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

```
df = pd.read_csv('bigmart_train.csv')
```

```
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_M
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.80
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.26
2	FDN15	17.50	Low Fat	0.016760	Meat	141.61
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.09
4	NCD19	8.93	Low Fat	0.000000	Household	53.86

```
  #### Pre-Processing for categorical Variables
```

```
## 1. Checking for null values
df.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.isnull().sum()/len(df)*100
```

```
     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
```

```
## Drop the variable outlet_size

df_new=df.drop('Outlet_Size', axis =1)
```

```
df_new.isnull().sum()/len(df_new)*100
```

```
     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_Location_Type          0.000000
     Outlet_Type                   0.000000
     Item_Outlet_Sales             0.000000
     dtype: float64
```

```
df.dtypes
```

```
Item_Identifier               object
Item_Weight                  float64
Item_Fat_Content              object
Item_Visibility              float64
Item_Type                     object
Item_MRP                     float64
Outlet_Identifier             object
Outlet_Establishment_Year      int64
Outlet_Size                   object
Outlet_Location_Type          object
Outlet_Type                   object
Item_Outlet_Sales            float64
dtype: object
```

```
df_new.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Out
count	7060.000000	8523.000000	8523.000000	8523.000000	85
mean	12.857645	0.066132	140.992782	1997.831867	21
std	4.643456	0.051598	62.275067	8.371760	17
min	4.555000	0.000000	31.290000	1985.000000	
25%	8.773750	0.026989	93.826500	1987.000000	8
50%	12.600000	0.053931	143.012800	1999.000000	17
75%	16.850000	0.094585	185.643700	2004.000000	31
max	21.350000	0.328391	266.888400	2009.000000	130

```
df_new["Item_Weight"]=df_new["Item_Weight"].fillna(12.65)
```

```
df_new.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_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
dtype: int64
```

```
df_new['Item_Fat_Content'].unique() ## Checking the unique values for categorical variable
```

```
array(['Low Fat', 'Regular', 'low fat', 'LF', 'reg'], dtype=object)
```

```
### Need to replace all the repititive values to unique values i.e. Low Fat, Regular

df_new["Item_Fat_Content"] = df_new["Item_Fat_Content"].str.replace("low fat", "Low Fat")
```

```
df_new["Item_Fat_Content"] = df_new["Item_Fat_Content"].str.replace("LF", "Low Fat")
```

```
df_new["Item_Fat_Content"] = df_new["Item_Fat_Content"].str.replace("reg", "Regular")
```

```
df_new['Item_Fat_Content'].unique()
```

```
array(['Low Fat', 'Regular'], dtype=object)
```

```
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_M
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2

```
df_new['Item_Type'].unique()
```

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

```
df_new['Outlet_Identifier'].unique()
```

```
array(['OUT049', 'OUT018', 'OUT010', 'OUT013', 'OUT027', 'OUT045',
       'OUT017', 'OUT046', 'OUT035', 'OUT019'], dtype=object)
```

```
df_new['Outlet_Establishment_Year'].nunique()
```

```
9
```

Now We have to seperate the categorical variables and numerical variables

```
df_new.dtypes
```

```
Item_Identifier               object
Item_Weight                  float64
Item_Fat_Content              object
Item_Visibility              float64
Item_Type                     object
Item_MRP                     float64
Outlet_Identifier             object
Outlet_Establishment_Year      int64
Outlet_Location_Type          object
Outlet_Type                   object
Item_Outlet_Sales            float64
dtype: object
```

change the datatypes for Outlet_establishment_Year as it content only 9 unique values

```
df_new["Outlet_Establishment_Year"]=df_new["Outlet_Establishment_Year"].astype("category")
```

```
df_new.dtypes
```

```
Item_Identifier                object
Item_Weight                   float64
Item_Fat_Content               object
Item_Visibility               float64
Item_Type                      object
Item_MRP                      float64
Outlet_Identifier              object
Outlet_Establishment_Year    category
Outlet_Location_Type           object
Outlet_Type                    object
Item_Outlet_Sales             float64
dtype: object
```

```
df_num=df_new.select_dtypes(include=["int64", "float64"])
```

```
df_cat=df_new.select_dtypes(include=["object","category"])
```

```
df_num.head()
```

```
df_cat.head()
```

	Item_Identifier	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Establishm
0	FDA15	Low Fat	Dairy	OUT049	
1	DRC01	Regular	Soft Drinks	OUT018	
2	FDN15	Low Fat	Meat	OUT049	

