EDA ON BIGMART SALES ANALYSIS

Dataset Problems

Big Mart's data scientists collected sales data for 1559 commodities across 10 locations in various cities during 2013. Additionally, specific characteristics of each product and retailer were identified. This notebook will undertake data pre-processing and feature engineering to ensure the dataset is ready for use by the machine learning model. Additionally, this notebook will involve initial data exploration, EDA, and some hypothesis testing. These steps ensure a thorough understanding of the data and its nuances before applying it to the machine-learning model.

Variable Name	Description
Item_Identifier	Product ID
Item_Weight	Weight of product
Item_Fat_Content	Content of product (low fat or regular)
Item_Visibility	The percentage of all products in the store that are assigned to a specific product in the total display area
Item_Type	Category of product
Item_MRP	Maximum retail price of a product
Outlet_Identifier	Store ID
Outlet_Establishment_Year	Year the store established
Outlet_Size	Size of the store
Outlet_Location_Type	The type of city where the store is located
Outlet_Type	Type of the store
Item_Outlet_Sales	Sales of product

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Reading Dataset

```
df1=pd.read_csv('/content/bigmart.csv')
```

```
df1.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

```
df1.shape
```

(8523,12)

Finding Missing value

```
df1.isnull().sum()
```

```
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
```

```
df1.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)
```

Imputing Missing Values

```
f1['Item_Weight']=df1['Item_Weight'].fillna(df1['Item_Weight'].mean())
df1['Outlet_Size']=df1['Outlet_Size'].fillna(df1['Outlet_Size'].mode()[0])
```

```
df1.isnull().sum()
```

```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
dtype: int64
```

```
df1.describe()
```

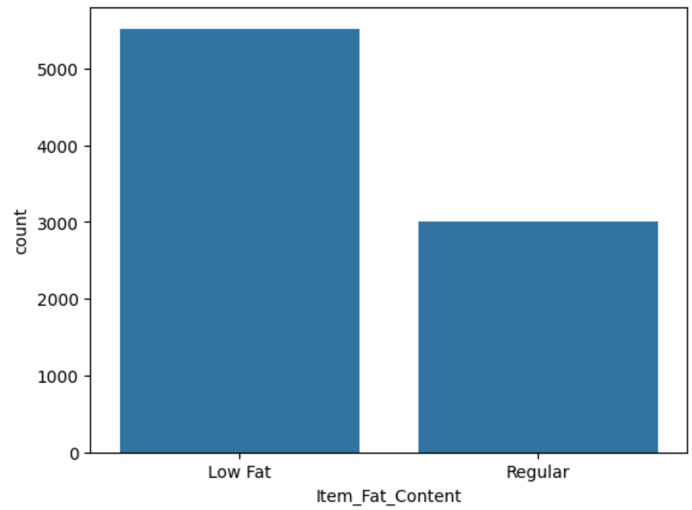
	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.226124	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	9.310000	0.026989	93.826500	1987.000000	834.247400
50%	12.857645	0.053931	143.012800	1999.000000	1794.331000
75%	16.000000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

#Data Cleaning or Preprocessing

```
fat_content_mapping = {
    'low fat': 'Low Fat',
    'LF': 'Low Fat',
    'reg': 'Regular',
    'low fat':'Low Fat',
    'reg':'Regular'
}

df1['Item_Fat_Content'] = df1['Item_Fat_Content'].replace(fat_content_mapping)


