```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn as sk
```

```
df=pd.read_csv("C:\\Users\\HARIHARAPRASAD R\\Downloads\\archive (8)\\Train.csv")
```

```
df.isnull().sum()
```

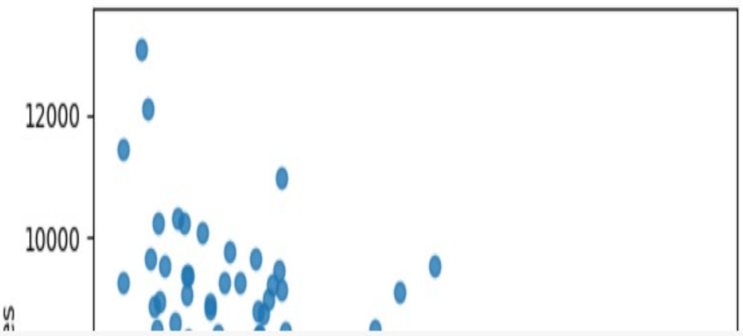
```
Item_Identifi
er                       0
Item_Weight           1463
Item_Fat_Content         0
Item_Visibili
ty                       0
Item_Type                0
Item_MRP                 0
Outlet_Identifier        0
Outlet_Establishment_Ye
ar                       0
Outlet_Size           2410
Outlet_Location_Type     0
Outlet_Type              0
Item_Outlet_Sales        0
dtype: int64
```

```
df.fillna(method="ffill")#cleaning & handling
missing data
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outle
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	
...	...	...	...	...	...	...	...	...	
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	OUT013	1987	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	OUT045	2002	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	OUT035	2004	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	OUT018	2009	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	OUT046	1997	

8523 rows × 12 columns

```
sns.regplot(x=df['Item_Visibility'],y=df['Item_Outlet_Sales'],fit_reg=False)
```



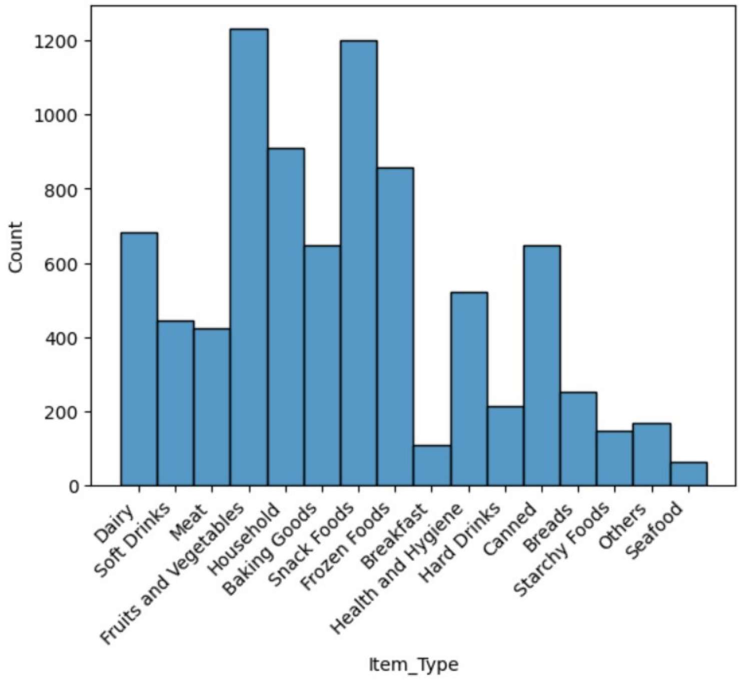
```
<Axes: xlabel='Item_Visibility', ylabel='Item_Outlet_Sales'>
```

```
print('product type and number of sales ')
print(df['Item_Type'].unique())
df['Item_Type'].value_counts()
```

```
product type and number of sales
['Dairy' 'Soft Drinks' 'Meat' 'Fruits and Vegetables' 'Household'
 'Baking Goods' 'Snack Foods' 'Frozen Foods' 'Breakfast'
 'Health and Hygiene' 'Hard Drinks' 'Canned' 'Breads' 'Starchy Foods'
 'Others' 'Seafood']
Item_Type
Fruits and Vegetables    1232
Snack Foods              1200
Household                 910
Frozen Foods              856
Dairy                     682
Canned                    649
Baking Goods              648
Health and Hygiene        520
Soft Drinks               445
Meat                      425
Breads                    251
Hard Drinks               214
Others                    169
Starchy Foods             148
Breakfast                 110
Seafood                    64
Name: count, dtype: int64
```

