```
#!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_train.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_S
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Med
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Med
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Med
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	N
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	H

```
#df_test
```

```
df_train.shape
```

```
(8523, 12)
```

```
df_train.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
```

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

```
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):
 #   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)
memory usage: 799.2+ KB
```

```
df_train.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
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

Item_Weight is numerical column so we fill it with Mean Imputation

```
df_train['Item_Weight'].describe()
```

```
count    7060.000000
mean       12.857645
std         4.643456
min         4.555000
25%         8.773750
50%        12.600000
75%        16.850000
max        21.350000
Name: Item_Weight, dtype: float64
```

```
df_train['Item_Weight'].fillna(df_train['Item_Weight'].mean(),inplace=True)
df_test['Item_Weight'].fillna(df_test['Item_Weight'].mean(),inplace=True)
```

```
df_train.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                  2410
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales               0
dtype: int64
```

```
df_train['Item_Weight'].describe()
```

```
count    8523.000000
mean       12.857645
std         4.226124
min         4.555000
25%         9.310000
50%        12.857645
75%        16.000000
max        21.350000
Name: Item_Weight, dtype: float64
```

Outlet_Size is catagorical column so we fill it with Mode Imputation

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

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

```
df_train['Outlet_Size'].mode()
```

```
0    Medium
dtype: object
```

```
df_train['Outlet_Size'].fillna(df_train['Outlet_Size'].mode()[0],inplace=True)
df_test['Outlet_Size'].fillna(df_test['Outlet_Size'].mode()[0],inplace=True)
```

```
df_train.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
```

```
df_test.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
dtype: int64
```

Selecting features based on general requirements

```
df_train.drop(['Item_Identifier','Outlet_Identifier'],axis=1,inplace=True)
df_test.drop(['Item_Identifier','Outlet_Identifier'],axis=1,inplace=True)
```

```
df_train
```

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	O
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

EDA with Dtale Library

```
import dtale
```

```
C:\Users\thero\Anaconda3\lib\site-packages\dtale\dash_application\charts.py:13: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
  import dash_core_components as dcc
C:\Users\thero\Anaconda3\lib\site-packages\dtale\dash_application\charts.py:14: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
```

```
dtale.show(df_train)
```

```
2021-10-18 12:57:23,097 - INFO     - NumExpr defaulting to 8 threads.
```

EDA using Pandas Profiling

```
from pandas_profiling import ProfileReport
```

```
profile = ProfileReport(df_train, title="Pandas Profiling Report")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-b38bd2f54af1> in <cell line: 1>()
```

```
profile
```

Overview

Overview Reproduction Warnings 1

Dataset statistics

Number of variables	10
Number of observations	8523
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	3.0 MiB
Average record size in memory	371.1 B

Variable types

NUM	5
CAT	5

Variables

Item_Weight

```
plt.figure(figsize=(10,5))
sns.heatmap(df_train.corr(),annot=True)
plt.show()
```

```
C:\Users\thero\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning:

Matplotlib is currently using agg, which is a non-GUI backend, so cannot show the figure.
```

EDA using Klib Library

```
import klib
```

```
# klib.describe - functions for visualizing datasets
klib.cat_plot(df_train) # returns a visualization of the number and frequency of categorical features
```

```
GridSpec(6, 5)
```

```
klib.corr_mat(df_train) # returns a color-encoded correlation matrix
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
Item_Weight	1.00	-0.01	0.02	-0.01	0.01
Item_Visibility	-0.01	1.00	-0.00	-0.07	-0.13
Item_MRP	0.02	-0.00	1.00	0.01	0.57
Outlet_Establishment_Year	-0.01	-0.07	0.01	1.00	-0.05

```
klib.corr_plot(df_train) # returns a color-encoded heatmap, ideal for correlations
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1daa3b33ec8>
```

```
klib.dist_plot(df_train) # returns a distribution plot for every numeric feature
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1daa9e7d688>
```

```
klib.missingval_plot(df_train) # returns a figure containing information about missing values
```

```
No missing values found in the dataset.
```

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_
0	9.300000	Low Fat	0.016047	Dairy	249.809204	
1	5.920000	Regular	0.019278	Soft Drinks	48.269199	
2	17.500000	Low Fat	0.016760	Meat	141.617996	
3	19.200001	Regular	0.000000	Fruits and Vegetables	182.095001	
4	8.930000	Low Fat	0.000000	Household	53.861401	
...	...	...	...	...	...	
8518	6.865000	Low Fat	0.056783	Snack Foods	214.521805	
8519	8.380000	Regular	0.046982	Baking Goods	108.156998	
8520	10.600000	Low Fat	0.035186	Health and Hygiene	85.122398	
8521	7.210000	Regular	0.145221	Snack Foods	103.133202	
8522	14.800000	Low Fat	0.044878	Soft Drinks	75.467003	

8523 rows x 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_y
0	9.300	Low Fat	0.016047	Dairy	249.8092	1
1	5.920	Regular	0.019278	Soft Drinks	48.2692	2
2	17.500	Low Fat	0.016760	Meat	141.6180	1
3	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	1
4	8.930	Low Fat	0.000000	Household	53.8614	1
...	...	...	...	...	...	
8518	6.865	Low Fat	0.056783	Snack Foods	214.5218	1
8519	8.380	Regular	0.046982	Baking Goods	108.1570	2
8520	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	2
8521	7.210	Regular	0.145221	Snack Foods	103.1332	2
8522	14.800	Low Fat	0.044878	Soft Drinks	75.4670	1

```
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_
0	9.300000	Low Fat	0.016047	Dairy	249.809204	
1	5.920000	Regular	0.019278	Soft Drinks	48.269199	
2	17.500000	Low Fat	0.016760	Meat	141.617996	
3	19.200001	Regular	0.000000	Fruits and Vegetables	182.095001	
4	8.930000	Low Fat	0.000000	Household	53.861401	
...	...	...	...	...	...	
8518	6.865000	Low Fat	0.056783	Snack Foods	214.521805	
8519	8.380000	Regular	0.046982	Baking Goods	108.156998	

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_
0	9.300000	1	0.016047	4	249.809204	
1	5.920000	2	0.019278	14	48.269199	
2	17.500000	1	0.016760	10	141.617996	
3	19.200001	2	0.000000	6	182.095001	
4	8.930000	1	0.000000	9	53.861401	
...	...	...	...	...	...	
8518	6.865000	1	0.056783	13	214.521805	
8519	8.380000	2	0.046982	0	108.156998	
8520	10.600000	1	0.035186	8	85.122398	
8521	7.210000	2	0.145221	13	103.133202	
8522	14.800000	1	0.044878	14	75.467003	

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
5695    1436.796387
8006    2167.844727
1361    2700.484863
1547     829.586792
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
3891    1358.232056
6629    2418.185547
Name: item_outlet_sales, Length: 1705, dtype: float32
```

```
import joblib
```

```

```