In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
# import the sklearn libraries for label encoder and linear regrssion
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
```

In [3]:
```
#import matplot for visulization
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:
```
#let's upload train and test data
train_data=pd.read_csv(r"C:\Users\s323\Desktop\Data Science\ML\Pratical Demo 3.2\b
test_data=pd.read_csv(r"C:\Users\s323\Desktop\Data Science\ML\Pratical Demo 3.2\bi
```

In [5]:
```
# Fist five datasets from train data
train_data.head()
```

Out[5]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	

In [6]:
```
# First five datasets from test data
test_data.head()
```

Out[6]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	

In [7]:
```
# check the no of rows and columns in the data
train_data.shape
```

Out[7]:
```
(8523, 12)
```

In [8]:
```
# Check the shape of test data
test_data.shape
```

Out[8]:
```
(5681, 11)
```

In [9]:
```
# Print the name of columns of train dataset
train_data.columns
```

Out[9]:
```
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')
```

In [10]:
```
# Print the name of columns of test dataset
test_data.columns
```

Out[10]:
```
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'],
      dtype='object')
```

In [11]:
```
# Combine both test and train data to perform EDA
train_data["source"]="train"
test_data["source"]="test"
data=pd.concat([train_data,test_data], ignore_index= True)
```

In [12]:
```
data.shape
```

Out[12]:
```
(14204, 13)
```

In [13]:
```
data.head()
```

Out[13]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	

In [14]:
```
data.columns
```

Out[14]:
```
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', 'source'],
      dtype='object')
```

In [15]:
```
# Describe function will show numerical data summary
data.describe()
```

Out[15]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	11765.000000	14204.000000	14204.000000	14204.000000	8523.000000
mean	12.792854	0.065953	141.004977	1997.830681	2181.288914
std	4.652502	0.051459	62.086938	8.371664	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.710000	0.027036	94.012000	1987.000000	834.247400
50%	12.600000	0.054021	142.247000	1999.000000	1794.331000
75%	16.750000	0.094037	185.855600	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

In [16]:
```
# checking missing values
data.isnull().sum()
```

Out[16]:
```
Item_Identifier               0
Item_Weight                2439
Item_Fat_Content               0
Item_Visibility                0
Item_Type                      0
Item_MRP                       0
Outlet_Identifier              0
Outlet_Establishment_Year      0
Outlet_Size                4016
Outlet_Location_Type           0
Outlet_Type                    0
Item_Outlet_Sales           5681
source                         0
dtype: int64
```

We have to predict these above missing values through the model

In [17]:
```
# We have to  find unique values,
# We will find in unique value in item_fat_content as it has only two unique value

data["Item_Fat_Content"].unique()
```

Out[17]:
```
array(['Low Fat', 'Regular', 'low fat', 'LF', 'reg'], dtype=object)
```

In [18]:
```
# Print the unique values in Outlet_Establishment_Year coulumn where data ranges fr
data["Outlet_Establishment_Year"].unique()
```

Out[18]:
```
array([1999, 2009, 1998, 1987, 1985, 2002, 2007, 1997, 2004], dtype=int64)
```

In [19]:
```
# Calculate the outlet age
data["Outlet_Age"]= 2022-data["Outlet_Establishment_Year"]
data.head(2)
```

Out[19]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	

In [20]:
```
#Unique values in outletsize
data["Outlet_Size"].unique()
```

Out[20]:
```
array(['Medium', nan, 'High', 'Small'], dtype=object)
```

There are missing values in this columnm

In [21]:
```
# Printing the value of Item fat content column
data["Item_Fat_Content"].count()
```

Out[21]:
```
14204
```

In [22]:
```
# Printing the count value of Item_fat content column
data["Item_Fat_Content"].value_counts()
```

Out[22]:
```
Low Fat    8485
Regular    4824
LF          522
reg         195
low fat     178
Name: Item_Fat_Content, dtype: int64
```