```
chart=sns.histplot(x='Item_Type',data=df,)
chart.set_xticklabels(chart.get_xticklabels(), rotation=45, horizontalalignment='right')
plt.show()
```

```
C:\Users\HARIHARAPRASAD R\AppData\Local\Temp\ipykernel_7368\2574513022.py:2: UserWarning: FixedFormatter should only be used
  together with FixedLocator chart.set_xticklabels(chart.get_xticklabels(), rotation=45, horizontalalignment='right')
```




```
print('item_type vs outlet_sales')
print(pd.crosstab(index=df['Item_Type'],columns='Item_Outlet_Sales',dropna=True,))
print("\njoint probalities of the item type and item_outletsales")
print(pd.crosstab(index=df['Item_Type'],columns='Item_Outlet_Sales',dropna=True,normalize=True))
```

```
item_type vs outlet_sales
col_0                Item_Outlet_Sales
```

```
Item_Type
Baking Goods                  648
Breads                        251
Breakfast                     110
Canned                        649
Dairy                         682
Frozen Foods                  856
Fruits and Vegetables        1232
Hard Drinks                   214
Health and Hygiene            520
Household                     910
Meat                          425
Others                        169
Seafood                        64
Snack Foods                  1200
Soft Drinks                   445
Starchy Foods                 148
```

```
joint probalities of the item type and item_outletsales
col_0                   Item_Outlet_Sales
Item_Type
Baking Goods                     0.076030
Breads                           0.029450
Breakfast                        0.012906
Canned                           0.076147
Dairy                            0.080019
Frozen Foods                     0.100434
Fruits and Vegetables            0.144550
Hard Drinks                      0.025109
Health and Hygiene               0.061011
Household                        0.106770
Meat                             0.049865
Others                           0.019829
Seafood                          0.007509
Snack Foods                      0.140795
Soft Drinks                      0.052212
Starchy Foods                    0.017365
```

```
chart1=sns.histplot(x=df['Item_Type'],y=df['Item_Outlet_Sales'],kde_kws=True)
chart1.set_xticklabels(chart1.get_xticklabels(), rotation=45, horizontalalignment='right')

plt.show()
print("\nsales describtion ")
print(df['Item_Outlet_Sales'].describe())
```

```
print('location wise sales')
pd.crosstab(index=df['Outlet_Location_Type'],columns='item_outlet_sales',dropna=True)
```

