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

warnings.filterwarnings('ignore')

data = pd.read_csv('big_mart_Train.csv')

data.sample(5)
```

```
     Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
5251           FDB14       20.250          Regular         0.103142
4842           NCH55       16.350          Low Fat         0.034726
1801           NCV17       18.850          Low Fat         0.016105
2721           DRE03       19.600          Low Fat         0.024326
5514           FDW02        4.805          Regular         0.037699

                Item_Type  Item_MRP Outlet_Identifier  \
5251               Canned   94.6120            OUT018
4842            Household  125.6020            OUT049
1801  Health and Hygiene  130.2626            OUT035
2721                Dairy   48.5718            OUT018
5514                Dairy  125.5704            OUT046

      Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
5251                       2009      Medium               Tier 3
4842                       1999      Medium               Tier 1
1801                       2004       Small               Tier 2
2721                       2009      Medium               Tier 3
5514                       1997       Small               Tier 1

            Outlet_Type  Item_Outlet_Sales
5251  Supermarket Type2           652.4840
4842  Supermarket Type1          3036.0480
1801  Supermarket Type1          2360.9268
2721  Supermarket Type2           425.4462
5514  Supermarket Type1          3880.2824
```

Find Shape of Our Dataset (Number of Rows And Number of Columns)

```
data.shape
```

```
(8523, 12)
```

Get Information About Our Dataset Like Total Number Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

```
data.describe()
```

```
       Item_Weight  Item_Visibility     Item_MRP
Outlet_Establishment_Year  \
count  7060.000000      8523.000000  8523.000000
8523.000000
mean     12.857645         0.066132   140.992782
1997.831867
std       4.643456         0.051598    62.275067
8.371760
min       4.555000         0.000000    31.290000
1985.000000
25%       8.773750         0.026989    93.826500
1987.000000
50%      12.600000         0.053931   143.012800
1999.000000
75%      16.850000         0.094585   185.643700
2004.000000
max      21.350000         0.328391   266.888400
2009.000000

       Item_Outlet_Sales
count        8523.000000
mean         2181.288914
std          1706.499616
min            33.290000
25%           834.247400
50%          1794.331000
75%          3101.296400
max         13086.964800
```

Check Null Values In The Dataset

```
data.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
```

```
per = data.isnull().sum() * 100 / len(data)
print(per)
```

```
Item_Identifier                0.000000
Item_Weight                   17.165317
Item_Fat_Content               0.000000
Item_Visibility                0.000000
Item_Type                      0.000000
Item_MRP                       0.000000
Outlet_Identifier              0.000000
Outlet_Establishment_Year      0.000000
Outlet_Size                   28.276428
Outlet_Location_Type           0.000000
Outlet_Type                    0.000000
Item_Outlet_Sales              0.000000
dtype: float64
```

Taking Care of Duplicate Values

```
data.duplicated().any()
```

```
np.False_
```

Handling The missing Values

```
data['Item_Weight']
```

```
0        9.300
1        5.920
2       17.500
3       19.200
4        8.930
         ...
8518     6.865
8519     8.380
8520    10.600
8521     7.210
8522    14.800
Name: Item_Weight, Length: 8523, dtype: float64
```

```
data['Outlet_Size']
```

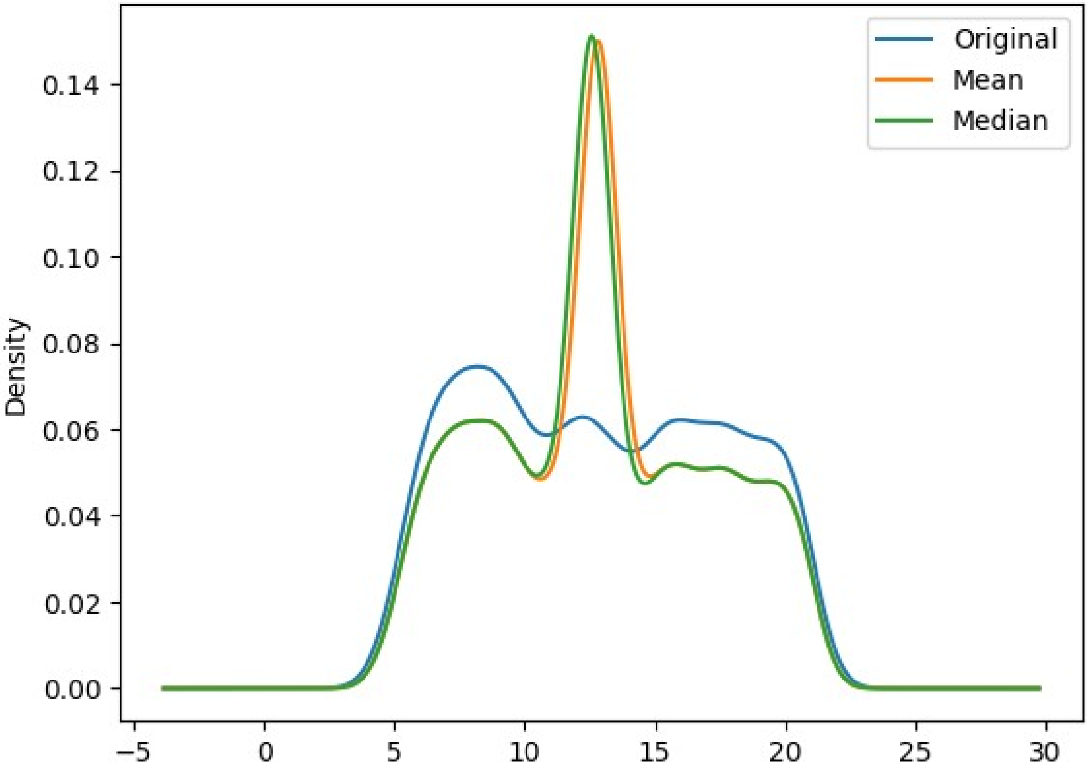
```
0       Medium
1       Medium
2       Medium
3          NaN
4         High
         ...
```