import seaborn as sns
sns.countplot(x='Item_Fat_Content',data=df1)
```

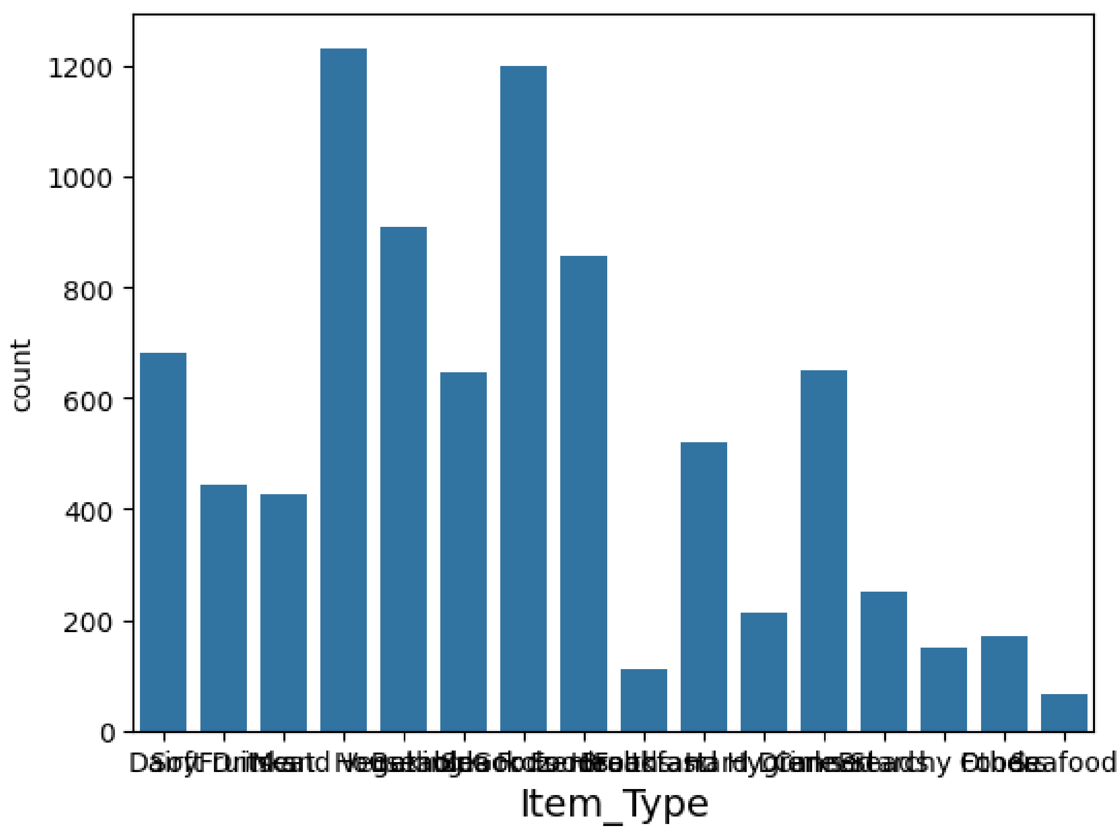
Visuvlizing the Relation ship between low fat and regular fat

<Axes: xlabel='Item_Fat_Content', ylabel='count'>




```
sns.countplot(x="Item_Type",data=df1)
plt.xlabel('Item_Type', fontsize=14)
plt.show()
```

<u>Visuvilzing Count of each item in bigmart</u>