We can see Low Fat Content are most abundant

In [23]:
```
# Print the count value of outlet_size
data["Outlet_Size"].value_counts()
```

Out[23]:
```
Medium    4655
Small     3980
High      1553
Name: Outlet_Size, dtype: int64
```

We can see that majority of outlets are in medium and small size

In [24]:
```
# Use the mode function to find out the most common value in outlet_size
data["Outlet_Size"].mode()[0]
```

Out[24]:
```
'Medium'
```

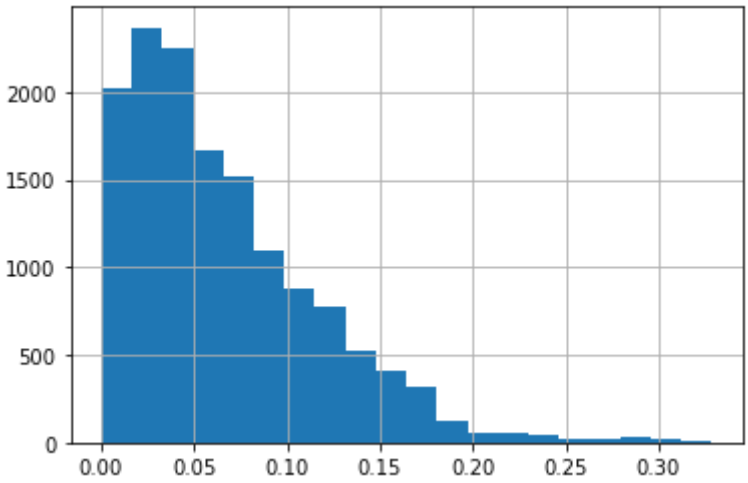
The output shows medium is the most coomonly occuring value

In [25]:
```
# Two variables Item_wieght and Outlet_Size have missing values
# Replacing missing value in outlet size with medium size
data["Outlet_Size"]=data["Outlet_Size"].fillna(data['Outlet_Size'].mode()[0])
```

In [26]:
```
# Replace missing item_wieght with mean weight
data["Item_Weight"]=data["Item_Weight"].fillna(data["Item_Weight"].mean())
```

In [27]:
```
#Plot a histogram to reveal the distribution of item_visiblity column
data["Item_Visibility"].hist(bins=20)
```

Out[27]: <AxesSubplot:>



In [28]:
```
# Detecting Outliers
# An outlier is a data point that lies outside the overall pattern in a distributi
# Acommonly used rule that states a data point is an outlier if it is 1.5*IQR above
# Using this one can remove outliers and output results in fill_data variable
# Calculating the first quantile for item visiblity

Q1= data["Item_Visibility"].quantile(0.25)
```

In [29]:
```
# Calculate the second quantile for item visiblity

Q3= data["Item_Visibility"].quantile(0.75)
```

In [30]:
```
# Calculate the IQR- Inter Quantile Range

IQR=Q3-Q1
```

In [31]:
```
#Now since the IQR range is known, lets remove the outliers
#The resulting data is stored in fill data varible

fill_data= data.query("(@Q1 - 1.5* @IQR)<=Item_Visibility<=(@Q3 +1.5* @IQR)")
```

In [32]:
```
# Display the data
fill_data.head(2)
```

Out[32]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	

In [33]:
```
# Check the shape of resulting dataset without the outliers
fill_data.shape
```

Out[33]: (13943, 14)

In [34]:
```
# Check the shape of original dataset with the outliers
data.shape
```

Out[34]: (14204, 14)

In [35]:
```
# Assign the fill data set to the data_frame
data=fill_data
```

In [36]:
```
data.shape
```

Out[36]:
```
(13943, 14)
```

In [47]:
```
# Modify item visiblity by converting the numerical value into cateogries as low V
data["Item_Visibility.bins"]=pd.cut(data["Item_Visibility"],[0.000,0.065,0.13,0.2]
```