```
8518      High
8519       NaN
8520     Small
8521    Medium
8522     Small
Name: Outlet_Size, Length: 8523, dtype: object
```

Univariate Imputation

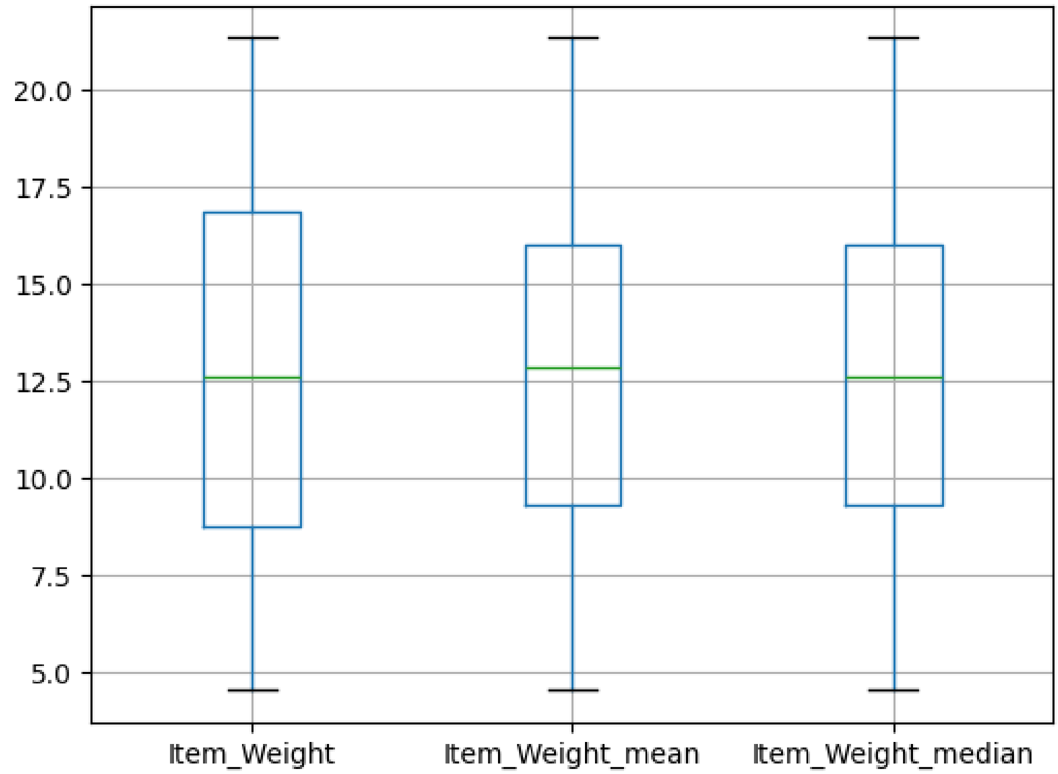
```
mean_weight = data['Item_Weight'].mean()
median_weight = data['Item_Weight'].median()

print(mean_weight,median_weight)

12.857645184135976 12.6

data['Item_Weight_mean']=data['Item_Weight'].fillna(mean_weight)
data['Item_Weight_median']=data['Item_Weight'].fillna(median_weight)

data.head(1)

  Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility
Item_Type  \
0           FDA15          9.3          Low Fat         0.016047
Dairy

   Item_MRP Outlet_Identifier  Outlet_Establishment_Year
Outlet_Size  \
0  249.8092            OUT049                       1999      Medium


