Big Sales Prediction using Random Forest Regressor

Get Understanding about Data set ¶

There are 12 variables in datasets
1.Item_identifier
2.Item_Weight
3.Item_Fat_Content
4.Item_Visibility
5.Item_Type
6.Item_MRP
7.Outlet_Identifier
8.Outlet_Establishment_year
9.Outlet_Size
10.Outlet_Location_Type
11.Outlet_Type
12.Item_Outlet_Sales

Import Library

In [42]:

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

Import CSV as DataFrame

In [43]:

```
df = pd.read_csv("Dataset-main/Dataset-main/Big Sales Data.csv")
```

In [44]:

```
df.head()
```

Out[44]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_I
0	FDT36	12.3	Low Fat	0.111448	Baking Goods	33.4874	
1	FDT36	12.3	Low Fat	0.111904	Baking Goods	33.9874	
2	FDT36	12.3	LF	0.111728	Baking Goods	33.9874	
3	FDT36	12.3	Low Fat	0.000000	Baking Goods	34.3874	
4	FDP12	9.8	Regular	0.045523	Baking Goods	35.0874	

Get the First five rows of Dataframe

In [45]:

```
df.head()
```

Out[45]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_I
0	FDT36	12.3	Low Fat	0.111448	Baking Goods	33.4874	
1	FDT36	12.3	Low Fat	0.111904	Baking Goods	33.9874	
2	FDT36	12.3	LF	0.111728	Baking Goods	33.9874	
3	FDT36	12.3	Low Fat	0.000000	Baking Goods	34.3874	
4	FDP12	9.8	Regular	0.045523	Baking Goods	35.0874	

Get Information of Dataframe

In [46]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14204 entries, 0 to 14203
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            14204 non-null  object
 1   Item_Weight                11815 non-null  float64
 2   Item_Fat_Content           14204 non-null  object
 3   Item_Visibility            14204 non-null  float64
 4   Item_Type                  14204 non-null  object
 5   Item_MRP                   14204 non-null  float64
 6   Outlet_Identifier          14204 non-null  object
 7   Outlet_Establishment_Year  14204 non-null  int64
 8   Outlet_Size                14204 non-null  object
 9   Outlet_Location_Type       14204 non-null  object
 10  Outlet_Type                14204 non-null  object
 11  Item_Outlet_Sales          14204 non-null  float64
dtypes: float64(4), int64(1), object(7)
memory usage: 1.3+ MB
```

Get Column Names

In [47]:

```
df.columns
```

Out[47]:

```
Index(['Item_Identifier', 'Item_Weight', 'Item_Fat_Content', 'Item_Visibilit
y',
       'Item_Type', 'Item_MRP', 'Outlet_Identifier',
       'Outlet_Establishment_Year', 'Outlet_Size', 'Outlet_Location_Type',
       'Outlet_Type', 'Item_Outlet_Sales'],
      dtype='object')
```

Get the Summery Statistics

In [48]:

```
df.describe()
```

Out[48]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	11815.000000	14204.000000	14204.000000	14204.000000	14204.000000
mean	12.788355	0.065953	141.004977	1997.830681	2185.836320
std	4.654126	0.051459	62.086938	8.371664	1827.479550
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.710000	0.027036	94.012000	1987.000000	922.135101
50%	12.500000	0.054021	142.247000	1999.000000	1768.287680
75%	16.750000	0.094037	185.855600	2004.000000	2988.110400
max	30.000000	0.328391	266.888400	2009.000000	31224.726950

Get Missing Values complete

In [49]:

```
df['Item_Weight'].fillna(df.groupby(['Item_Type'])['Item_Weight'].transform('mean'),inplace
```

In [50]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14204 entries, 0 to 14203
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            14204 non-null  object
 1   Item_Weight                14204 non-null  float64
 2   Item_Fat_Content           14204 non-null  object
 3   Item_Visibility            14204 non-null  float64
 4   Item_Type                  14204 non-null  object
 5   Item_MRP                   14204 non-null  float64
 6   Outlet_Identifier          14204 non-null  object
 7   Outlet_Establishment_Year  14204 non-null  int64
 8   Outlet_Size                14204 non-null  object
 9   Outlet_Location_Type       14204 non-null  object
 10  Outlet_Type                14204 non-null  object
 11  Item_Outlet_Sales          14204 non-null  float64
dtypes: float64(4), int64(1), object(7)
memory usage: 1.3+ MB
```

In [51]:

```
df.describe()
```

Out[51]:

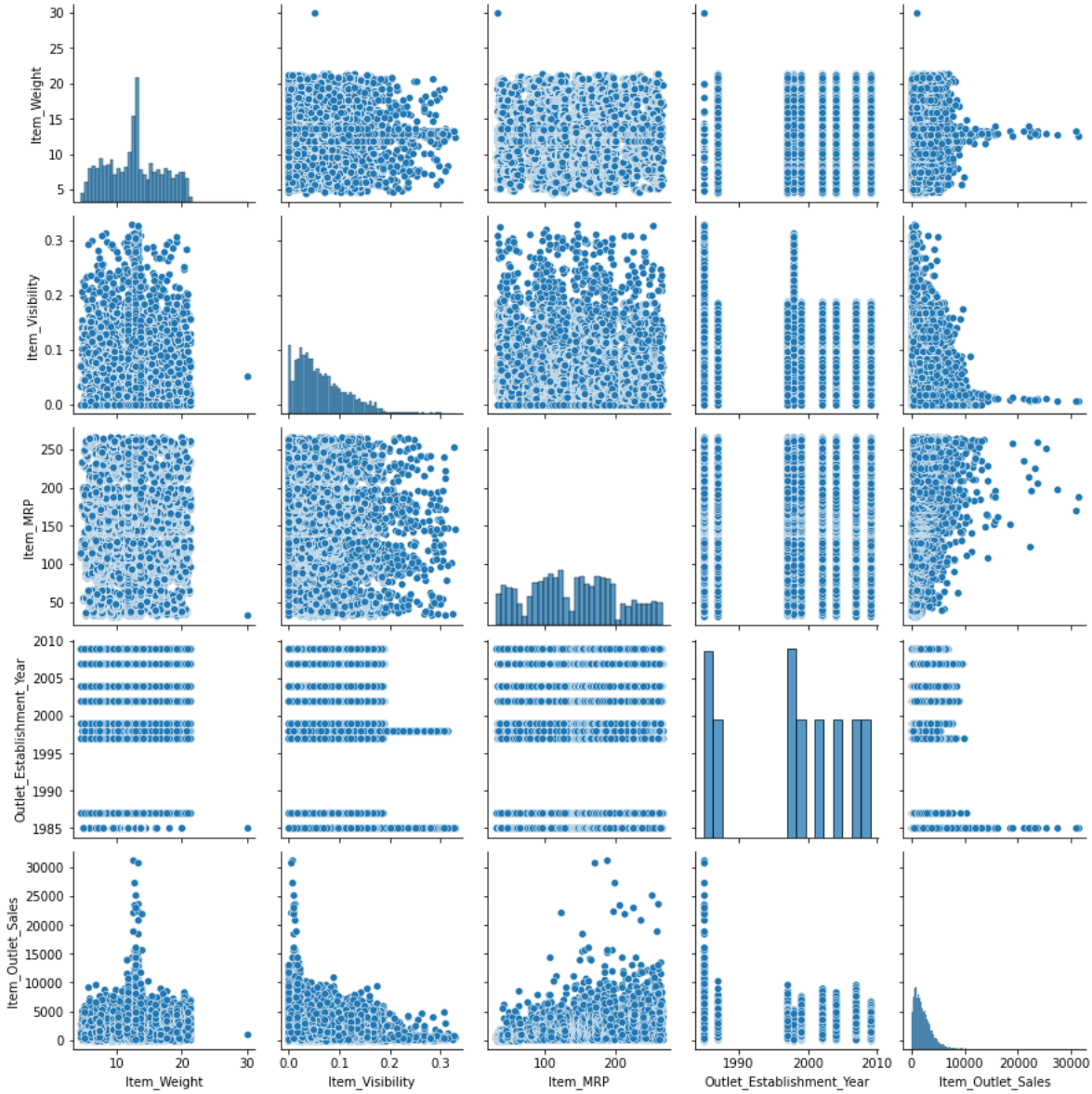
	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	14204.000000	14204.000000	14204.000000	14204.000000	14204.000000
mean	12.790642	0.065953	141.004977	1997.830681	2185.836320
std	4.251186	0.051459	62.086938	8.371664	1827.479550
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	9.300000	0.027036	94.012000	1987.000000	922.135101
50%	12.800000	0.054021	142.247000	1999.000000	1768.287680
75%	16.000000	0.094037	185.855600	2004.000000	2988.110400
max	30.000000	0.328391	266.888400	2009.000000	31224.726950

In [52]:

```
import seaborn as sns
sns.pairplot(df)
```

Out[52]:

```
<seaborn.axisgrid.PairGrid at 0x11a68296388>
```



Get Categories and Counts of Categorical Variables

In [53]:

```
df[['Item_Identifier']].value_counts()
```

Out[53]:

```
Item_Identifier
FDQ08              10
FDO24              10
FDQ19              10
FDQ28              10
FDQ31              10
                   ..
FDM52               7
FDM50               7
FDL50               7
FDM10               7
FDR51               7
Length: 1559, dtype: int64
```

In [132]:

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

Out[132]:

```
Item_Fat_Content
Low Fat             9185
1                   4824
regular              195
dtype: int64
```

In [213]:

```
df.replace({'Item_Fat_Content':{'LF':'Low Fat','reg':'Regular','low fat':'Low Fat','regular
```

In [216]:

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

Out[216]:

```
Item_Fat_Content
0                   9185
1                   5019
dtype: int64
```

In [217]:

```
df.replace({'Item_Fat_Content':{'Low Fat':0,'Regular':1}},inplace=True)
```

In [218]:

```
df[['Item_Type']].value_counts()
```

Out[218]:

```
Item_Type
0            11518
1             2406
2              280
dtype: int64
```

In [219]:

```
ealth and Hygiene':1,'Meat':0,'Soft Drinks':0,'Breads':0,'Hard Drinks':0,'Others':2,'Starchy
```

In [220]:

```
df[['Item_Type']].value_counts()
```

Out[220]:

```
Item_Type
0            11518
1             2406
2              280
dtype: int64
```

In [221]:

```
df[['Outlet_Identifier']].value_counts()
```

Out[221]:

```
Outlet_Identifier
0                    1559
1                    1553
2                    1550
3                    1550
4                    1550
5                    1548
6                    1546
7                    1543
8                     925
9                     880
dtype: int64
```

In [222]:

```
'Outlet_Identifier':{'OUT027':0,'OUT013':1,'OUT049':2,'OUT046':3,'OUT035':4,'OUT045':5,'OUT
```

In [223]:

```
df[['Outlet_Identifier']].value_counts()
```

Out[223]:

```
Outlet_Identifier
0                   1559
1                   1553
2                   1550
3                   1550
4                   1550
5                   1548
6                   1546
7                   1543
8                    925
9                    880
dtype: int64
```

In [224]:

```
df[['Outlet_Size']].value_counts()
```

Out[224]:

```
Outlet_Size
1              7122
0              5529
2              1553
dtype: int64
```

In [225]:

```
df.replace({'Outlet_Size':{'Small':0,'Medium':1,'High':2}},inplace=True)
```

In [226]:

```
df[['Outlet_Size']].value_counts()
```

Out[226]:

```
Outlet_Size
1              7122
0              5529
2              1553
dtype: int64
```

In [227]:

```
df[['Outlet_Location_Type']].value_counts()
```

Out[227]:

```
Outlet_Location_Type
2                      5583
1                      4641
0                      3980
dtype: int64
```

In [228]:

```
df.replace({'Outlet_Location_Type':{'Tier 1':0,'Tier 2':1,'Tier 3':2}},inplace=True)
```

In [229]:

```
df[['Outlet_Location_Type']].value_counts()
```

Out[229]:

```
Outlet_Location_Type
2                      5583
1                      4641
0                      3980
dtype: int64
```

In [230]:

```
df[['Outlet_Type']].value_counts()
```

Out[230]:

```
Outlet_Type
1              9294
0              1805
3              1559
2              1546
dtype: int64
```

In [231]:

```
df.replace({'Outlet_Type':{'Grocery Store':0,'Supermarket Type1':1,'Supermarket Type2':2,'S
```

In [232]:

```
df[['Outlet_Type']].value_counts()
```

Out[232]:

```
Outlet_Type
1              9294
0              1805
3              1559
2              1546
dtype: int64
```

In [233]:

```
df.head()
```

Out[233]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_I
0	FDT36	12.3	0	0.111448	0	33.4874	
1	FDT36	12.3	0	0.111904	0	33.9874	
2	FDT36	12.3	0	0.111728	0	33.9874	
3	FDT36	12.3	0	0.000000	0	34.3874	
4	FDP12	9.8	1	0.045523	0	35.0874	

In [234]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14204 entries, 0 to 14203
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            14204 non-null  object
 1   Item_Weight                14204 non-null  float64
 2   Item_Fat_Content           14204 non-null  int64
 3   Item_Visibility            14204 non-null  float64
 4   Item_Type                  14204 non-null  int64
 5   Item_MRP                   14204 non-null  float64
 6   Outlet_Identifier          14204 non-null  int64
 7   Outlet_Establishment_Year  14204 non-null  int64
 8   Outlet_Size                14204 non-null  int64
 9   Outlet_Location_Type       14204 non-null  int64
 10  Outlet_Type                14204 non-null  int64
 11  Item_Outlet_Sales          14204 non-null  float64
dtypes: float64(4), int64(7), object(1)
memory usage: 1.3+ MB
```

Get Shape of Dataframe

In [235]:

```
df.shape
```

Out[235]:

```
(14204, 12)
```

Define y (dependent or label or target variable) and X(indepenent or features or atribute Variable)

In [236]:

```
y = df['Item_Outlet_Sales']
```

In [237]:

```
y.shape
```

Out[237]:

```
(14204,)
```

In [238]:

```
y
```

Out[238]:

```
0         436.608721
1         443.127721
2         564.598400
3        1719.370000
4         352.874000
             ...
14199    4984.178800
14200    2885.577200
14201    2885.577200
14202    3803.676434
14203    3644.354765
Name: Item_Outlet_Sales, Length: 14204, dtype: float64
```

In [239]:

```
pe','Item_MRP','Outlet_Identifier','Outlet_Establishment_Year','Outlet_Size','Outlet_Locati
```

or use drop function to define X

In [240]:

```
X = df.drop(['Item_Identifier','Item_Outlet_Sales'],axis=1)
```

In [241]:

```
X.shape
```

Out[241]:

```
(14204, 10)
```

In [242]:

```
X
```

Out[242]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	O
0	12.300000	0	0.111448	0	33.4874	2	
1	12.300000	0	0.111904	0	33.9874	7	
2	12.300000	0	0.111728	0	33.9874	6	
3	12.300000	0	0.000000	0	34.3874	9	
4	9.800000	1	0.045523	0	35.0874	7	
...	...	...	...	...	...	...	
14199	12.800000	0	0.069606	0	261.9252	4	
14200	12.800000	0	0.070013	0	262.8252	7	
14201	12.800000	0	0.069561	0	263.0252	1	
14202	13.659758	0	0.069282	0	263.5252	0	
14203	12.800000	0	0.069727	0	263.6252	2	

14204 rows × 10 columns

Get X Variables Standardized

Standardzation of datasets is a common requirement for many machine learning estimators implemented in scikit-learn they might behave badly if the individual features do not more or less look like standard normally distributed data Gaussian with zero mean and unit variance

In [243]:

```
from sklearn.preprocessing import StandardScaler
```

In [244]:

```
sc = StandardScaler()
```

In [245]:

```
X_std = df[['Item_Weight','Item_Visibility','Item_MRP','Outlet_Establishment_Year']]
```

In [246]:

```
X_std = sc.fit_transform(X_std)
```

In [247]:

```
X_std
```

Out[247]:

```
array([[-0.11541705,  0.88413635, -1.73178716,  0.13968068],
       [-0.11541705,  0.89300616, -1.72373366,  1.09531886],
       [-0.11541705,  0.88958331, -1.72373366,  1.3342284 ],
       ...,
       [ 0.00220132,  0.07011952,  1.96538148, -1.29377659],
       [ 0.20444792,  0.06469366,  1.97343499, -1.53268614],
       [ 0.00220132,  0.07334891,  1.97504569,  0.13968068]])
```

In [248]:

```
_Year']] = pd.DataFrame(X_std,columns = [['Item_weight','Item_Visibility','Item_MRP','Outle
```

In [249]:

```
X
```

Out[249]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	O
0	-0.115417	0	0.884136	0	-1.731787	2	
1	-0.115417	0	0.893006	0	-1.723734	7	
2	-0.115417	0	0.889583	0	-1.723734	6	
3	-0.115417	0	-1.281712	0	-1.717291	9	
4	-0.703509	1	-0.397031	0	-1.706016	7	
...	...	...	...	...	...	...	
14199	0.002201	0	0.070990	0	1.947664	4	
14200	0.002201	0	0.078898	0	1.962160	7	
14201	0.002201	0	0.070120	0	1.965381	1	
14202	0.204448	0	0.064694	0	1.973435	0	
14203	0.002201	0	0.073349	0	1.975046	2	

14204 rows × 10 columns

Get Train Test Split

In [250]:

```
from sklearn.model_selection import train_test_split
```

In [251]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.1,random_state=2529)
```

In [252]:

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[252]:

```
((12783, 10), (1421, 10), (12783,), (1421,))
```

Get Model Train

In [253]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [254]:

```
rfr = RandomForestRegressor(random_state=2529)
```

In [255]:

```
rfr.fit(X_train,y_train)
```

Out[255]:

```
RandomForestRegressor(random_state=2529)
```

Get Model Prediction

In [256]:

```
y_pred = rfr.predict(X_test)
```

Get Model Evaluation

In [257]:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error,r2_score
```

In [258]:

```
mean_squared_error(y_test,y_pred)
```

Out[258]:

```
1611351.4218735117
```

In [259]:

```
mean_absolute_error(y_test,y_pred)
```

Out[259]:

828.4427522913378

In [260]:

```
r2_score(y_test,y_pred)
```

Out[260]:

0.5805891490212769

Get Visualization of Actual vs Predicted Results

In [266]:

```
import matplotlib.pyplot as plt
plt.scatter(y_test,y_pred)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted prices')
plt.title('Actual Price vs Predicted Price')
plt.show()
```