```
location wise sales
```

col_0	item_outlet_sales
Outlet_Location_Type	
Tier 1	2388
Tier 2	2785
Tier 3	3350

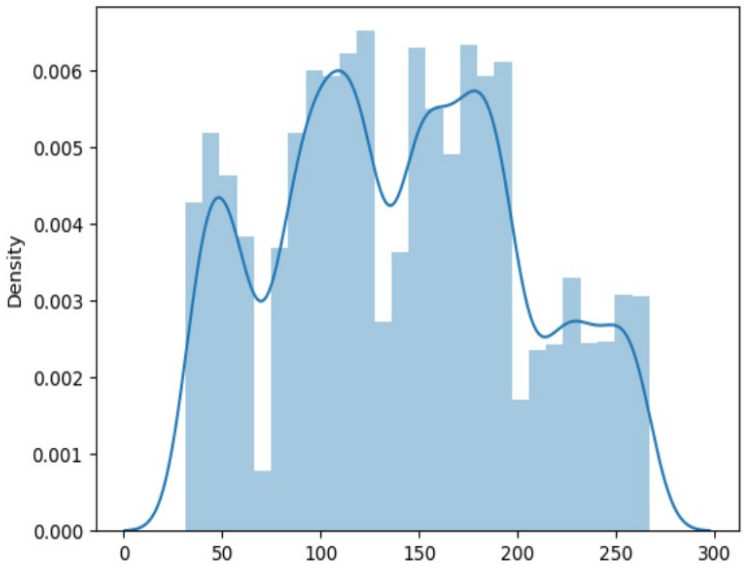
```
print("box distribution plot for item_weight")
sns.boxplot(y=df['Item_Weight'])
plt.show()
print('box distribution plot for item mrp')
sns.boxplot(y=df['Item_MRP'])
plt.show()
```

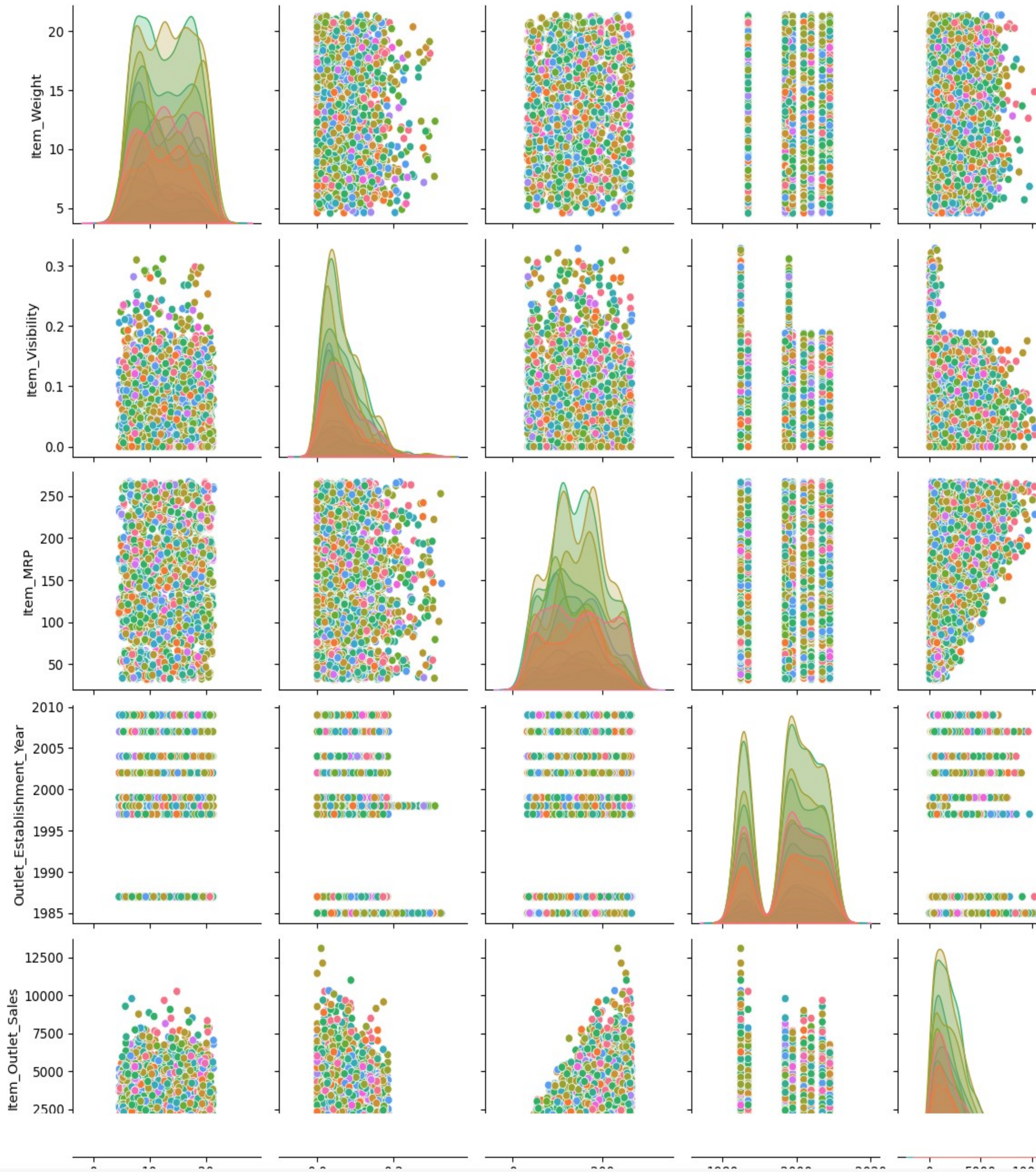
```
box distribution plot for item_weight
```

```
pd.crosstab(index=df['Outlet_Type'],columns=df['Outlet_Establishment_Year'],dropna=True)
```