```
### Dropping the dependent variables.
```

```
df_num=df_num.drop('Item_Outlet_Sales', axis=1)
```

```
df_num.head()
```

	Item_Weight	Item_Visibility	Item_MRP
0	9.30	0.016047	249.8092
1	5.92	0.019278	48.2692
2	17.50	0.016760	141.6180
3	19.20	0.000000	182.0950
4	8.93	0.000000	53.8614

```
df_cat.head()
```

	Item_Identifier	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Establishm
0	FDA15	Low Fat	Dairy	OUT049	
1	DRC01	Regular	Soft Drinks	OUT018	
2	FDN15	Low Fat	Meat	OUT049	

```
df_num.describe()
```

	Item_Weight	Item_Visibility	Item_MRP
count	8523.000000	8523.000000	8523.000000
mean	12.822002	0.066132	140.992782
std	4.226849	0.051598	62.275067
min	4.555000	0.000000	31.290000
25%	9.310000	0.026989	93.826500
50%	12.650000	0.053931	143.012800
75%	16.000000	0.094585	185.643700
max	21.350000	0.328391	266.888400

```
## no need to do transformation as data is normally distributed
```

```
## Applying the scaling techniques
```

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

```
mn=MinMaxScaler()
```

```
df_sc=mn.fit_transform(df_num)
```

```
df_sc_df=pd.DataFrame(df_sc, columns=df_num.columns, index=df_num.index)
```

```
df_sc_df.head()
```

	Item_Weight	Item_Visibility	Item_MRP
0	0.282525	0.048866	0.927507
1	0.081274	0.058705	0.072068
2	0.770765	0.051037	0.468288
3	0.871986	0.000000	0.640093
4	0.260494	0.000000	0.095805

```
## Preprocessing for categorical data
## Applying One-Hot Encoding

df_cat_dum = pd.get_dummies(df_cat, drop_first=True)
```

```
df_cat_dum.head()
```

	Item_Identifier_DRA24	Item_Identifier_DRA59	Item_Identifier_DRB01	Item_Identif
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	

5 rows × 1596 columns

```
## Combining the both numerical and categorical data after preprocessing

df_final = pd.concat([df_sc_df,df_cat_dum], axis=1)
```

```
df_final.head()
```

	Item_Weight	Item_Visibility	Item_MRP	Item_Identifier_DRA24	Item_Identifier_DR/
0	0.282525	0.048866	0.927507	0	
1	0.081274	0.058705	0.072068	0	
2	0.770765	0.051037	0.468288	0	
3	0.871986	0.000000	0.640093	0	
4	0.260494	0.000000	0.095805	0	

5 rows × 1599 columns

```
X=df_final
y=df_new["Item_Outlet_Sales"]
```

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

```
X_train, X_test,  y_train, y_train = train_test_split(X,y, test_size=0.2, random_state=1)
```

```
from sklearn.linear_model import LinearRegression
```

```
y_train, X_train
```

```
(1070     952.7598
 6305    1133.8574
 8504    4138.6128
 5562    1657.1762
 1410     679.1160
           ...
 376     5715.2272
 7708    4832.3764
 3812    2972.1312
 3928    2492.7552
 7654    1717.7640
 Name: Item_Outlet_Sales, Length: 1705, dtype: float64,
       Item_Weight  ...  Outlet_Type_Supermarket Type3
 1945     0.821375  ...                             0
 1720     0.761834  ...                               0
 1954     0.330158  ...                               0
 1919     0.374814  ...                               0
 2461     0.155701  ...                               0
 ...           ...  ...                             ...
 2895     0.481989  ...                               0
 7813     0.481989  ...                               0
 905      0.791605  ...                               0
 5192     0.300387  ...                               0
 235      0.481989  ...                               1

 [6818 rows x 1599 columns])
```

```
lr=LinearRegression()
```

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-54-4311d9257fbf> in <module>()
----> 1 lr.fit(X_train, y_train)
```

3 frames

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py in
check_consistent_length(*arrays)
    332         raise ValueError(
    333             "Found input variables with inconsistent numbers of samples: %r"
--> 334             % [int(l) for l in lengths]
    335         )
    336 

ValueError: Found input variables with inconsistent numbers of samples: [6818, 1705]
```

SEARCH STACK OVERFLOW