  Outlet_Location_Type        Outlet_Type  Item_Outlet_Sales  \
0               Tier 1  Supermarket Type1           3735.138

   Item_Weight_mean  Item_Weight_median
0               9.3                 9.3

print("Original Weight variable variance",data['Item_Weight'].var())
print("Item Weight variance after mean
imputation",data['Item_Weight_mean'].var())
print("Item Weight variance after median
imputation",data['Item_Weight_median'].var())

Original Weight variable variance 21.561688259836558
Item Weight variance after mean imputation 17.86012173506058
Item Weight variance after median imputation 17.869561454073647

data['Item_Weight'].plot(kind = "kde",label="Original")

data['Item_Weight_mean'].plot(kind = "kde",label = "Mean")

data['Item_Weight_median'].plot(kind = "kde",label = "Median")
```

```
plt.legend()
plt.show()
```

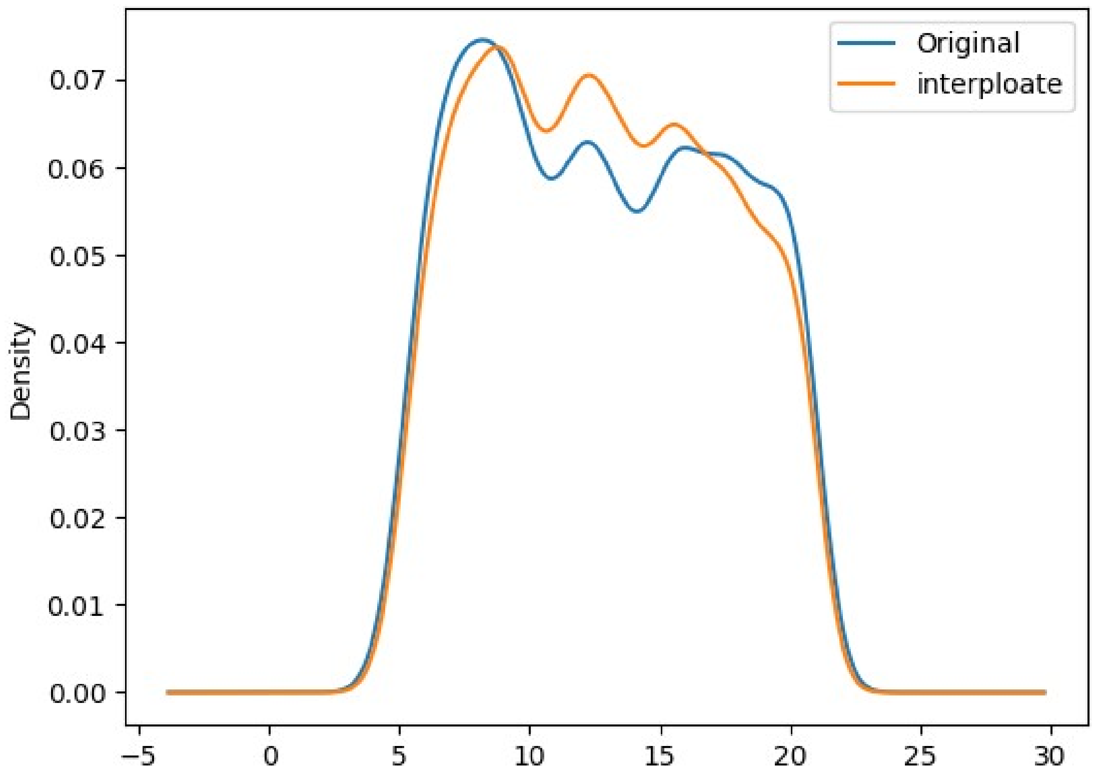


```
data[['Item_Weight','Item_Weight_mean','Item_Weight_median']].boxplot(
)
```

```
<Axes: >
```

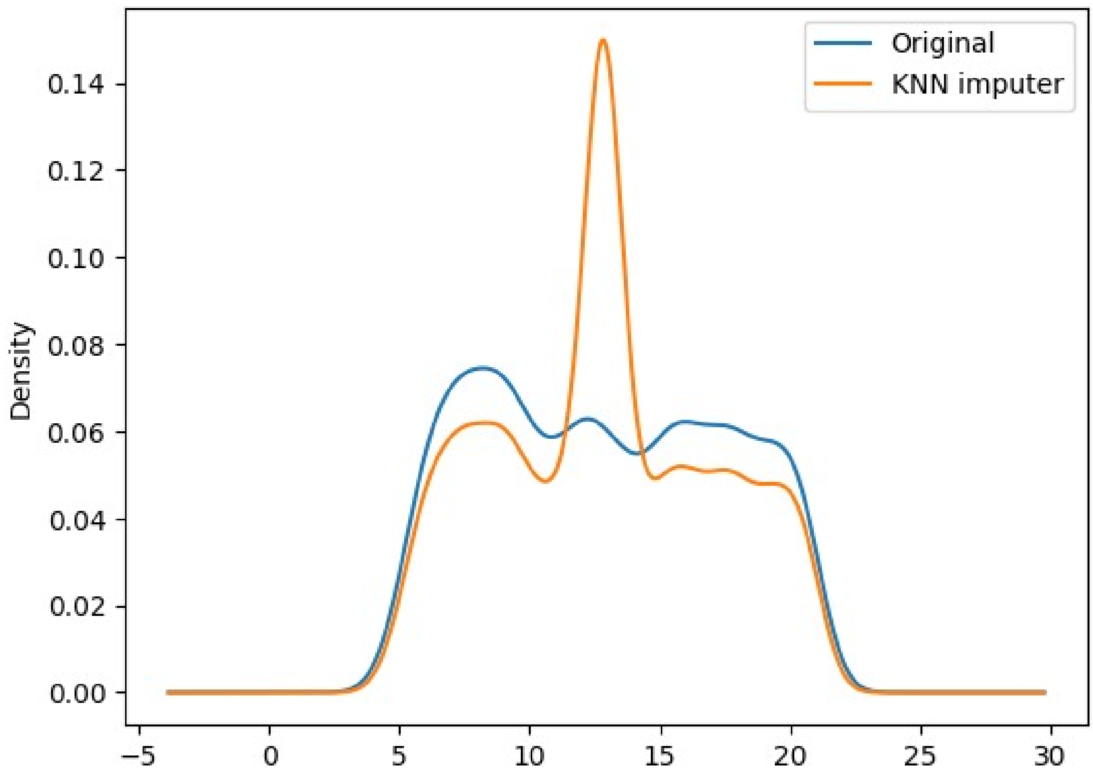


```
data['Item_Weight_interploate']=data['Item_Weight'].interpolate(method
="linear")

data['Item_Weight'].plot(kind = "kde",label="Original")

data['Item_Weight_interploate'].plot(kind = "kde",label =
"interploate")

plt.legend()
plt.show()
```



Multivariate Imputaion

```
from sklearn.impute import KNNImputer

knn = KNNImputer(n_neighbors=10,weights="distance")

data['knn_imputer']= knn.fit_transform(data[['Item_Weight']]).ravel()

data['Item_Weight'].plot(kind = "kde",label="Original")

data['knn_imputer'].plot(kind = "kde",label = "KNN imputer")

plt.legend()
plt.show()
```



```
data =
data.drop(['Item_Weight','Item_Weight_mean','Item_Weight_median','knn_
imputer'],axis=1)

data.head(1)

  Item_Identifier Item_Fat_Content  Item_Visibility Item_Type
Item_MRP  \
0           FDA15          Low Fat         0.016047     Dairy
249.8092

  Outlet_Identifier  Outlet_Establishment_Year Outlet_Size  \
0            OUT049                       1999      Medium