```
print(df['Item_MRP'].describe())
sns.distplot(x=df['Item_MRP'],kde=True)
```

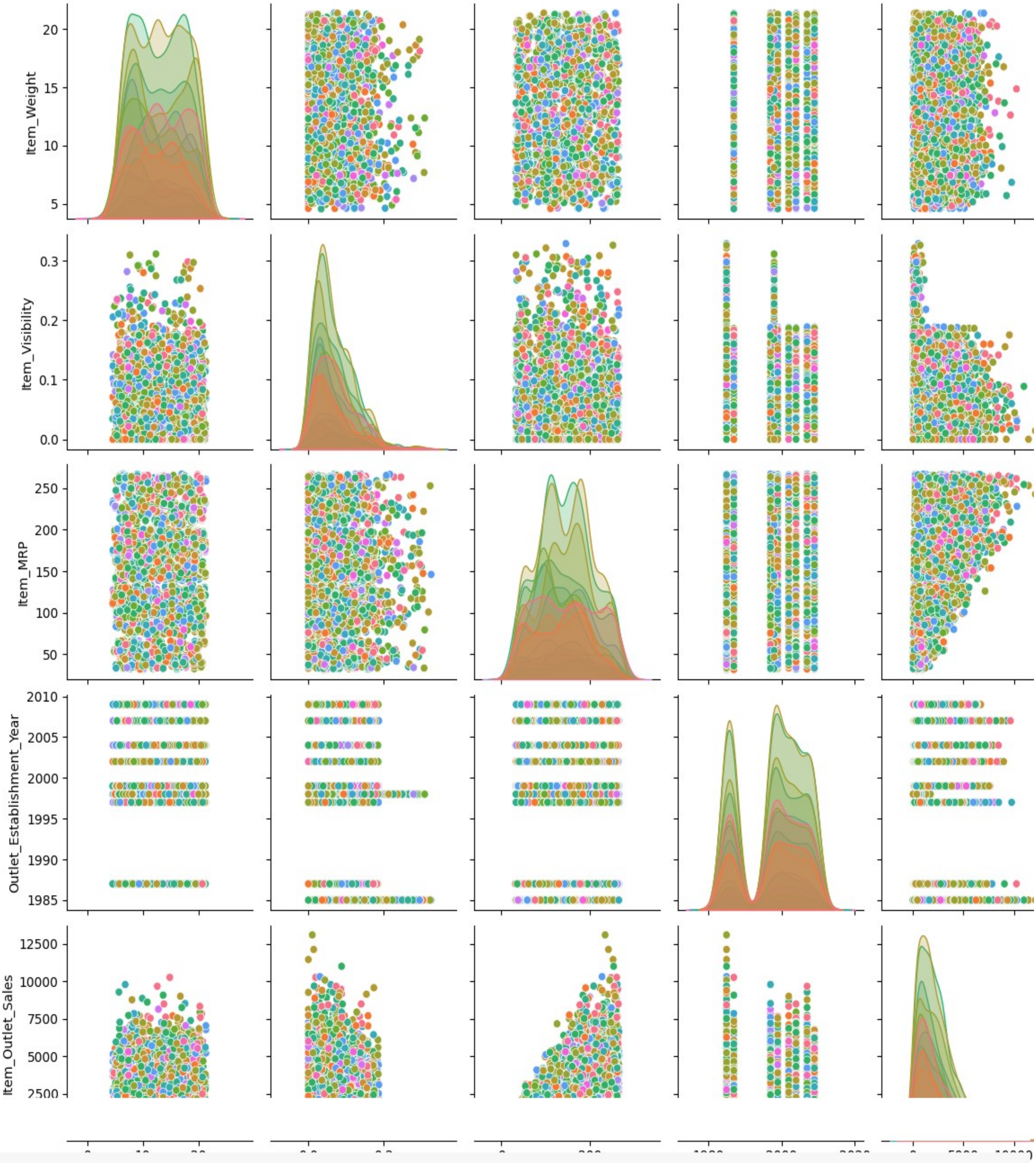
```
count  8523.000000
mean    140.992782
std      62.275067
min      31.290000
25%      93.826500
50%     143.012800
75%     185.643700
max     266.888400
Name: Item_MRP, dtype: float64
  sns.distplot(x=df['Item_MRP'],kde=True)
<Axes: ylabel='Density'>
```





Item_Weight
Item_Visibility
Item_MRP
Outlet_Establishment_Year
Item_Outlet_Sales

```
c:\Users\HARIHARAPRASAD R\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed
  self._figure.tight_layout(*args, **kwargs)
<seaborn.axisgrid.PairGrid at 0x25693caebc0>
```



```
#!pip install pandas numpy seaborn matplotlib klib dtale scikit-learn joblib pandas-profiling
```

```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df_train= pd.read_csv(r'D:\Python37\Projects\iNeuron Intership Projects\ML_BigMart Sales Prediction\Dataset\train.csv')
df_test= pd.read_csv(r'D:\Python37\Projects\iNeuron Intership Projects\ML_BigMart Sales Prediction\Dataset\test.csv')
```

```
df_test.isnull().sum()
```

a

```
Item_Identifier            0
Item_Weight              976
Item_Fat_Content           0
Item_Visibility            0
Item_Type                  0
Item_MRP                   0
Outlet_Identifier          0
Outlet_Establishment
_Year                      0
Outlet_Size             1606
Outlet_Location_Type       0
Outlet_Type                0
dtype: int64
```

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to
8522
Data columns (total 12
columns):
                               Non-Null
 #   Column                       Count Dtype
                           -------------
--- ------                 -            -----
                           852
 0   Item_Identifier       3    non-null  object
                           706
 1   Item_Weight           0    non-null  float64
                           852
 2   Item_Fat_Content      3    non-null  object
                           852
 3   Item_Visibility       3    non-null  float64
                           852
 4   Item_Type             3    non-null  object
                           852
 5   Item_MRP              3    non-null  float64
     Outlet_Identifie      852
 6   r                     3    non-null  object
     Outlet_Establishment_Ye   8523 non-
 7   ar                             null  int64
                           611
 8   Outlet_Size           3    non-null  object
                              8523 non-
 9   Outlet_Location_Type           null  object
                              8523 non-
 10  Outlet_Type                    null  object
     Item_Outlet_Sale         8523 non-
 11  s                              null  float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+
KB
```

```
df_train.describe(
                 )
```

	Item_Weight	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales

		Item_Visibility			
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247400
50%	12.600000	0.053931	143.012800	1999.000000	1794.331000
75%	16.850000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

Data Cleaning using Klib Library

```
# klib.clean - functions for cleaning datasets
klib.data_cleaning(df_train) # performs datacleaning (drop duplicates & empty rows/cols, adjust dtypes,...)
```

```
Shape of cleaned data: (8523, 10)Remaining NAs: 0

Changes:
Dropped rows: 0
   of which 0 duplicates. (Rows: [])
Dropped columns: 0
   of which 0 single valued.  Columns: []
Dropped missing values: 0
Reduced memory by at least: 0.08 MB (-12.31%)
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	
0	9.300000	Low Fat	0.016047	Dairy	249.809204	1999	Medium	Tier 1	Supe
1	5.920000	Regular	0.019278	Soft Drinks	48.269199	2009	Medium	Tier 3	Supe
2	17.500000	Low Fat	0.016760	Meat	141.617996	1999	Medium	Tier 1	Supe
3	19.200001	Regular	0.000000	Fruits and Vegetables	182.095001	1998	Medium	Tier 3	
4	8.930000	Low Fat	0.000000	Household	53.861401	1987	High	Tier 3	Supe
...	...	...	...	...	...	...	...	...	
8518	6.865000	Low Fat	0.056783	Snack Foods	214.521805	1987	High	Tier 3	Supe
8519	8.380000	Regular	0.046982	Baking Goods	108.156998	2002	Medium	Tier 2	Supe
8520	10.600000	Low Fat	0.035186	Health and Hygiene	85.122398	2004	Small	Tier 2	Supe
8521	7.210000	Regular	0.145221	Snack Foods	103.133202	2009	Medium	Tier 3	Supe
8522	14.800000	Low Fat	0.044878	Soft Drinks	75.467003	1997	Small	Tier 1	Supe

8523 rows × 10 columns

```
klib.clean_column_names(df_train) # cleans and standardizes column names, also called inside data_cleaning()
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	c
0	9.300	Low Fat	0.016047	Dairy	249.8092	1999	Medium	Tier 1	Superm
1	5.920	Regular	0.019278	Soft Drinks	48.2692	2009	Medium	Tier 3	Superm
2	17.500	Low Fat	0.016760	Meat	141.6180	1999	Medium	Tier 1	Superm
3	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	1998	Medium	Tier 3	G
4	8.930	Low Fat	0.000000	Household	53.8614	1987	High	Tier 3	Superm
...	...	...	...	...	...	...	...	...	
8518	6.865	Low Fat	0.056783	Snack Foods	214.5218	1987	High	Tier 3	Superm
8519	8.380	Regular	0.046982	Baking Goods	108.1570	2002	Medium	Tier 2	Superm
8520	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	2004	Small	Tier 2	Superm
8521	7.210	Regular	0.145221	Snack Foods	103.1332	2009	Medium	Tier 3	Superm
8522	14.800	Low Fat	0.044878	Soft Drinks	75.4670	1997	Small	Tier 1	Superm

8523 rows × 10 columns

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
```

```
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   item_weight                8523 non-null   float64
 1   item_fat_content           8523 non-null   object 
 2   item_visibility            8523 non-null   float64
 3   item_type                  8523 non-null   object 
 4   item_mrp                   8523 non-null   float64
 5   outlet_establishment_year  8523 non-null   int64  
 6   outlet_size                8523 non-null   object 
 7   outlet_location_type       8523 non-null   object 
 8   outlet_type                8523 non-null   object 
 9   item_outlet_sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(5)
memory usage: 666.0+ KB
```

```
df_train=klib.convert_datatypes(df_train) # converts existing to more efficient dtypes, also called inside data_cleaning()
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype   
---  ------                     --------------  -----   
 0   item_weight                8523 non-null   float32 
 1   item_fat_content           8523 non-null   category
 2   item_visibility            8523 non-null   float32 
 3   item_type                  8523 non-null   category
 4   item_mrp                   8523 non-null   float32 
 5   outlet_establishment_year  8523 non-null   int16   
 6   outlet_size                8523 non-null   category
 7   outlet_location_type       8523 non-null   category
 8   outlet_type                8523 non-null   category
 9   item_outlet_sales          8523 non-null   float32 
dtypes: category(5), float32(4), int16(1)
memory usage: 192.9 KB
```

```
klib.mv_col_handling(df_train)
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	
0	9.300000	Low Fat	0.016047	Dairy	249.809204	1999	Medium	Tier 1	Supe
1	5.920000	Regular	0.019278	Soft Drinks	48.269199	2009	Medium	Tier 3	Supe
2	17.500000	Low Fat	0.016760	Meat	141.617996	1999	Medium	Tier 1	Supe
3	19.200001	Regular	0.000000	Fruits and Vegetables	182.095001	1998	Medium	Tier 3	
4	8.930000	Low Fat	0.000000	Household	53.861401	1987	High	Tier 3	Supe
...	...	...	...	...	...	...	...	...	
8518	6.865000	Low Fat	0.056783	Snack Foods	214.521805	1987	High	Tier 3	Supe
8519	8.380000	Regular	0.046982	Baking Goods	108.156998	2002	Medium	Tier 2	Supe
8520	10.600000	Low Fat	0.035186	Health and Hygiene	85.122398	2004	Small	Tier 2	Supe
8521	7.210000	Regular	0.145221	Snack Foods	103.133202	2009	Medium	Tier 3	Supe
8522	14.800000	Low Fat	0.044878	Soft Drinks	75.467003	1997	Small	Tier 1	Supe

8523 rows × 10 columns

Preprocessing Task before Model Building

1) Label Encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
df_train['item_fat_content']= le.fit_transform(df_train['item_fat_content'])
df_train['item_type']= le.fit_transform(df_train['item_type'])
df_train['outlet_size']= le.fit_transform(df_train['outlet_size'])
df_train['outlet_location_type']= le.fit_transform(df_train['outlet_location_type'])
df_train['outlet_type']= le.fit_transform(df_train['outlet_type'])
```

```
df_train
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type
0	9.300000	1	0.016047	4	249.809204	1999	1	0	1
1	5.920000	2	0.019278	14	48.269199	2009	1	2	2
2	17.500000	1	0.016760	10	141.617996	1999	1	0	1
3	19.200001	2	0.000000	6	182.095001	1998	1	2	0
4	8.930000	1	0.000000	9	53.861401	1987	0	2	1
...	...	...	...	...	...	...	...	...	...
8518	6.865000	1	0.056783	13	214.521805	1987	0	2	1
8519	8.380000	2	0.046982	0	108.156998	2002	1	1	1
8520	10.600000	1	0.035186	8	85.122398	2004	2	1	1
8521	7.210000	2	0.145221	13	103.133202	2009	1	2	2
8522	14.800000	1	0.044878	14	75.467003	1997	2	0	1

8523 rows × 10 columns

2) **Splitting our data into train and test**

```
X=df_train.drop('item_outlet_sales',axis=1)
```

```
Y=df_train['item_outlet_sales']
```

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

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, random_state=101, test_size=0.2)
```

3) **Standarization**

```
X.describe()
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_typ
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000	8523.00000
mean	12.858088	1.369354	0.066132	7.226681	140.992767	1997.831867	1.170832	1.112871	1.20122
std	4.226130	0.644810	0.051598	4.209990	62.275051	8.371760	0.600327	0.812757	0.79645
min	4.555000	0.000000	0.000000	0.000000	31.290001	1985.000000	0.000000	0.000000	0.00000
25%	9.310000	1.000000	0.026989	4.000000	93.826500	1987.000000	1.000000	0.000000	1.00000
50%	12.857645	1.000000	0.053931	6.000000	143.012802	1999.000000	1.000000	1.000000	1.00000
75%	16.000000	2.000000	0.094585	10.000000	185.643700	2004.000000	2.000000	2.000000	1.00000
max	21.350000	4.000000	0.328391	15.000000	266.888397	2009.000000	2.000000	2.000000	3.00000

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
```

```
X_train_std= sc.fit_transform(X_train)
```

```
X_test_std= sc.transform(X_test)
```

```
X_train_std
```

```
array([[ 1.52290023, -0.57382672,   0.68469731, ..., -1.95699503,
        1.08786619, -0.25964107],
      [-1.239856 , -0.57382672, -0.09514746, ..., -0.28872895,
       -0.13870429, -0.25964107],
      [ 1.54667619, 0.97378032, -0.0083859 , ..., -
       0.28872895, -0.13870429, -0.25964107],
      ...,
      [-0.08197109, -0.57382672, -0.91916229, ...,   1.37953713,
       -1.36527477, -0.25964107],
      [-0.74888436,            0.97378032,   1.21363045, ..., -0.28872895,
       -0.13870429, -0.25964107],
      [ 0.67885675, -0.57382672, 1.83915361, ..., -
        0.28872895, 1.08786619, 0.98524841]])
```

```
X_test_std
```

```
array([[-0.43860916, -0.57382672, -0.21609253, ..., -0.28872895,
        1.08786619,0.98524841],
      [ 1.22570184, -0.57382672, -0.52943464, ..., -1.95699503,
        1.08786619, -0.25964107],
      [-1.2184578 ,            0.97378032,     0.16277341, ...,       1.37953713,
       -1.36527477, -0.25964107],
      ...,
      [ 0.65508101, -0.57382672,     0.8782423 , ..., -0.28872895,
        1.08786619, -1.50453056],
      [ 1.01171909, -0.57382672, -1.28409256, ..., -
        0.28872895, 1.08786619, 0.98524841],
      [-1.56558541, 0.97378032, -1.09265374, ..., -0.28872895,
       -0.13870429, -0.25964107]])
```

```
Y_train
```

```
3684   163.786804
1935   1607.241211
5142   1510.034424
4978   1784.343994
2299   3558.035156
          ...
599    5502.836914
56951436.796387
80062167.844727
13612700.484863
1547829.586792
Name: item_outlet_sales, Length: 6818, dtype: float32
```

```
Y_test
```

```
8179   904.822205
8355   2795.694092
3411   1947.464966
7089   872.863770
6954   2450.144043
          ...
1317   1721.093018
4996   914.809204
531    370.184814
38911358.232056
66292418.185547
Name: item_outlet_sales, Length: 1705, dtype: float32
```

```
import joblib
```

```
joblib.dump(sc,r'D:\BigMart-Sales-Prediction-using-Machine-Learning-main (1)\BigMart-Sales-Prediction-using-Machine-Learning-
main\sc.sav')
```

