LP1

Assignment C3

Data Analysis

Title :- Bigmart Sales Analysis Date of Completion:- 20/11/20

Problem Statement :- For data comprising of transaction records of sales store. The data has 8523 rows of 12 variables. predict the sales of a store.

~~Of~~ Learning objectives :- 1) Learn regression algorithm
2) Summarize properties of dataset
3) Learn to split the dataset.

Learning Outcomes :- Students will be able to develop a predictive model for sales of an item at Bigmart.

~~Learning~~ Software/Hardware requirement :- OS (Linux), python ~~libra~~ libraries.

Theory :-

Linear Regression :-

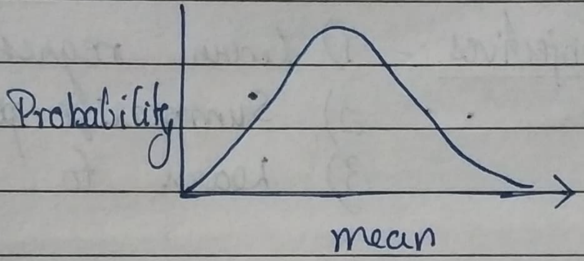
1) It is a ~~model to~~ linear approach to model the relationship between a scalar response & one or more explanatory variables. The case of one explanatory variable is called simple linear regression.

$$y = B_0 + B_1 * x$$

input x , output y
The line is a plane or hyperplane.

2) Gaussian Distribution:-

1) It is a symmetric distribution where most of the observations cluster ~~&~~ around the central ~~peak~~ peak of the probability for values further away the mean taper off equally in both direction



Dataset:-

2013 Sales data for 1559 products across 10 stores in different cities.
Certain attributes of each product & store have been defined. The aim is to build a predictive model.

Attributes :-

Item_Identifier
Item_weight
Item_fat_Content
Item_Visibility
Item_Type
Item_MRP

Outlet_Identifier
Outlet_Establishment_Ye
Outlet_Size
Outlet_Location_Type
Outlet_Type
ItemOutlet_Sales.

Test Cases:-

Case	Expected O/P	Actual O/P	Remark
~~Mean~~ K Neighbour ~~for~~ . Regression	MSE:- 2720662 Root MSE:- 1649 MAE:-1239 R2:- 0.029	MSE:-2720662 Root MSE:- 1649 MAE:-1239 R2 :- 0.029	Passed
DecisionTree	RMSE:- 1546 MAE:- 1076 R2:- 0.146	RMSE:-1546 MAE:-1076 R2:- 0.146	Passed
Linear regression	RMSE:- 1065 MAE:- 752 R2 :- 0.59	RMSE :- 1065 MAE :- 752 R2 :- 0.59	Passed

MAE → mean absolute error R2 → Coefficient of determination.

Conclusion:-

Thus I have completed bigmart Sales Analysis. ~~and concluded that~~

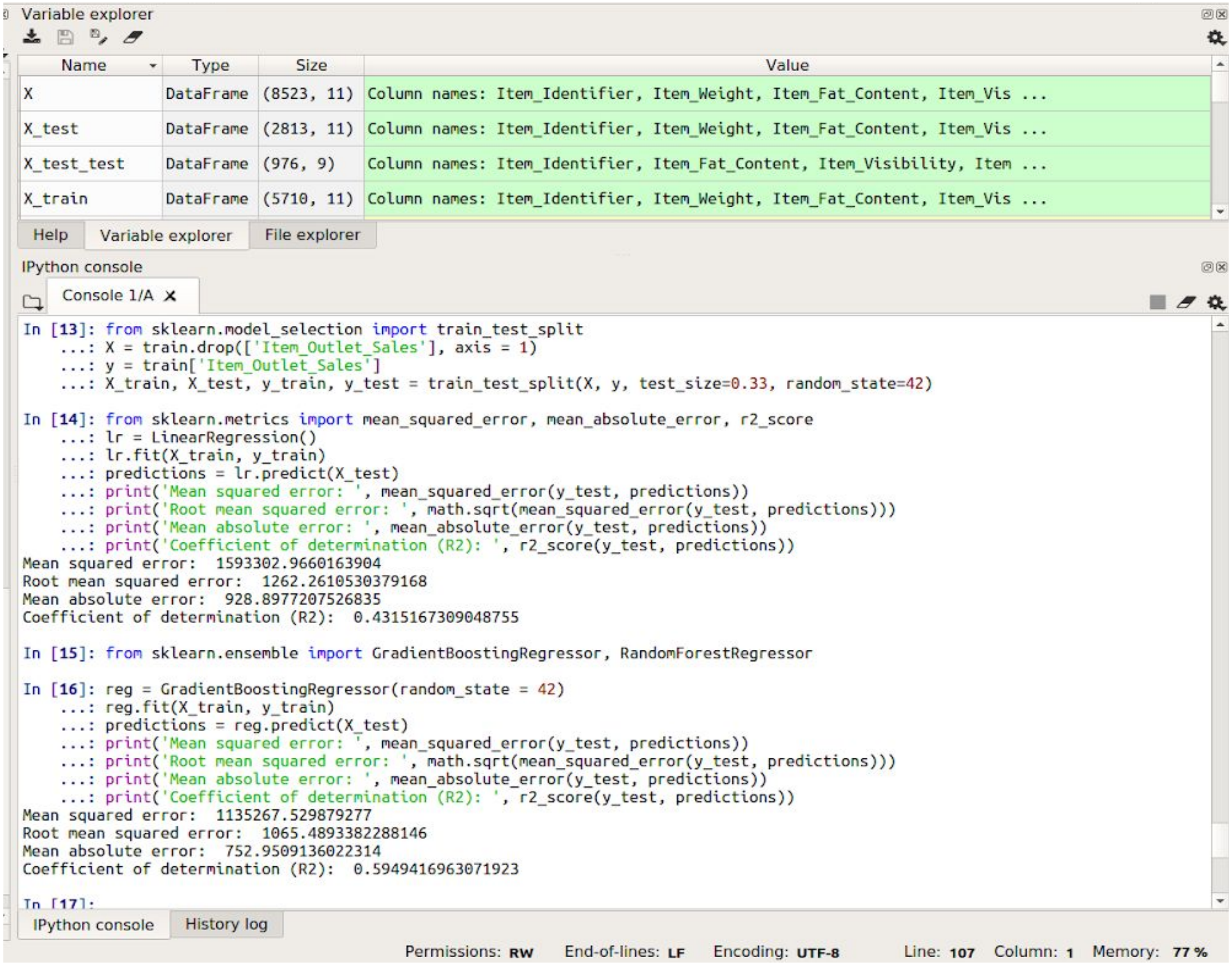
CODE

```
import math
import pandas as pd
import numpy as np

from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt

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

print(train.head())
print(test.head())

print(train.info())
print(test.info())

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

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 )

col_enc = ['Item_Identifier', 'Item_Fat_Content', 'Item_Type', 'Outlet_Identifier',
'Outlet_Establishment_Year', 'Outlet_Location_Type', 'Outlet_Type']
for x in col_enc:
    train[x], _ = pd.factorize(train[x])
    test[x], _ = pd.factorize(test[x])

test.isnull().sum()

from sklearn.linear_model import LinearRegression
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


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

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'])

from sklearn.model_selection import train_test_split
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=42)

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
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))

from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
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

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))



from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from xgboost import XGBRegressor

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))
```

```
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
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))

# 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))

# 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))

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))
```

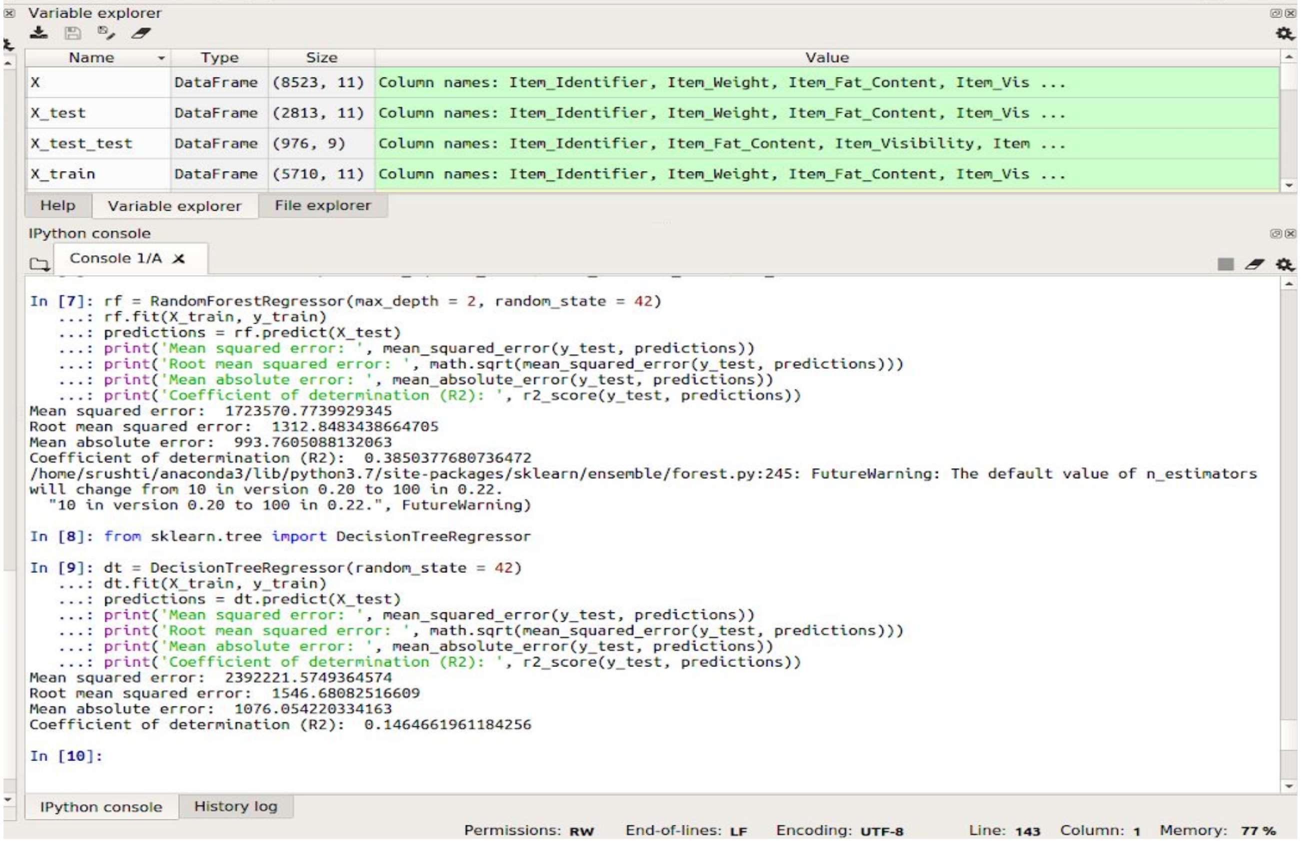
OUTPUT



Name	Type	Size	Value
X	DataFrame	(8523, 11)	Column names: Item_Identifier, Item_Weight, Item_Fat_Content, Item_Vis ...
X_test	DataFrame	(2813, 11)	Column names: Item_Identifier, Item_Weight, Item_Fat_Content, Item_Vis ...
X_test_test	DataFrame	(976, 9)	Column names: Item_Identifier, Item_Fat_Content, Item_Visibility, Item ...
X_train	DataFrame	(5710, 11)	Column names: Item_Identifier, Item_Weight, Item_Fat_Content, Item_Vis ...

```
In [13]: from sklearn.model_selection import train_test_split
    ...: 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=42)

In [14]: from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
    ...: 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:  1593302.9660163904
Root mean squared error:  1262.2610530379168
Mean absolute error:  928.8977207526835
Coefficient of determination (R2):  0.4315167309048755

In [15]: from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor

In [16]: 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:  1135267.529879277
Root mean squared error:  1065.4893382288146
Mean absolute error:  752.9509136022314
Coefficient of determination (R2):  0.5949416963071923

In [17]:
```