  Outlet_Location_Type        Outlet_Type  Item_Outlet_Sales  \
0               Tier 1  Supermarket Type1           3735.138

   Item_Weight_interploate
0                      9.3

data.isnull().sum()

Item_Identifier                0
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
Item_Weight_interploate         0
dtype: int64
```

Outlet_Size

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

```
Outlet_Size
Medium    2793
Small     2388
High       932
Name: count, dtype: int64
```

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

```
Outlet_Type
Supermarket Type1    5577
Grocery Store        1083
Supermarket Type3     935
Supermarket Type2     928
Name: count, dtype: int64
```

```
mode_outlet = data.pivot_table(values='Outlet_Size',columns='Outlet_Type',aggfunc=(lambda x:x.mode()[0]))
```

```
mode_outlet
```

```
Outlet_Type Grocery Store Supermarket Type1 Supermarket Type2  \
Outlet_Size         Small             Small            Medium   

Outlet_Type Supermarket Type3  
Outlet_Size            Medium  
```

```
missing_values = data['Outlet_Size'].isnull()
```

```
missing_values
```

```
0       False
1       False
2       False
3        True
4       False
        ...
```

```
8518    False
8519     True
8520    False
8521    False
8522    False
Name: Outlet_Size, Length: 8523, dtype: bool
```

```
data.loc[missing_values,'Outlet_Size'] =
data.loc[missing_values,'Outlet_Type'].apply(lambda x :mode_outlet[x])
```

```
data.isnull().sum()
```

```
Item_Identifier              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
Item_Weight_interploate      0
dtype: int64
```

Item_Fat_Content

```
data.columns
```

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

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

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

```
data.replace({'Item_Fat_Content':{'Low Fat':'LF','low
fat':'LF','reg':'Regular'}},inplace=True)
```

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