In [48]:
```
# Total no of counts of item_visiblity

data["Item_Visibility.bins"].value_counts()
```

Out[48]:
```
Low_viz     7363
Med_viz     4283
High_viz    1418
Name: Item_Visibility.bins, dtype: int64
```

In [51]:
```
# Replace nan values with low_viz
data["Item_Visibility.bins"].isnull().sum()
```

Out[51]:
```
879
```

In [52]:
```
data["Item_Visibility.bins"]= data["Item_Visibility.bins"].replace(np.nan,"Low_viz
```

In [54]:
```
# We have found typos and diffrences in representation in cateogris of Item_fat va
# This can be corrected below

# Replace all other representation of low fat with low fat
data["Item_Fat_Content"]= data["Item_Fat_Content"].replace (["low fat", "LF"],"Low
```

In [55]:
```
# replace all reg with Regular
data["Item_Fat_Content"]= data["Item_Fat_Content"].replace (["reg"],"Regular")
```

In [56]:
```
# Print unique fat counts
data["Item_Fat_Content"].unique()
```

Out[56]:
```
array(['Low Fat', 'Regular'], dtype=object)
```

In [57]:
```
# Code all catogorical variable as numeric using "Label Encoder" from SK learn's p
# Intialize label encoder

le= LabelEncoder()
```

In [58]:
```
# Transform Item_Fat_Content
data["Item_Fat_Content"]=le.fit_transform(data["Item_Fat_Content"])
```

In [59]:
```
# Transform Item_Visiblity_bins
data["Item_Visibility.bins"]=le.fit_transform(data["Item_Visibility.bins"])
```

In [60]:
```
# Transform outletsize
data["Outlet_Size"]=le.fit_transform(data["Outlet_Size"])
```

In [61]:
```
# Transform outlet location type
data["Outlet_Location_Type"]=le.fit_transform(data["Outlet_Location_Type"])
```

In [62]:
```
# Print the unique values of Outlet type
data["Outlet_Type"].unique()
```

Out[62]:
```
array(['Supermarket Type1', 'Supermarket Type2', 'Grocery Store',
       'Supermarket Type3'], dtype=object)
```

In [65]:
```
# Create dummies for outlet type
dummies = pd.get_dummies(data["Outlet_Type"])
dummies.head()
```

Out[65]:

	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3
0	0	1	0	0
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0
4	0	1	0	0

In [66]:
```
# Explore the column_Item_identifier
data["Item_Identifier"]
```

Out[66]:
```
0        FDA15
1        DRC01
2        FDN15
3        FDX07
4        NCD19
         ...
14199    FDB58
14200    FDD47
14201    NCO17
14202    FDJ26
14203    FDU37
Name: Item_Identifier, Length: 13943, dtype: object
```

In [67]:
```
# As we can see there are multiple values for food,non consumable items,and drinks
data["Item_Identifier"].value_counts()
```

Out[67]:
```
FDE33    10
FDM12    10
FDY47    10
FDT03    10
FDO01    10
         ..
FDA10     7
FDO33     7
FDZ60     7
NCW54     7
FDG21     7
Name: Item_Identifier, Length: 1559, dtype: int64
```

In [69]:
```
# As multiple item cateogries are present in item_identifier, reduce this by mappi
data["Item_type_Combined"]=data["Item_Identifier"].apply(lambda x: x[0:2])
data["Item_type_Combined"]=data["Item_type_Combined"].map({"FD":"Food","NC":"Non-C
```

In [70]:
```
# Only three catogries are present in the item_type_combined column
data["Item_type_Combined"].value_counts()
```

Out[70]:
```
Food              9991
Non-Consumable    2652
Drinks            1300
Name: Item_type_Combined, dtype: int64
```

In [71]:
```
data.shape
```

Out[71]:
```
(13943, 16)
```

In [72]:
```
# Perform one hot encoding on all columns as model works on numerical value not on

data=pd.get_dummies(data, columns=["Item_Fat_Content","Outlet_Location_Type","Outl
```

