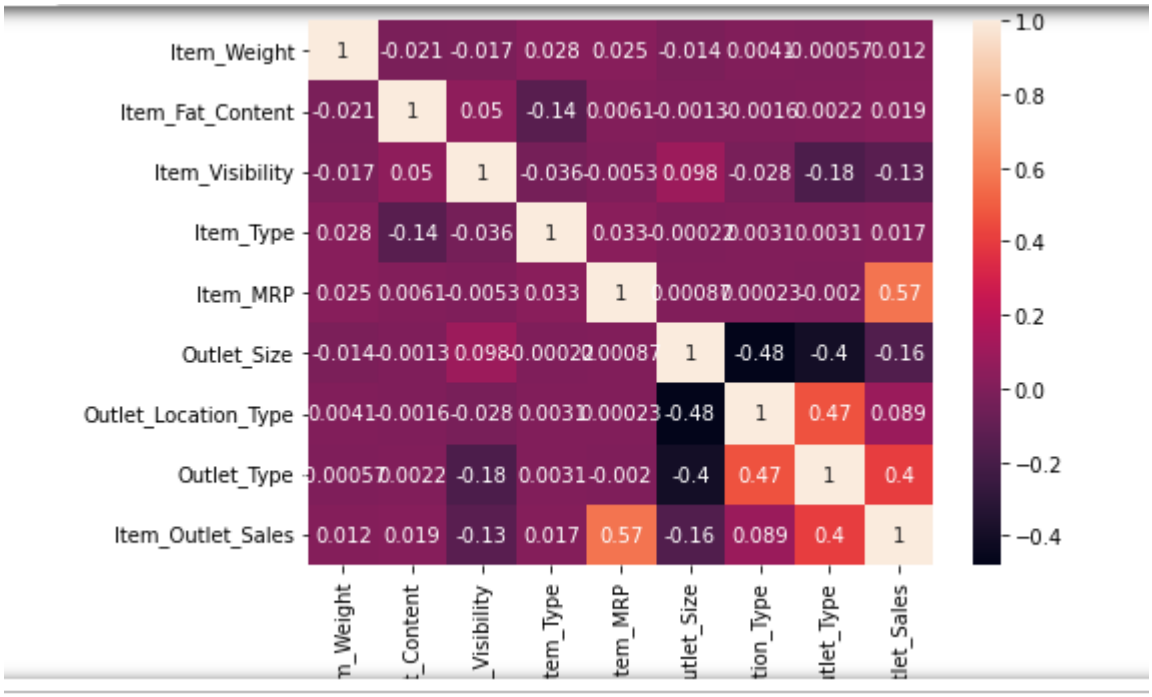
7-Corraltion Matrix

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
Item_Weight	1.000000	-0.021157	-0.017450	0.028015	0.024756	-0.014105	0.004088	-0.000566	0.011550
Item_Fat_Content	-0.021157	1.000000	0.049915	-0.139434	0.006063	-0.001262	-0.001598	0.002199	0.018719
Item_Visibility	-0.017450	0.049915	1.000000	-0.036000	-0.005259	0.097533	-0.027859	-0.179604	-0.134138
Item_Type	0.028015	-0.139434	-0.036000	1.000000	0.032651	-0.000218	0.003084	0.003053	0.017048
Item_MRP	0.024756	0.006063	-0.005259	0.032651	1.000000	0.000872	0.000232	-0.001975	0.567574
Outlet_Size	-0.014105	-0.001262	0.097533	-0.000218	0.000872	1.000000	-0.480075	-0.401373	-0.162753
Outlet_Location_Type	0.004088	-0.001598	-0.027859	0.003084	0.000232	-0.480075	1.000000	0.467219	0.089367
Outlet_Type	-0.000566	0.002199	-0.179604	0.003053	-0.001975	-0.401373	0.467219	1.000000	0.401522
Item_Outlet_Sales	0.011550	0.018719	-0.134138	0.017048	0.567574	-0.162753	0.089367	0.401522	1.000000

```
6]: corr['Item_Outlet_Sales'].sort_values(ascending=False)
```

```
6]: Item_Outlet_Sales       1.000000
    Item_MRP               0.567574
    Outlet_Type            0.401522
    Outlet_Location_Type   0.089367
    Item_Fat_Content       0.018719
    Item_Type              0.017048
    Item_Weight            0.011550
    Item_Visibility       -0.134138
    Outlet_Size           -0.162753
    Name: Item_Outlet_Sales, dtype: float64
```

we can see that Item_MRP have the most positive correlation and the Item_Visibility has the lowest correlation with our target variable.



8- split the data, train and test model

I am going to split the train data to train and validation data ,then I will:

1- train models on train
2- score them on validation
3- retrain best candidat on train and validation
4- score the final model on test.

Models I will use

- Random Forest Regressor
- Linear Regression