```
Item_Fat_Content
LF         5517
Regular    3006
Name: count, dtype: int64
```

Item_Visibility

```
data.columns
```

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

```
data['Item_Visibility'].value_counts()
```

```
Item_Visibility
0.000000    526
0.076975      3
0.107274      2
0.074613      2
0.045166      2
           ...
0.056783      1
0.046982      1
0.035186      1
0.145221      1
0.016827      1
Name: count, Length: 7880, dtype: int64
```

```
data['Item_Visibility_interpolate']=data['Item_Visibility'].replace(0,
np.nan).interpolate(method='linear')
```

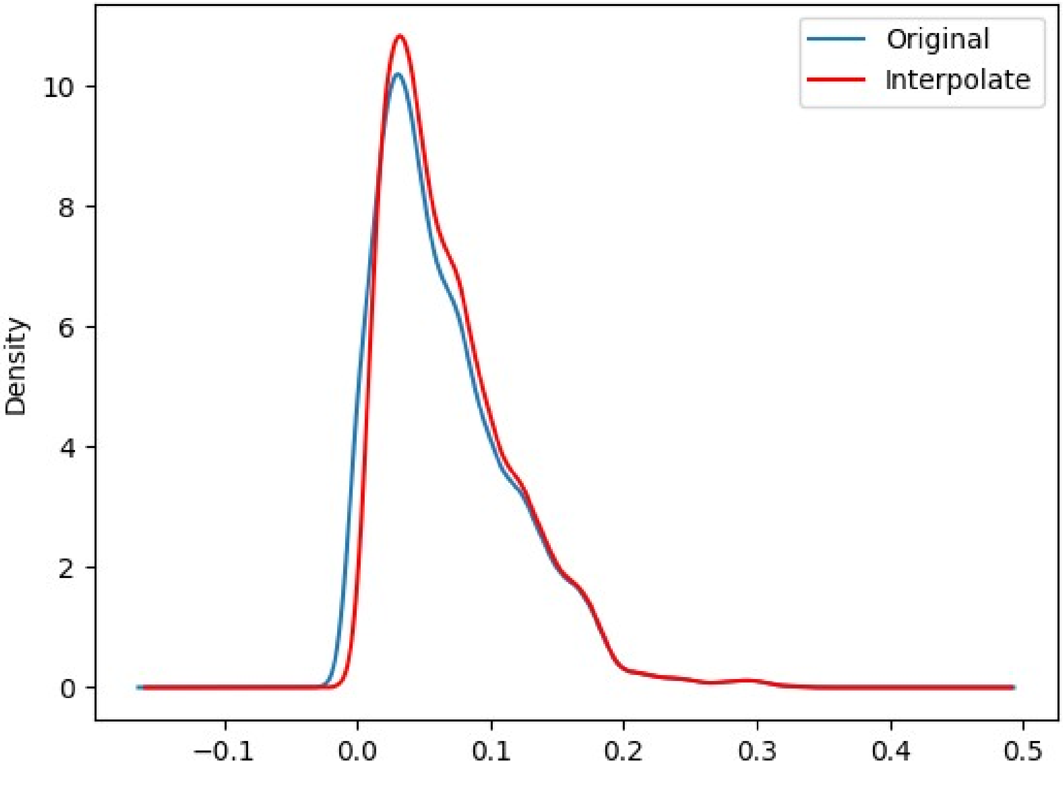
```
data.head(1)
```

```
  Item_Identifier Item_Fat_Content  Item_Visibility Item_Type
Item_MRP  \
0           FDA15               LF         0.016047     Dairy
249.8092

  Outlet_Identifier  Outlet_Establishment_Year Outlet_Size  \
0            OUT049                       1999      Medium

  Outlet_Location_Type        Outlet_Type  Item_Outlet_Sales  \
0               Tier 1  Supermarket Type1           3735.138

   Item_Weight_interploate  Item_Visibility_interpolate
0                      9.3                     0.016047
```

```
data['Item_Visibility_interpolate'].value_counts()
```

```
Item_Visibility_interpolate
0.076975    3
0.096592    2
0.093308    2
0.076792    2
0.107274    2
           ..
0.070712    1
0.036133    1
0.124111    1
0.094146    1
0.138190    1
Name: count, Length: 8405, dtype: int64
```

```
data['Item_Visibility'].plot(kind="kde",label="Original")

data['Item_Visibility_interpolate'].plot(kind="kde",color='red',label=
"Interpolate")

plt.legend()
plt.show()
```



```
data = data.drop('Item_Visibility',axis=1)
```

```
data.head(1)
```

```
  Item_Identifier Item_Fat_Content Item_Type  Item_MRP 
Outlet_Identifier  \
0           FDA15               LF     Dairy  249.8092           
OUT049   

   Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
0                       1999      Medium               Tier 1   

         Outlet_Type  Item_Outlet_Sales  Item_Weight_interploate  \
0  Supermarket Type1           3735.138                      9.3   

   Item_Visibility_interpolate  
0                     0.016047  
```

Item_Type

```
data.columns
```

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

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

```
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
```

Item_Identifier

```
data.columns
```

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

```
data['Item_Identifier'].value_counts().sample(5)
```

```
Item_Identifier
FDW37    3
FDX59    5
DRM37    7
FDB27    6
FDV47    6
Name: count, dtype: int64
```

```
data['Item_Identifier'] =data['Item_Identifier'].apply(lambda x :
x[:2])
```

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

```
Item_Identifier
FD    6125
NC    1599
DR     799
Name: count, dtype: int64
```

Outlet_Establishment_Year

```
data.columns
```

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

```
data['Outlet_Establishment_Year']
```

```
0       1999
1       2009
2       1999
3       1998
4       1987
        ...
8518    1987
```

```
8519    2002
8520    2004
8521    2009
8522    1997
Name: Outlet_Establishment_Year, Length: 8523, dtype: int64
```

```
import datetime as dt
```

```
current_year = dt.datetime.today().year
```

```
current_year
```

```
2025
```

```
data['Outlet_age']= current_year - data['Outlet_Establishment_Year']
```

```
data.head(1)
```

```
  Item_Identifier Item_Fat_Content Item_Type  Item_MRP
Outlet_Identifier  \
0              FD               LF     Dairy  249.8092
OUT049

   Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
0                       1999      Medium               Tier 1

         Outlet_Type  Item_Outlet_Sales  Item_Weight_interploate  \
0  Supermarket Type1           3735.138                      9.3

   Item_Visibility_interpolate  Outlet_age
0                     0.016047          26
```

```
data = data.drop('Outlet_Establishment_Year',axis=1)
```

```
data.head()
```

```
  Item_Identifier Item_Fat_Content              Item_Type  Item_MRP  \
0              FD               LF                  Dairy  249.8092
1              DR          Regular            Soft Drinks   48.2692
2              FD               LF                   Meat  141.6180
3              FD          Regular  Fruits and Vegetables  182.0950
4              NC               LF              Household   53.8614

  Outlet_Identifier Outlet_Size Outlet_Location_Type
Outlet_Type  \
0            OUT049      Medium               Tier 1  Supermarket
Type1
1            OUT018      Medium               Tier 3  Supermarket
Type2
2            OUT049      Medium               Tier 1  Supermarket
Type1
3            OUT010       Small               Tier 3      Grocery
```

```
Store
4             OUT013        High               Tier 3  Supermarket
Type1

   Item_Outlet_Sales  Item_Weight_interploate
Item_Visibility_interpolate  \
0          3735.1380                     9.30
0.016047
1           443.4228                     5.92
0.019278
2          2097.2700                    17.50
0.016760
3           732.3800                    19.20
0.015755
4           994.7052                     8.93
0.014751

   Outlet_age
0          26
1          16
2          26
3          27
4          38
```

Handling Categorical Columns

```python
from sklearn.preprocessing import OrdinalEncoder

data_encoded = data.copy()

cat_cols = data.select_dtypes(include=['object']).columns

for col in cat_cols:
    oe = OrdinalEncoder()
    data_encoded[col]=oe.fit_transform(data_encoded[[col]])
    print(oe.categories_)
```

```
[array(['DR', 'FD', 'NC'], dtype=object)]
[array(['LF', 'Regular'], dtype=object)]
[array(['Baking Goods', 'Breads', 'Breakfast', 'Canned', 'Dairy',
       'Frozen Foods', 'Fruits and Vegetables', 'Hard Drinks',
       'Health and Hygiene', 'Household', 'Meat', 'Others', 'Seafood',
       'Snack Foods', 'Soft Drinks', 'Starchy Foods'], dtype=object)]
[array(['OUT010', 'OUT013', 'OUT017', 'OUT018', 'OUT019', 'OUT027',
       'OUT035', 'OUT045', 'OUT046', 'OUT049'], dtype=object)]
[array(['High', 'Medium', 'Small'], dtype=object)]
[array(['Tier 1', 'Tier 2', 'Tier 3'], dtype=object)]
[array(['Grocery Store', 'Supermarket Type1', 'Supermarket Type2',
       'Supermarket Type3'], dtype=object)]
```

```
data_encoded.head(3)
```

```
   Item_Identifier  Item_Fat_Content  Item_Type  Item_MRP  
Outlet_Identifier  \
0              1.0               0.0        4.0  249.8092  
9.0   
1              0.0               1.0       14.0   48.2692  
3.0   
2              1.0               0.0       10.0  141.6180  
9.0   