In [77]:
```
data.dtypes
# Now we can see some of the columns are cateogrical as they are mentioned as objec
```

Out[77]:
```
Item_Identifier                  object
Item_Weight                     float64
Item_Visibility                 float64
Item_Type                        object
Item_MRP                        float64
Outlet_Identifier                object
Outlet_Establishment_Year         int64
Item_Outlet_Sales               float64
source                           object
Outlet_Age                        int64
Item_Visibility.bins              int32
Item_Fat_Content_0                uint8
Item_Fat_Content_1                uint8
Outlet_Location_Type_0            uint8
Outlet_Location_Type_1            uint8
Outlet_Location_Type_2            uint8
Outlet_Size_0                     uint8
Outlet_Size_1                     uint8
Outlet_Size_2                     uint8
Outlet_Type_Grocery Store         uint8
Outlet_Type_Supermarket Type1     uint8
Outlet_Type_Supermarket Type2     uint8
Outlet_Type_Supermarket Type3     uint8
Item_type_Combined_Drinks         uint8
Item_type_Combined_Food           uint8
Item_type_Combined_Non-Consumable uint8
dtype: object
```

In [82]:
```
import warnings
warnings.filterwarnings('ignore')

# Drop the columns which have been converted into diffrent types

data.drop(["Item_Type","Outlet_Establishment_Year"],  axis=1,inplace=True)

# Divide the dataset created earlier into train and test datasets

train = data.loc[data["source"] == "train"]
test = data.loc[data["source"] == "test"]

# Drop unnecessay columns

test.drop(["Item_Outlet_Sales", "source"], axis=1, inplace= True)
train.drop(["source"], axis= 1, inplace=True)

#Export modified versions of the files
```

```
train.to_csv("train_modified.csv", index=False)
test.to_csv("test_modified.csv", index= False)
```

In [83]:
```
# Read the train_modified.csv and test_modified.csv

train2= pd.read_csv(r"train_modified.csv")
test2= pd.read_csv(r"test_modified.csv")
```

In [84]:
```
# Print the data types of train 2 columns
train2.dtypes
```

Out[84]:
```
Item_Identifier                    object
Item_Weight                       float64
Item_Visibility                   float64
Item_MRP                          float64
Outlet_Identifier                  object
Item_Outlet_Sales                 float64
Outlet_Age                          int64
Item_Visibility.bins                int64
Item_Fat_Content_0                  int64
Item_Fat_Content_1                  int64
Outlet_Location_Type_0              int64
Outlet_Location_Type_1              int64
Outlet_Location_Type_2              int64
Outlet_Size_0                       int64
Outlet_Size_1                       int64
Outlet_Size_2                       int64
Outlet_Type_Grocery Store           int64
Outlet_Type_Supermarket Type1       int64
Outlet_Type_Supermarket Type2       int64
Outlet_Type_Supermarket Type3       int64
Item_type_Combined_Drinks           int64
Item_type_Combined_Food             int64
Item_type_Combined_Non-Consumable   int64
dtype: object
```

In [85]:
```
# Drop the irrelevant variables from train2 datasets
# Create the indepedent varible X_train and depedent variable Y_train

X_train=train2.drop(["Item_Outlet_Sales","Outlet_Identifier","Item_Identifier"], a
y_train=train2.Item_Outlet_Sales
```

In [86]:
```
# Drop those irelevant variables from test2 dataset

X_test=test2.drop(["Outlet_Identifier","Item_Identifier"],axis=1)
```

