import numpy as np  
impport pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.preprocessing import LabelEncoder  
from sklearn.model\_selection import train\_test\_split  
from xgboost import XGBRegressor  
from sklearn import metrics  
  
Data collection and analysis  
  
#loading dataset from csv file to a pandas Dataframe  
big\_mart\_data = pd.read\_csv()  
  
#first five rows  
big\_mart\_data.head()  
  
#number of data points and number of features  
big\_mart\_data.shape  
  
#getting some information about dataset  
[big\_mart\_data.info](http://big_mart_data.info/)()  
  
Categorical features:  
  
Item\_Identifier  
Item\_Fat\_Content  
Item\_Type  
Outlet\_Identifier  
Outlet\_Size  
Outlet\_Location\_Type  
Outlet\_Type  
  
#checking for missing values  
big\_mart\_data.isnull().sum()  
  
Handling missing values through Mean and Mode  
#mean of "Item\_weight" column  
big\_mart\_data['Item\_weight'].mean()  
  
#filling the missing values in "Item\_weight" column with "Mean" value  
big\_mart\_data['Item\_weight'].fillna(big\_mart\_data['Item\_weight'].mean(), inplace = True)  
  
#checking for missing values  
big\_mart\_data.isnull().sum()  
  
#Replacing the missing values in "Outlet\_size" with mode  
mode\_of\_outlet\_size = big\_mart\_data.pivot\_table(values= 'Outlet\_size', columns = 'Outlet\_Type', aggfunc(lambda x: x.mode()[0]))  
print(mode\_of\_outlet\_size)  
missing\_values = big\_mart\_data['Outlet\_size'].isnull()  
print(missing\_values)  
big\_mart\_data.loc[missing\_values, 'Outlet\_Size'] = big\_mart\_data.loc[missing\_values, 'Outlet\_Type'].apply(lambda x: mode\_of\_outlet\_size)  
  
#checking for missing values  
big\_mart\_data.isnull().sum()  
  
Data Analysis  
  
#statistical measures about the data  
big\_mart\_data.describe()  
  
Numerical features  
sns.set()  
  
#Item\_weight\_distribution  
plt.figure(figsize=(6,6))  
sns.distplot(big\_mart\_data['Item\_weight'])  
plt.show()  
  
#Item\_Visibility\_distribution  
plt.figure(figsize=(6,6))  
sns.distplot(big\_mart\_data['Item\_Visibility'])  
plt.show()  
  
#Item\_Outlet\_sales\_distribution  
plt.figure(figsize=(6,6))  
sns.distplot(big\_mart\_data['Item\_Outlet\_Sales'])  
plt.show()  
  
#Outlet\_Establishment\_Year column  
ply.figure(figsize=(6,6))  
sns.countplot(x= 'Outlet\_Establishment\_Year', data=big\_mart\_data)  
plt.show()  
  
#Item\_Fat\_Content\_column  
plt.figure(figsize=(6,6))  
sns.countplot(x= 'Item\_Fat\_Content', data=big\_mart\_data)  
plt.show()  
  
#Item\_Type\_column  
plt.figure(figsize=(30,6))  
sns.countplot(x= 'Item\_Type', data=big\_mart\_data)  
plt.show()  
  
#Outlet\_Size\_column  
plt.figure(figsize=(30,6))  
sns.countplot(x='Outlet\_Size', data=big\_mart\_data)  
plt.show()  
  
Data pre-processing  
big\_mart\_data.head()  
  
big\_mart\_data['Item\_Fat\_Content'].value\_counts()  
  
big\_mart\_data.replace({'Item\_Fat\_Content':{'low fat':'Low Fat', 'LF:Low Fat', 'reg':'Regular'}}, inplace=True)  
  
big\_mart\_data['Item\_Fat\_Content'].value\_counts()  
  
Label Encoding  
  
encoder = LabelEncoder()  
  
big\_mart\_data['Item\_Identifier'] = encoder.fit\_transform(big\_mart\_data['Item\_Identifier'])  
big\_mart\_data['Item\_Fat\_Content'] = encoder.fit\_transform(big\_mart\_data['Item\_Fat\_Content'])  
big\_mart\_data['Item\_Type'] = encoder.fit\_transform(big\_mart\_data['Item\_Type'])  
big\_mart\_data['Outlet\_Identifier'] = encoder.fit\_transform(big\_mart\_data['Outlet\_Identifier'])  
big\_mart\_data['Outlet\_Size'] = encoder.fit\_transform(big\_mart\_data['Outlet\_Size'])  
big\_mart\_data['Outlet\_Location\_Type'] = encoder.fit\_transform(big\_mart\_data['Outlet\_Location\_Type'])  
big\_mart\_data['Outlet\_Type'] = encoder.fit\_transform(big\_mart\_data['Outlet\_Type'])  
  
big\_mart\_data.head()  
  
Splitting feautures and Target  
  
X = big\_mart\_data.drop(columns='Item\_Outlet\_Sales', axis=1)  
Y = big\_mart\_data['Item\_Outlet\_Sales']  
  
print(X)  
print(Y)  
  
Splitting the data into Training and Testing data  
  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=2)  
  
print(X.shape, X\_train.shape, X\_test.shape)  
  
Machine learning model training  
  
XGBoost Regressor  
regressor = XGBRegressor()  
regressor.fit(X\_train, Y\_train)  
  
#prediction on training data  
training\_data\_prediction = regressor.predict(X\_train)  
  
#R squared value  
r2\_train = metrics.r2\_score(Y\_train, training\_data\_prediction)  
  
print('R squared value = ',r2\_train)  
  
#prediction on test data  
test\_data\_prediction = regressor.predict(X\_test)  
  
#R squared value  
r2\_test = metrics.r2\_score(Y\_test, test\_data\_prediction)  
  
print('R squared value = ', r2\_test)