   Outlet_Size  Outlet_Location_Type  Outlet_Type  
Item_Outlet_Sales  \
0          1.0                   0.0          1.0          3735.1380   

1          1.0                   2.0          2.0           443.4228   

2          1.0                   0.0          1.0          2097.2700   


   Item_Weight_interploate  Item_Visibility_interpolate  Outlet_age  
0                     9.30                     0.016047          26  
1                     5.92                     0.019278          16  
2                    17.50                     0.016760          26  
```

```
X = data_encoded.drop('Item_Outlet_Sales',axis=1)
y = data_encoded['Item_Outlet_Sales']
```

```
y
```

```
0       3735.1380
1        443.4228
2       2097.2700
3        732.3800
4        994.7052
          ...    
8518    2778.3834
8519     549.2850
8520    1193.1136
8521    1845.5976
8522     765.6700
Name: Item_Outlet_Sales, Length: 8523, dtype: float64
```

Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor

from sklearn.model_selection import cross_val_score

rf = RandomForestRegressor(n_estimators=100,random_state=42)
```

```
scores = cross_val_score(rf,X,y,cv=5,scoring='r2')
print(scores.mean())
```

```
0.5549938762817801
```

XGBRFRegressor

```
from xgboost import XGBRFRegressor

xg = XGBRFRegressor(n_estimators=100,random_state=42)
scores = cross_val_score(xg,X,y,cv=5,scoring='r2')
print(scores.mean())
```

```
0.5956605980255832
```

XGBRFRegressor Feature importances

```
xg = XGBRFRegressor(n_estimators=100,random_state=42)


xg1 = xg.fit(X,y)
pd.DataFrame({
    'feature':X.columns,
    'XGBRF_importance':xg1.feature_importances_

}).sort_values(by='XGBRF_importance',ascending=False)
```

```
                        feature  XGBRF_importance
7                   Outlet_Type          0.395703
4             Outlet_Identifier          0.205234
3                      Item_MRP          0.148953
10                   Outlet_age          0.138809
5                   Outlet_Size          0.077087
6          Outlet_Location_Type          0.026407
9    Item_Visibility_interpolate          0.002478
8        Item_Weight_interploate          0.002064
2                     Item_Type          0.001822
0               Item_Identifier          0.000924
1              Item_Fat_Content          0.000519
```

```
['Item_Visibility_interpolate','Item_Weight_interploate','Item_Type','Outlet_Location_Type','Item_Identifier','Item_Fat_Content']
```

```
['Item_Visibility_interpolate',
 'Item_Weight_interploate',
 'Item_Type',
 'Outlet_Location_Type',
 'Item_Identifier',
 'Item_Fat_Content']
```

```
from xgboost import XGBRFRegressor

xg = XGBRFRegressor(n_estimators=100,random_state=42)
scores =
cross_val_score(xg1,X.drop(['Item_Visibility_interpolate','Item_Weight
_interploate',
'Item_Type','Outlet_Location_Type','Item_Identifier','Item_Fat_Content
'],axis=1),y,cv=5,scoring='r2')
print(scores.mean())
```

```
0.5964911457646872
```

```
final_data =
X.drop(columns=['Item_Visibility_interpolate','Item_Weight_interploate
',
'Item_Type','Outlet_Location_Type','Item_Identifier','Item_Fat_Content
'],axis=1)

final_data
```

```
      Item_MRP  Outlet_Identifier  Outlet_Size  Outlet_Type
Outlet_age
0     249.8092                9.0          1.0          1.0
26
1      48.2692                3.0          1.0          2.0
16
2     141.6180                9.0          1.0          1.0
26
3     182.0950                0.0          2.0          0.0
27
4      53.8614                1.0          0.0          1.0
38
...        ...                ...          ...          ...         ..
.
8518  214.5218                1.0          0.0          1.0
38
8519  108.1570                7.0          2.0          1.0
23
8520   85.1224                6.0          2.0          1.0
21
8521  103.1332                3.0          1.0          2.0
16
8522   75.4670                8.0          2.0          1.0
28

[8523 rows x 5 columns]
```

Best Model

```
from xgboost import XGBRFRegressor
```

```
xg_final = XGBRFRegressor()
```

```
xg_final.fit(final_data,y)
```

```
XGBRFRegressor(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bytree=None,
device=None,
               early_stopping_rounds=None, enable_categorical=False,
               eval_metric=None, feature_types=None,
feature_weights=None,
               gamma=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=None, max_leaves=None,
               min_child_weight=None, missing=nan,
monotone_constraints=None,
               multi_strategy=None, n_estimators=None, n_jobs=None,
               num_parallel_tree=None, objective='reg:squarederror',
               random_state=None, ...)
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
```

```
X_train,X_test,y_train,y_test =
train_test_split(final_data,y,test_size=0.20,random_state=42)
```

