In [1]:

```
import math
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [2]:

```
train = pd.read_csv("./Train.csv")
test = pd.read_csv("./Test.csv")
```

In [3]:

```
train.head()
```

Out[3]:

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

In [4]:

```
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
```

In [5]:

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

Out[5]:

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

In [6]:

```
# Correcting mislabeleld columns

train['Item_Fat_Content'].replace(to_replace='low fat', value='Low Fat', inplace=True)
train['Item_Fat_Content'].replace(to_replace='LF', value='Low Fat', inplace=True)
train['Item_Fat_Content'].replace(to_replace='reg', value='Regular', inplace=True)

test['Item_Fat_Content'].replace(to_replace='low fat', value='Low Fat', inplace=True)
test['Item_Fat_Content'].replace(to_replace='LF', value='Low Fat', inplace=True)
test['Item_Fat_Content'].replace(to_replace='reg', value='Regular', inplace=True)
```

In [7]:

```
# Factorising categorical columns in the dataset

col_enc = ['Item_Identifier', 'Item_Fat_Content', 'Item_Type', 'Outlet_Identifier', 'Out
let_Establishment_Year', 'Outlet_Location_Type', 'Outlet_Type']

for x in col_enc:
    train[x], _ = pd.factorize(train[x])
    test[x], _ = pd.factorize(test[x])
```

In [8]:

```
test.isnull().sum()
```

Out[8]:

```
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
```

In [9]:

```
# Handling the missing values

# Use regression to fill missing values in the 'Item_Weight' column.

# Train set
train_sub = train.drop(['Outlet_Size'], axis = 1)
train_sub_test = train_sub[train_sub["Item_Weight"].isnull()]

train_sub = train_sub.dropna()

y_train = train_sub["Item_Weight"]
X_train = train_sub.drop("Item_Weight", axis=1)

X_test = train_sub_test.drop("Item_Weight", axis=1)

lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

```
train.loc[train.Item_Weight.isnull(), 'Item_Weight'] = y_pred
```

In [10]:

```
# Test set
test_sub = test.drop(['Outlet_Size'], axis = 1)
test_sub_test = test_sub[test_sub["Item_Weight"].isnull()]

test_sub = test_sub.dropna()

y_test = test_sub["Item_Weight"]
X_test = test_sub.drop("Item_Weight", axis=1)

X_test_test = test_sub_test.drop("Item_Weight", axis=1)

lr = LinearRegression()
lr.fit(X_test, y_test)
y_pred = lr.predict(X_test_test)

test.loc[test.Item_Weight.isnull(), 'Item_Weight'] = y_pred
```

In [11]:

```
# Filling in 'Outlet_Size' column using mode replacement.

train['Outlet_Size'].fillna(train['Outlet_Size'].mode()[0], inplace=True)
test['Outlet_Size'].fillna(test['Outlet_Size'].mode()[0], inplace=True)

train['Outlet_Size'], _ = pd.factorize(train['Outlet_Size'])
test['Outlet_Size'], _ = pd.factorize(test['Outlet_Size'])
```

In [12]:

```
# Preparing training and test sets

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=4
2)
```

In [13]:

```
# Linear Regression

lr = LinearRegression()
lr.fit(X_train, y_train)
predictions = lr.predict(X_test)

print('Mean squared error: ', mean_squared_error(y_test, predictions))
print('Root mean squared error: ', math.sqrt(mean_squared_error(y_test, predictions)))
print('Mean absolute error: ', mean_absolute_error(y_test, predictions))
print('Coefficient of determination (R2): ', r2_score(y_test, predictions))
```

```
Mean squared error:  1593302.966016391
Root mean squared error:  1262.261053037917
Mean absolute error:  928.8977207526835
Coefficient of determination (R2):  0.4315167309048753
```

In [14]:

```
# Gradient Boosting

reg = GradientBoostingRegressor(random_state = 42)
reg.fit(X_train, y_train)
predictions = reg.predict(X_test)

print('Mean squared error: ', mean_squared_error(y_test, predictions))
print('Root mean squared error: ', math.sqrt(mean_squared_error(y_test, predictions)))
print('Mean absolute error: ', mean_absolute_error(y_test, predictions))
print('Coefficient of determination (R2): ', r2_score(y_test, predictions))
```

```
Mean squared error:  1135809.2466410724
Root mean squared error:  1065.7435182261595
Mean absolute error:  753.1713728419547
Coefficient of determination (R2):  0.5947484142244764
```

In [107]:

```
# Extreme Gradient Boosting

xgb = XGBRegressor()
xgb.fit(X_train, y_train)
predictions = xgb.predict(X_test)

print('Mean squared error: ', mean_squared_error(y_test, predictions))
print('Root mean squared error: ', math.sqrt(mean_squared_error(y_test, predictions)))
print('Mean absolute error: ', mean_absolute_error(y_test, predictions))
print('Coefficient of determination (R2): ', r2_score(y_test, predictions))
```

```
Mean squared error:  1328878.6941220842
Root mean squared error:  1152.7700092048215
Mean absolute error:  807.3120965056385
Coefficient of determination (R2):  0.5258621113634385
```

In [15]:

```
# Random Forest

rf = RandomForestRegressor(max_depth = 2, random_state = 42)
rf.fit(X_train, y_train)
predictions = rf.predict(X_test)

print('Mean squared error: ', mean_squared_error(y_test, predictions))
print('Root mean squared error: ', math.sqrt(mean_squared_error(y_test, predictions)))
print('Mean absolute error: ', mean_absolute_error(y_test, predictions))
print('Coefficient of determination (R2): ', r2_score(y_test, predictions))
```

```
Mean squared error:  1701378.0748748793
Root mean squared error:  1304.3688415762158
Mean absolute error:  986.4445918560846
Coefficient of determination (R2):  0.3929560224256238
```

In [16]:

```
# Decision Tree

dt = DecisionTreeRegressor(random_state = 42)
dt.fit(X_train, y_train)
predictions = dt.predict(X_test)

print('Mean squared error: ', mean_squared_error(y_test, predictions))
print('Root mean squared error: ', math.sqrt(mean_squared_error(y_test, predictions)))
print('Mean absolute error: ', mean_absolute_error(y_test, predictions))
print('Coefficient of determination (R2): ', r2_score(y_test, predictions))
```

```
Mean squared error:  2433598.059110748
Root mean squared error:  1559.9993779199876
Mean absolute error:  1082.4324565232846
Coefficient of determination (R2):  0.13170325429959928
```

In [17]:

```
# K Nearest Neighbors

knn = KNeighborsRegressor(n_neighbors = 2)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)

print('Mean squared error: ', mean_squared_error(y_test, predictions))
print('Root mean squared error: ', math.sqrt(mean_squared_error(y_test, predictions)))
print('Mean absolute error: ', mean_absolute_error(y_test, predictions))
print('Coefficient of determination (R2): ', r2_score(y_test, predictions))
```

```
Mean squared error:  3064472.267630474
Root mean squared error:  1750.5634143413583
Mean absolute error:  1277.088068787771
Coefficient of determination (R2):  -0.09338980087984106
```

In [18]:

```
# Support Vector Machine

rng = np.random.RandomState(42)
regr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
regr.fit(X_train, y_train)
predictions = regr.predict(X_test)

print('Mean squared error: ', mean_squared_error(y_test, predictions))
print('Root mean squared error: ', math.sqrt(mean_squared_error(y_test, predictions)))
print('Mean absolute error: ', mean_absolute_error(y_test, predictions))
print('Coefficient of determination (R2): ', r2_score(y_test, predictions))
```

```
Mean squared error:  2720662.385723626
Root mean squared error:  1649.4430531920846
Mean absolute error:  1239.0536801696262
Coefficient of determination (R2):  0.02928000504055006
```

In [ ]:

```
# Gradient Boosting Regressor gives the best performance with the
# lease RMSE of 1065.74.
```