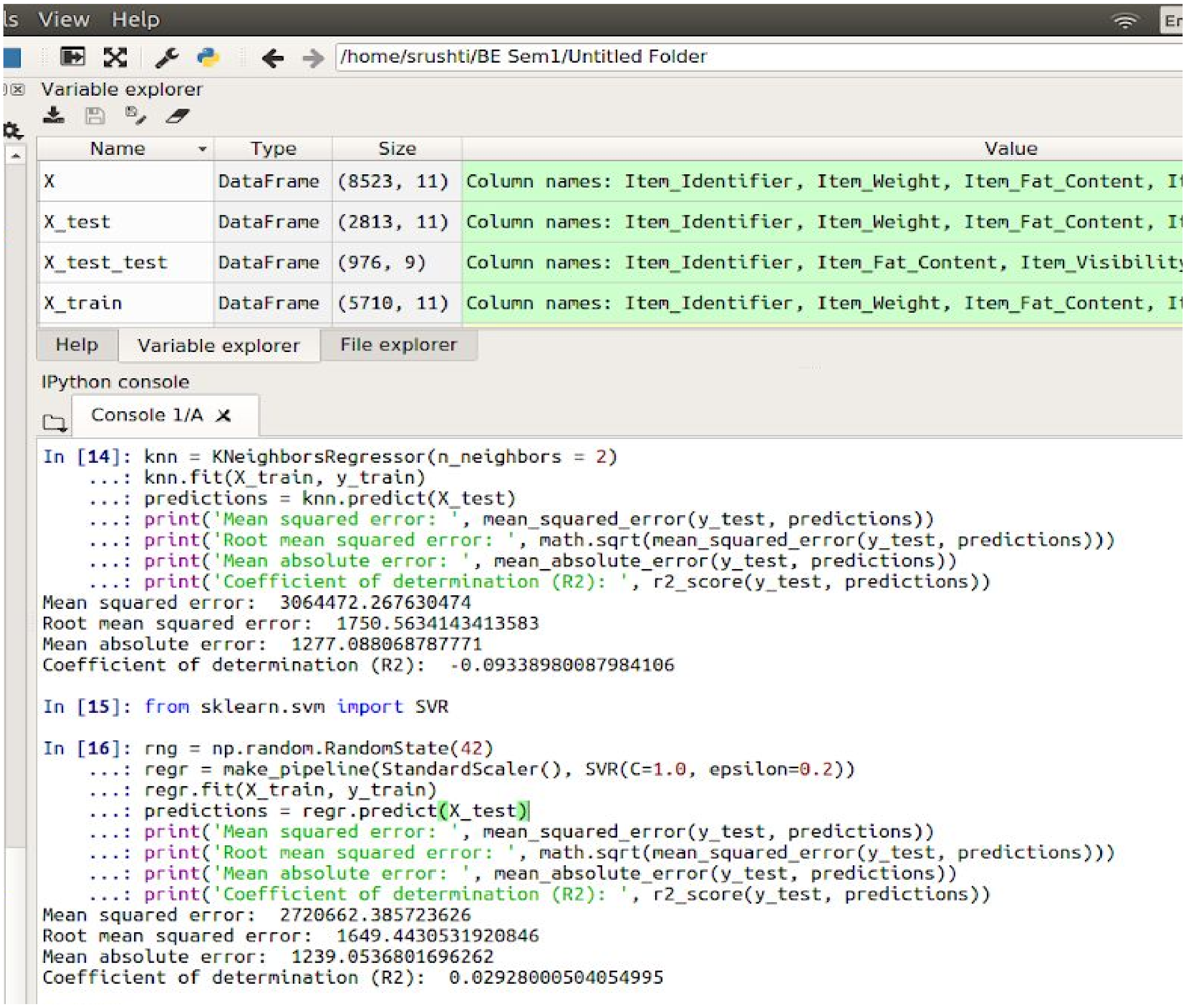
Permissions: RW   End-of-lines: LF   Encoding: UTF-8   Line: 107   Column: 1   Memory: 77 %



Name	Type	Size	Value
X	DataFrame	(8523, 11)	Column names: Item_Identifier, Item_Weight, Item_Fat_Content, Item_Vis ...
X_test	DataFrame	(2813, 11)	Column names: Item_Identifier, Item_Weight, Item_Fat_Content, Item_Vis ...
X_test_test	DataFrame	(976, 9)	Column names: Item_Identifier, Item_Fat_Content, Item_Visibility, Item ...
X_train	DataFrame	(5710, 11)	Column names: Item_Identifier, Item_Weight, Item_Fat_Content, Item_Vis ...

```
In [7]: 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:  1723570.7739929345
Root mean squared error:  1312.8483438664705
Mean absolute error:  993.7605088132063
Coefficient of determination (R2):  0.3850377680736472
/home/srushti/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n_estimators
will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

In [8]: from sklearn.tree import DecisionTreeRegressor

In [9]: 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:  2392221.5749364574
Root mean squared error:  1546.68082516609
Mean absolute error:  1076.054220334163
Coefficient of determination (R2):  0.1464661961184256

In [10]:
```

Permissions: RW   End-of-lines: LF   Encoding: UTF-8   Line: 143   Column: 1   Memory: 77 %



/home/srushti/BE Sem1/Untitled Folder

Name	Type	Size	Value
X	DataFrame	(8523, 11)	Column names: Item_Identifier, Item_Weight, Item_Fat_Content, It
X_test	DataFrame	(2813, 11)	Column names: Item_Identifier, Item_Weight, Item_Fat_Content, It
X_test_test	DataFrame	(976, 9)	Column names: Item_Identifier, Item_Fat_Content, Item_Visibility
X_train	DataFrame	(5710, 11)	Column names: Item_Identifier, Item_Weight, Item_Fat_Content, It

```
In [14]: 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 [15]: from sklearn.svm import SVR

In [16]: 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):  0.02928000504054995
```