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

```
XGBRFRegressor(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bytree=None,
device=None,
               early_stopping_rounds=None, enable_categorical=False,
               eval_metric=None, feature_types=None,
feature_weights=None,
               gamma=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=None, max_leaves=None,
               min_child_weight=None, missing=nan,
monotone_constraints=None,
               multi_strategy=None, n_estimators=None, n_jobs=None,
               num_parallel_tree=None, objective='reg:squarederror',
               random_state=None, ...)
```

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

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

```
713.1429493857894
```

Prediction on Unseen Data

```
pred = xg_final.predict(np.array([[141.6180,9.0,1.0,1.0,24]]))[0]
print(pred)
```

```
2049.021
```

```
print(f"Sales Value is between {pred-714.42} and {pred+714.42}")
```

```
Sales Value is between 1334.60107421875 and 2763.44091796875
```

Save Model Using Joblib

```
import joblib
```

```
joblib.dump(xg_final,'bigmart_model')
```

```
['bigmart_model']
```

```
model = joblib.load('bigmart_model')
```

```
pred = model.predict(np.array([[141.6180,9.0,1.0,1.0,24]]))[0]
print(pred)
```

```
2049.021
```

```
print(f"Sales Value is between {pred-714.42} and {pred+714.42}")
```

```
Sales Value is between 1334.60107421875 and 2763.44091796875
```

GUI Implementation

```
##### import numpy as np
import datetime as dt
from tkinter import *
import joblib
current_year = dt.datetime.today().year
def show_entry_fields():
    p1=float(e1.get())
    text = clicked.get()
    if text == "OUT010":
        p2=0
        print(p2)
    elif text=="OUT013":
        p2=1
        print(p2)
    elif text=="OUT017":
        p2=2
        print(p2)
    elif text=="OUT018":
        p2=3
        print(p2)
    elif text=="OUT019":
        p2=4
        print(p2)
    elif text=="OUT027":
        p2=5
        print(p2)
    elif text=="OUT035":
        p2=6
        print(p2)
    elif text=="OUT045":
        p2=7
        print(p2)
    elif text=="OUT046":
        p2=8
        print(p2)
    elif text=="OUT049":
        p2=9
        print(p2)
    text0 = clicked0.get()
    if text0 == "High":
        p3=0
        print(p3)
    elif text0=="Medium":
        p3=1
        print(p3)
    elif text0=="Small":
        p3=2
        print(p3)

    text1 = clicked1.get()
    if text1 == "Supermarket Type1":
        p4=1
        print(p4)
    elif text1=="Supermarket Type2":
        p4=2
        print(p4)
    elif text1=="Supermarket Type3":
        p4=3
        print(p4)
    elif text1=="Grocery Store":
        p4=0
        print(p4)

    p5=current_year - int(e5.get())
    print(p5)

    model = joblib.load('bigmart_model')
    result=model.predict(np.array([[p1,p2,p3,p4,p5]]))
    Label(master, text="Sales Amount is in between").grid(row=8)
    Label(master, text=float(result) -714.42 ).grid(row=10)
    Label(master, text="and").grid(row=11)
    Label(master, text=float(result) + 714.42) .grid(row=12)
    print("Sales amount", result)

master = Tk()
master.title("Big Mart Sales Prediction using Machine Learning")


label = Label(master, text = " Big Mart Sales Prediction using ML"
                          , bg = "black", fg = "white"). \
                               grid(row=0,columnspan=2)

# Item_MRP Outlet_Identifier     Outlet_Size     Outlet_Type
     Outlet_age
Label(master, text="Item_MRP").grid(row=1)
Label(master, text="Outlet_Identifier").grid(row=2)
Label(master, text="Outlet_Size").grid(row=3)
Label(master, text="Outlet_Type").grid(row=4)
Label(master, text="Outlet_Establishment_Year").grid(row=5)


clicked = StringVar()
options = ['OUT010', 'OUT013', 'OUT017', 'OUT018', 'OUT019', 'OUT027',
       'OUT035', 'OUT045', 'OUT046', 'OUT049']

clicked0 = StringVar()

options0 = ['High', 'Medium', 'Small']

clicked1 = StringVar()
options1 = ['Grocery Store', 'Supermarket Type1', 'Supermarket Type2',
       'Supermarket Type3']

e1 = Entry(master)

e2 = OptionMenu(master , clicked , *options )
e2.configure(width=15)


e3 = OptionMenu(master , clicked0 , *options0 )
e3.configure(width=15)


e4 = OptionMenu(master , clicked1 , *options1 )
e4.configure(width=15)

e5 = Entry(master)


e1.grid(row=1, column=1)
```

```
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)


Button(master, text='Predict', command=show_entry_fields).grid()
mainloop()
```

```
4
1
2
11
Sales amount [3716.1257]
```