```
df1.drop(columns= ['Outlet_Identifier'],inplace=True)
```

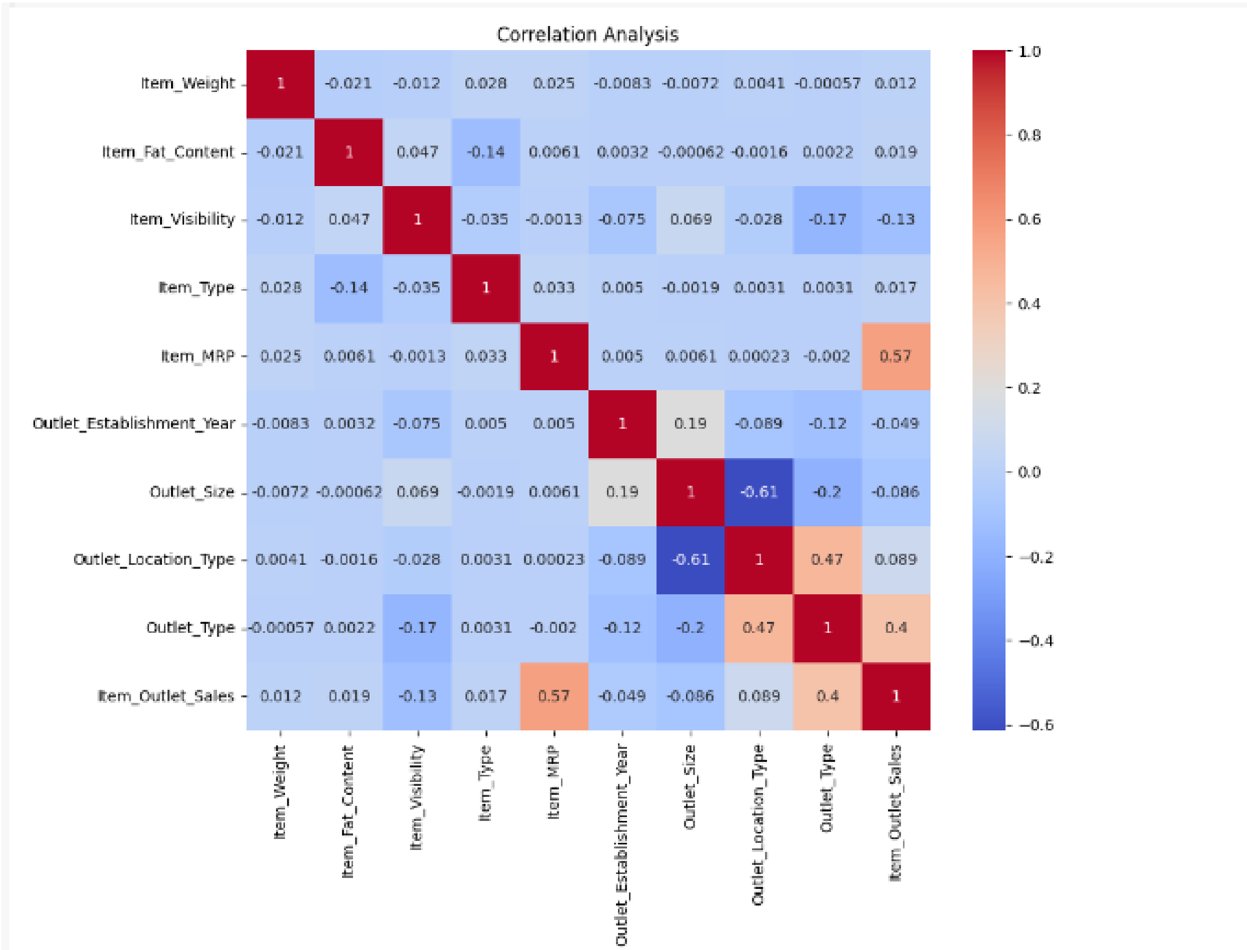
One Hot Encoding

```
from sklearn.preprocessing import LabelEncoder as le

df1['Item_Fat_Content']=le().fit_transform(df1['Item_Fat_Content'])
df1['Outlet_Size']=le().fit_transform(df1['Outlet_Size'])
df1['Outlet_Location_Type']=le().fit_transform(df1['Outlet_Location_Type'])
df1['Item_Type']=le().fit_transform(df1['Item_Type'])
df1['Outlet_Type']=le().fit_transform(df1['Outlet_Type'])


df1.drop(columns= ['Item_Identifier'],inplace=True)
df1.head()
```

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	9.30	0	0.016047	4	249.8092	1999	1	0	1	3735.1380
1	5.92	1	0.019278	14	48.2692	2009	1	2	2	443.4228
2	17.50	0	0.016760	10	141.6180	1999	1	0	1	2097.2700
3	19.20	1	0.000000	6	182.0950	1998	1	2	0	732.3800
4	8.93	0	0.000000	9	53.8614	1987	0	2	1	994.7052

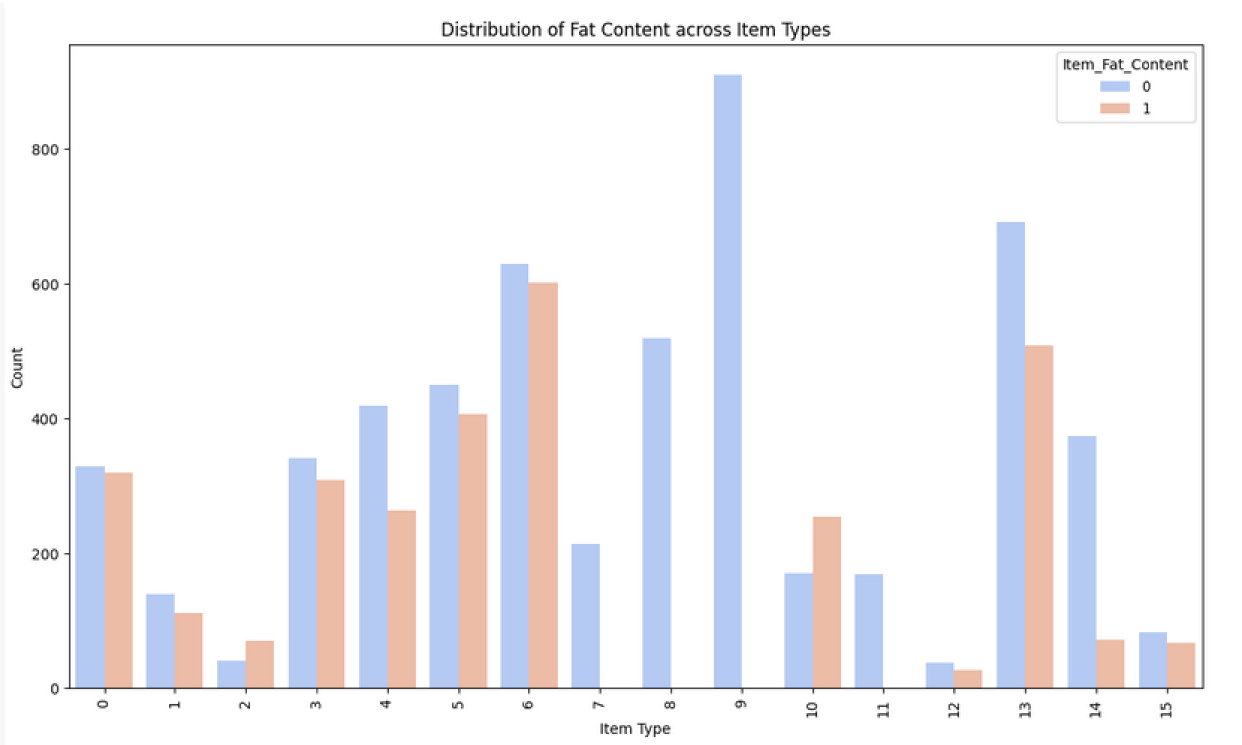
 # Correlation Identify the relation ship between the attributes

```
import matplotlib.pyplot as plt

corr_matrix=df1.corr()
plt.figure(figsize=(10,8))
sns.heatmap(corr_matrix,annot=True,cmap='coolwarm')
plt.title('Correlation Analysis')
plt.show()
```




```
plt.figure(figsize=(14, 8))
sns.countplot(x='Item_Type', hue='Item_Fat_Content', data=df1
            , palette='coolwarm')
plt.xticks(rotation=90)
plt.title('Distribution of Fat Content across Item Types')
plt.xlabel('Item Type')
plt.ylabel('Count')
plt.show()
```

#Distribution of Fat content in each item



Standardizing

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error,accuracy_score

df1.drop(columns='Item_Identifier',inplace=True)
```

Model Creation

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import accuracy_score
X = df1.drop(columns=['Item_Outlet_Sales'])
y = df1['Item_Outlet_Sales']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model selection and training
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
```

```python
# Evaluation
y_pred = model.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f'RMSE: {rmse}')
r2 = r2_score(y_test, y_pred)
print(r2)
```

```
RMSE: 1085.089372443194
0.5668020922486489
```