big sale pridiction using random forest regressor

```
import pandas as pd
```

```
import numpy as np
```

```
df=pd.read_csv(r'https://github.com/YBI-Foundation/Dataset/raw/main/Big%20Sales%20Data.csv
```

```
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_M
0	FDT36	12.3	Low Fat	0.111448	Baking Goods	33.48
1	FDT36	12.3	Low Fat	0.111904	Baking Goods	33.98
2	FDT36	12.3	LF	0.111728	Baking Goods	33.98
3	FDT36	12.3	Low Fat	0.000000	Baking Goods	34.38
4	FDP12	9.8	Regular	0.045523	Baking Goods	35.08

Saved successfully!

```
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
```

✓ 0s completed at 7:42 PM

```
 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
```

```
df.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Ou
count	11815.000000	14204.000000	14204.000000	14204.000000	1
mean	12.788355	0.065953	141.004977	1997.830681	
std	4.654126	0.051459	62.086938	8.371664	
min	4.555000	0.000000	31.290000	1985.000000	
25%	8.710000	0.027036	94.012000	1987.000000	
50%	12.500000	0.054021	142.247000	1999.000000	
75%	16.750000	0.094037	185.855600	2004.000000	
max	30.000000	0.328391	266.888400	2009.000000	3

```
df.columns
```

```
Index(['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'],
      dtype='object')
```

Saved successfully!

get missing value

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

```
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
```

```
df.describe()
```

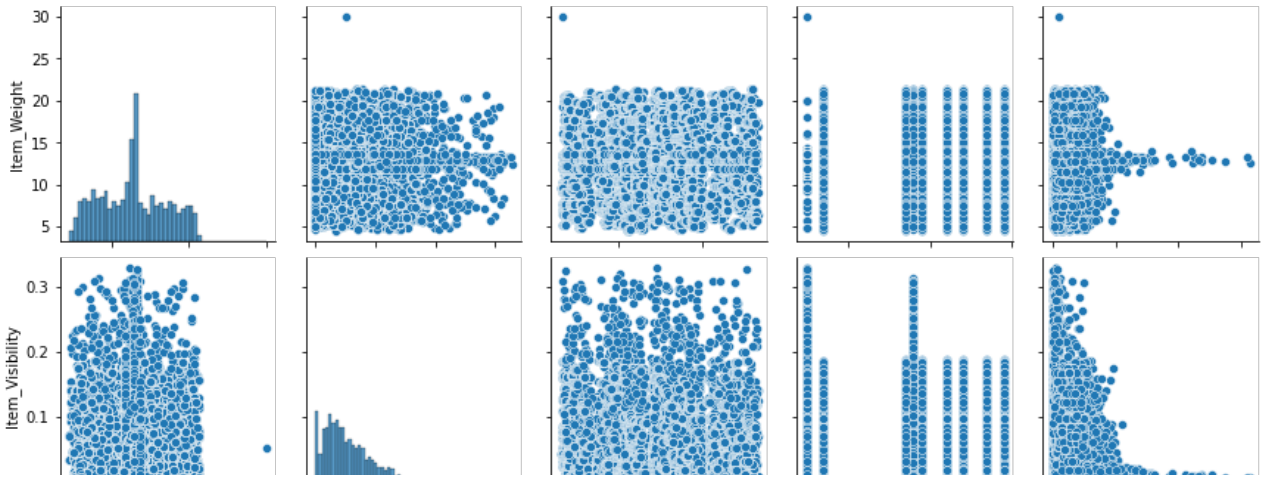
	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Ou
count	14204.000000	14204.000000	14204.000000	14204.000000	1
mean	12.790642	0.065953	141.004977	1997.830681	
std	4.251186	0.051459	62.086938	8.371664	
min	4.555000	0.000000	31.290000	1985.000000	
25%	9.300000	0.027036	94.012000	1987.000000	
50%	12.800000	0.054021	142.247000	1999.000000	
75%	16.000000	0.094037	185.855600	2004.000000	
max	30.000000	0.328391	266.888400	2009.000000	3

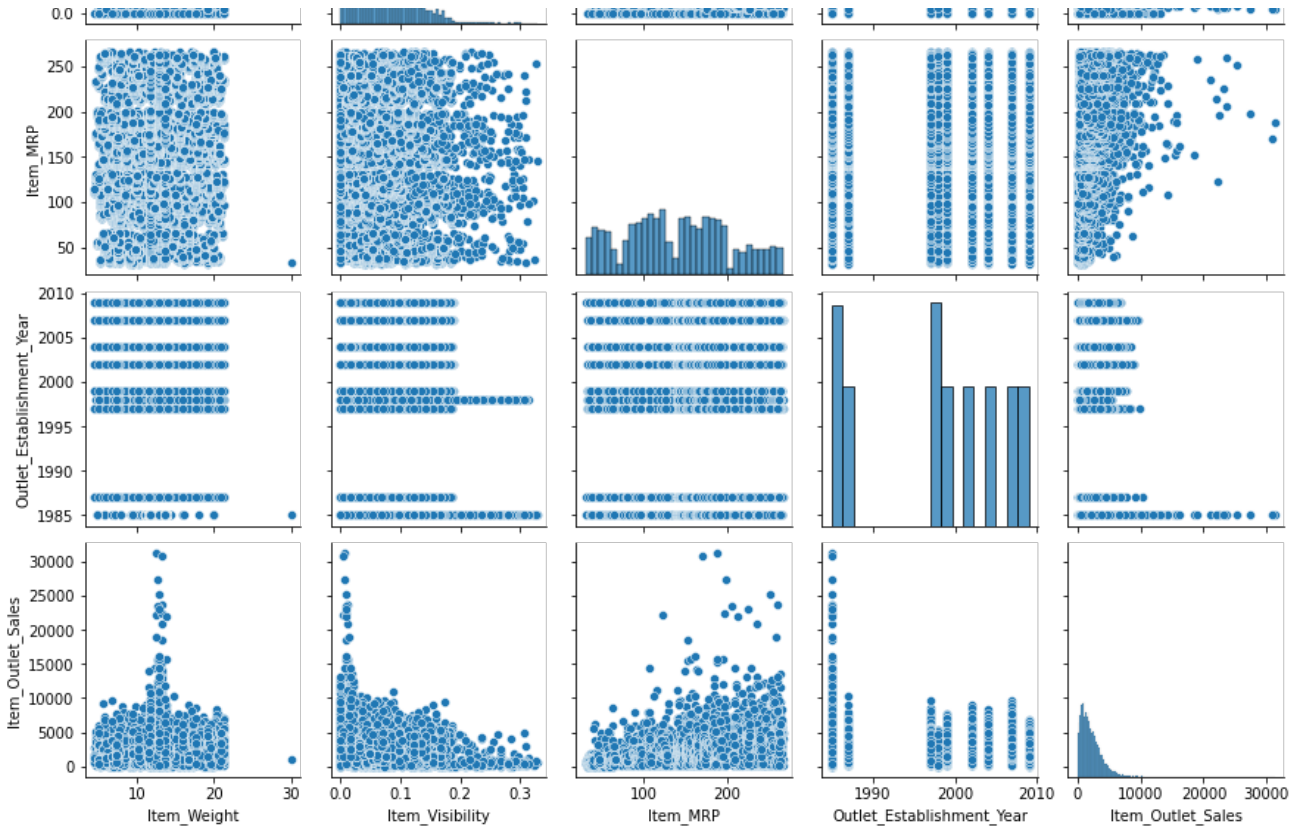
```
import seaborn as sns
```

```
sns.pairplot(df)
```

Saved successfully!

```
<seaborn.axisgrid.PairGrid at 0x7f6330a32890>
```





get categories and count the categorial variable

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

```
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
```

Saved successfully!

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

```
Item_Fat_Content
Low Fat             8485
Regular             4824
LF                   522
```

```
reg                 195
low fat             178
dtype: int64
```

```
df.replace({'Item_Fat_Content':{'LF':0,'Low Fat':1,'reg':2,'Regular':3,'low fat':4}},inplac
```

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

```
Item_Fat_Content
1                  8485
3                  4824
0                   522
2                   195
4                   178
dtype: int64
```

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

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

```
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
dtype: int64
```

Saved successfully!

```
df.replace({'Item_Type':{'Fruits and Vegetables':0,'Snack Foods':0,'Household':1,'Frozen Fo
                         'Meat':0,'Soft Drinks':0,'Breads':0,'Hard Drinks':0,'Others':2,'St
```

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

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

```
dtype: int64
```

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

```
Outlet_Identifier
OUT027              1559
OUT013              1553
OUT035              1550
OUT046              1550
OUT049              1550
OUT045              1548
OUT018              1546
OUT017              1543
OUT010               925
OUT019               880
dtype: int64
```

```
df.replace({'Outlet_Identifier':{'OUT027':0,
'OUT013':1,
'OUT035':2,
'OUT046':3,
'OUT049':4,
'OUT045':5,
'OUT018':6,
'OUT017':7,
'OUT010':8,
'OUT019':9,}},inplace=True)
```

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

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

Saved successfully!

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

```
Outlet_Size
Medium         7122
Small          5529
High           1553
dtype: int64
```

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

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

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

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

```
Outlet_Location_Type
Tier 3                 5583
Tier 2                 4641
Tier 1                 3980
dtype: int64
```

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

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

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

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

```
Outlet_Type
Supermarket Type1    9294
Grocery Store        1805
Supermarket Type3    1559
Supermarket Type2    1546
dtype: int64
```

Saved successfully!

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

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

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

```
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_M
0	FDT36	12.3	1	0.111448	0	33.48
1	FDT36	12.3	1	0.111904	0	33.98
2	FDT36	12.3	0	0.111728	0	33.98
3	FDT36	12.3	1	0.000000	0	34.38
4	FDP12	9.8	3	0.045523	0	35.08

```
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
```

Saved successfully!

get shape of the data frame

```
df.shape
```

```
(14204, 12)
```

define y(depedent or lable or target variable) and

x(indepedent or feature of attribute variable)

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

```
y.shape
```

```
(14204,)
```

```
y
```

```
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
```

```
x=df[['Item_Weight','Item_Fat_Content','Item_Visibility','Item_Type','Item_MRP','Outlet_Ide
      'Outlet_Location_Type','Outlet_Type',]]
```

use .drop function to define x

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

Saved successfully!

```
x.shape
```

```
(14204, 10)
```

```
x
```

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
0	12.300000	1	0.111448	0	33.4874	
1	12.300000	1	0.111904	0	33.9874	

2	12.300000	0	0.111728	0	33.9874
3	12.300000	1	0.000000	0	34.3874
4	9.800000	3	0.045523	0	35.0874
...	...	...	...	...	...
14199	12.800000	1	0.069606	0	261.9252
14200	12.800000	1	0.070013	0	262.8252
14201	12.800000	1	0.069561	0	263.0252
14202	13.659758	1	0.069282	0	263.5252
14203	12.800000	1	0.069727	0	263.6252

14204 rows × 10 columns

get variable standerdized

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

```
sc=StandardScaler()
```

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

```
x_std=sc.fit_transform(x_std)
```

Saved successfully!

```
x_std
```

```
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]])
```

```
x[['Item_Weight','Item_Visibility','Item_MRP','Outlet_Establishment_Year']]=pd.DataFrame(x_
```

```
x
```

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

14204 rows × 10 columns

get train test split

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=2529)

x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Saved successfully!

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

get model train

```
from sklearn.ensemble import RandomForestRegressor

rfr = RandomForestRegressor(random_state=2529)

rfr.fit(x_train, y_train)
```

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

get model prediction

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

```
y_pred.shape
```

```
(1421,)
```

```
y_pred
```

```
array([1453.8576999 ,  663.09849558, 1860.98185249, ..., 2225.18456175,
       3345.50213621,  423.82137329])
```

get model Evaluation

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

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

```
1635876.060775036
```

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

```
827.163932541674
```

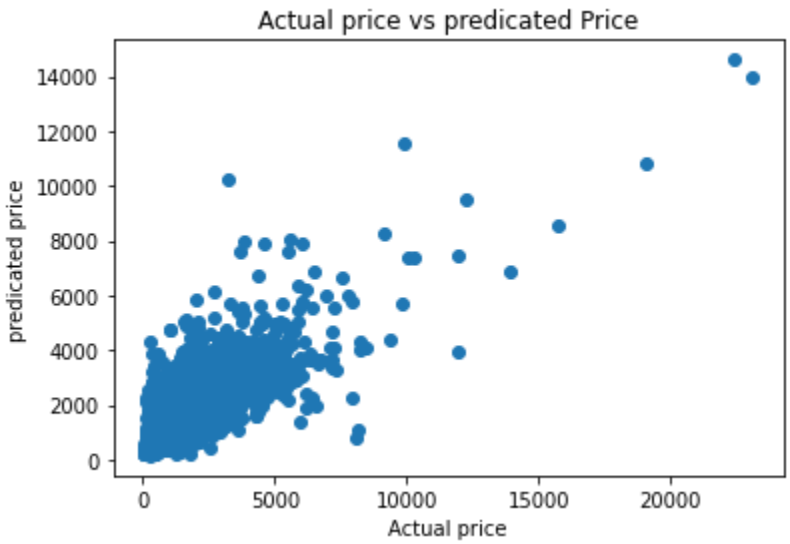
Saved successfully!

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

```
0.5742057496386178
```

get visualization of Actual vs predicted result

```
import matplotlib.pyplot as plt
plt.scatter(y_test,y_pred)
plt.xlabel("Actual price")
plt.ylabel("predicated price")
plt.title("Actual price vs predicated Price")
plt.show()
```



Saved successfully!