```
['D:\\BigMart-Sales-Prediction-using-Machine-Learning-main (1)\\BigMart-Sales-Prediction-using-Machine-Learning-
main\\sc.sav']
```

Model Building

```python
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
```

```python
lr.fit(X_train_std,Y_train)
```

```
LinearRegression()
```

```python
X_test.head()
```

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type
8179	11.000000	1	0.055163	8	100.335800	2009	1	2	2
8355	18.000000	1	0.038979	13	148.641800	1987	0	2	1
3411	7.720000	2	0.074731	1	77.598602	1997	2	0	1
7089	20.700001	1	0.049035	6	39.950600	2007	1	1	1
6954	7.550000	1	0.027225	3	152.934006	2002	1	1	1

```python
Y_pred_lr=lr.predict(X_test_std)
```

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

```python
print(r2_score(Y_test,Y_pred_lr))
print(mean_absolute_error(Y_test,Y_pred_lr))
print(np.sqrt(mean_squared_error(Y_test,Y_pred_lr)))
```

```
0.5041875773270632
880.9999044084501
1162.4412631603454
```

```python
joblib.dump(lr,r'D:\Python37\Projects\iNeuron Intership Projects\BigMart-Sales\models\lr.sav')
```

```
['D:\\Python37\\Projects\\iNeuron Intership Projects\\BigMart-Sales\\models\\lr.sav']
```

```python
from sklearn.ensemble import RandomForestRegressor
rf= RandomForestRegressor(n_estimators=1000)
```

```python
rf.fit(X_train_std,Y_train)
```

```
RandomForestRegressor(n_estimators=1000)
```

```python
Y_pred_rf= rf.predict(X_test_std)
```

```python
print(r2_score(Y_test,Y_pred_rf))
print(mean_absolute_error(Y_test,Y_pred_rf))
print(np.sqrt(mean_squared_error(Y_test,Y_pred_rf)))
```

```
0.5473443624488158
783.2320127534548
1110.6987486230885
```

Hyper Parameter Tuning

```
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV

# define models and parameters
model = RandomForestRegressor()
n_estimators = [10, 100, 1000]
max_depth=range(1,31)
min_samples_leaf=np.linspace(0.1,
1.0) max_features=["auto",
"sqrt", "log2"]

min_samples_split=np.linspace(0.1, 1.0, 10)

# define grid search
grid = dict(n_estimators=n_estimators)

#cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=101)

grid_search_forest = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1,


grid_search_forest.fit(X_train_std, Y_train)

# summarize results
print(f"Best: {grid_search_forest.best_score_:.3f} using {grid_search_forest.best_params_}")
means = grid_search_forest.cv_results_['mean_test_score']
stds = grid_search_forest.cv_results_['std_test_score']
params = grid_search_forest.cv_results_['params']

for mean, stdev, param in zip(means, stds, params):
   print(f"{mean:.3f} ({stdev:.3f}) with: {param}")
```

```
Fitting 2 folds for each of 3 candidates, totalling 6 fits
Best: 0.623 using {'n_estimators': 1000}
0.586 (0.007) with: {'n_estimators': 10}
0.618 (0.004) with: {'n_estimators': 100}
0.623 (0.003) with: {'n_estimators': 1000}
```

```
grid_search_forest.best_params_
```

```
{'n_estimators': 1000}
```

```
grid_search_forest.best_score_
```

```
0.6225226127048935
```

```
Y_pred_rf_grid=grid_search_forest.predict(X_test_std)
```

```
r2_score(Y_test,Y_pred_rf_grid)
```

```
0.6156338068690421
```

Save your model

```
joblib.dump(grid_search_forest,r'D:\Python37\Projects\iNeuron Intership Projects\ML_BigMart Sales Prediction\models\random_forest_grid.sav')
```

```
['D:\\Python37\\Projects\\iNeuron Intership Projects\\ML_BigMart Sales Prediction\\models\\random_forest_grid.sav']
```

```
model=joblib.load(r'D:\Python37\Projects\iNeuron Intership Projects\ML_BigMart Sales Prediction\models\random_forest_grid.sav')
```

```
                          scoring='r2',error_score=0,verbose=2,cv=2)
```