In [87]:
```
X_test
```

Out[87]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Age	Item_Visibility.bins	Item_Fat_Content_0
0	20.750000	0.007565	107.8622	23	1	1
1	8.300000	0.038428	87.3198	15	1	0
2	14.600000	0.099575	241.7538	24	2	1
3	7.315000	0.015388	155.0340	15	1	1
4	12.792854	0.118599	234.2300	37	2	0
...	...	...	...	...	...	...
5563	10.500000	0.013496	141.3154	25	1	0
5564	7.600000	0.142991	169.1448	13	0	0
5565	10.000000	0.073529	118.7440	20	2	1
5566	15.300000	0.000000	214.6218	15	3	0
5567	9.500000	0.104720	79.7960	20	2	0

5568 rows × 20 columns

In [89]:
```
X_train.head(2)
```

Out[89]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Age	Item_Visibility.bins	Item_Fat_Content_0	It
0	9.30	0.016047	249.8092	23	1	1	
1	5.92	0.019278	48.2692	13	1	0	

In [90]:
```
y_train.head(2)
```

Out[90]:
```
0    3735.1380
1     443.4228
Name: Item_Outlet_Sales, dtype: float64
```

In [97]:
```
# import sklearn libraries for model selection
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
```

In [98]:
```
# create a train and test split
xtrain, xtest, ytrain, ytest = model_selection.train_test_split(X_train,y_train,te
```

In [100…
```
# fit linear regression to training dataset
lin=LinearRegression()
lin.fit(xtrain,ytrain)
```

Out[100]:
```
LinearRegression()
```

In [101…
```
# find the coefficent and intercept of the line
# use X train and Y train for linear regression
print(lin.coef_)
lin.intercept_
```

```
[-1.93054423e+00 -3.47449893e+02  1.58788136e+01 -3.19310190e+01
 -4.64698319e+00 -1.55426303e+00  1.55426303e+00  1.88969149e+02
  4.83874237e+01 -2.37356573e+02  5.39972452e+02 -3.11545503e+02
 -2.28426949e+02 -1.63452944e+03 -1.26820412e+02 -3.48655405e+02
  2.11000526e+03  9.06454503e+00  3.46403369e+01 -4.37048820e+01]
1074.1452929564798
```

Out[101]:

In [102…

```
# predict the test set results for training data
predictions=lin.predict(xtest)
predictions
```

Out[102]:

```
array([2077.92386202, 3704.93216108, 2961.28180591, ..., 3541.28725867,
       3464.78656382, 1249.59368407])
```

In [105…

```
import math
```

In [106…

```
# Find the rmse of the model
print(math.sqrt(mean_squared_error(ytest,predictions)))
```

```
1126.2352692434756
```

In [107…

```
# A good RMSE for this problem is 1130. Here we can improve the RMSE by using algor
# Next, we will predict the sales of each product at a particular store in test dat
```

In [109…

```
# predict the column of Item_outlet_sales of test data
y_sales_pred=lin.predict(X_test)
y_sales_pred
```

Out[109]:

```
array([1788.81075495, 1593.90875193, 1845.79649796, ..., 1831.65421972,
       3605.8573969 , 1284.78769015])
```

In [110…

```
test_predictions=pd.DataFrame({
    "Item_Identifier":test2["Item_Identifier"],
    "Outlet_Identifier": test2["Outlet_Identifier"],
    "Item_Outlet_Sales":y_sales_pred
},columns=["Item_Identifier","Outlet_Identifier","Item_Outlet_Sales"])
```

In [111…

```
test_predictions
```

Out[111]:

	Item_Identifier	Outlet_Identifier	Item_Outlet_Sales
0	FDW58	OUT049	1788.810755
1	FDW14	OUT017	1593.908752
2	NCN55	OUT010	1845.796498
3	FDQ58	OUT017	2675.927968
4	FDY38	OUT027	5134.091429
...	...	...	...
5563	FDB58	OUT046	2360.100059
5564	FDD47	OUT018	2419.143609
5565	NCO17	OUT045	1831.654220
5566	FDJ26	OUT017	3605.857397
5567	FDU37	OUT045	1284.787690

5568 rows × 3 columns

In [ ]: