In [35]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [36]:
```
df = pd.read_csv('Final.csv')
df.head()
```

Out[36]:

	Item_Identifier	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Item_Outlet_Sales	Outl
0	FDA15	9.30	0.922960	Dairy	249.8092	OUT049	1999	3735.1380	
1	DRC01	5.92	1.003057	Soft Drinks	48.2692	OUT018	2009	443.4228	
2	FDN15	17.50	0.831990	Meat	141.6180	OUT049	1999	2097.2700	
3	FDX07	19.20	0.750000	Fruits and Vegetables	182.0950	OUT010	1998	732.3800	
4	NCD19	8.93	0.666667	Household	53.8614	OUT013	1987	994.7052	

5 rows × 30 columns

In [37]:
```
remove_cols = [
    'Item_Identifier',
    'Item_Type',
    'Outlet_Identifier',
    'Outlet_Establishment_Year'
]
df = df.drop(remove_cols,axis =1)
df.head()
```

Out[37]:

	Item_Weight	Item_Visibility	Item_MRP	Item_Outlet_Sales	Outlet_Years	Item_Fat_Content_1	Item_Fat_Content_2	Outlet_Location_Type_
0	9.30	0.922960	249.8092	3735.1380	14	False	False	Fals
1	5.92	1.003057	48.2692	443.4228	4	False	True	Fals
2	17.50	0.831990	141.6180	2097.2700	14	False	False	Fals
3	19.20	0.750000	182.0950	732.3800	15	False	True	Fals
4	8.93	0.666667	53.8614	994.7052	26	True	False	Fals

5 rows × 26 columns

In [38]:
```
df.shape
```

Out[38]:
```
(8519, 26)
```

In [39]:
```
y = df.Item_Outlet_Sales.values
X = df.drop('Item_Outlet_Sales',axis = 1)
```

In [40]:
```
print(X.shape,y.shape)
```

```
(8519, 25) (8519,)
```

In [41]:
```
X.head()
```

Out[41]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Years	Item_Fat_Content_1	Item_Fat_Content_2	Outlet_Location_Type_1	Outlet_Location_
0	9.30	0.922960	249.8092	14	False	False	False	
1	5.92	1.003057	48.2692	4	False	True	False	
2	17.50	0.831990	141.6180	14	False	False	False	
3	19.20	0.750000	182.0950	15	False	True	False	
4	8.93	0.666667	53.8614	26	True	False	False	

5 rows × 25 columns

In [42]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [43]:
```
cols = [
    'Item_Weight',
    'Item_Visibility',
    'Item_MRP',
    'Outlet_Years'
]
X[cols]
```

Out[43]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Years
0	9.300	0.922960	249.8092	14
1	5.920	1.003057	48.2692	4
2	17.500	0.831990	141.6180	14
3	19.200	0.750000	182.0950	15
4	8.930	0.666667	53.8614	26
...	...	...	...	...
8514	6.865	0.920247	214.5218	26
8515	8.380	1.000657	108.1570	11
8516	10.600	0.999512	85.1224	9
8517	7.210	1.031393	103.1332	4
8518	14.800	0.870321	75.4670	16

8519 rows × 4 columns

In [44]:
```
X[cols]  =  sc.fit_transform(X[cols])
```

In [45]:
```
X.head()
```

Out[45]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Years	Item_Fat_Content_1	Item_Fat_Content_2	Outlet_Location_Type_1	Outlet_Location_
0	-0.769598	-0.391478	1.746938	-0.138865	False	False	False	
1	-1.497133	0.015532	-1.489096	-1.333806	False	True	False	
2	0.995427	-0.853739	0.009762	-0.138865	False	False	False	
3	1.361347	-1.270366	0.659682	-0.019371	False	True	False	
4	-0.849240	-1.693822	-1.399305	1.295064	True	False	False	

5 rows × 25 columns

In [46]:
```
params={
 "learning_rate"    : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30 ] ,
 "max_depth"        : [ 3, 4, 5, 6, 8, 10, 12, 15],
 "min_child_weight" : [ 1, 3, 5, 7 ],
 "gamma"            : [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
 "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]

}
```

In [47]:
```
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
import xgboost
```

In [48]:
```
model = xgboost.XGBRegressor()
```

In [49]:
```
def timer(start_time=None):
    if not start_time:
        start_time = datetime.now()
        return start_time
    elif start_time:
        thour, temp_sec = divmod((datetime.now() - start_time).total_seconds(), 3600)
        tmin, tsec = divmod(temp_sec, 60)
        print('\n Time taken: %i hours %i minutes and %s seconds.' % (thour, tmin, round(tsec, 2)))
```

In [50]:
```
from sklearn.metrics import mean_squared_error,make_scorer
random_search=RandomizedSearchCV(model,param_distributions=params,n_iter=5,scoring='neg_mean_squared_error',n_jobs=-1,cv=5,ver
```

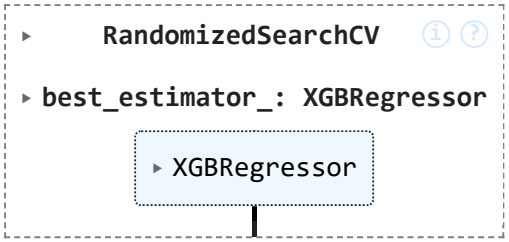
In [51]:
```
from datetime import datetime
start_time = timer(None)
random_search.fit(X,y)
timer(start_time)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits

 Time taken: 0 hours 0 minutes and 10.22 seconds.
```

In [52]:
```
random_search
```

Out[52]:

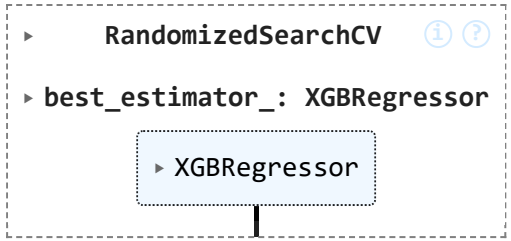


In [53]:
```
means = random_search.cv_results_['mean_test_score']
params = random_search.cv_results_['params']
for mean, param in zip(means, params):
    print("%f with: %r" % (mean, param))
    if mean == min(means):
        print('Best parameters with the minimum Mean Square Error are:',param)
```

```
-1400439.459290 with: {'min_child_weight': 3, 'max_depth': 10, 'learning_rate': 0.15, 'gamma': 0.0, 'colsample_bytree': 0.5}
-1200778.533061 with: {'min_child_weight': 5, 'max_depth': 3, 'learning_rate': 0.3, 'gamma': 0.0, 'colsample_bytree': 0.3}
-1492711.659390 with: {'min_child_weight': 3, 'max_depth': 15, 'learning_rate': 0.15, 'gamma': 0.4, 'colsample_bytree': 0.5}
-1503880.540561 with: {'min_child_weight': 1, 'max_depth': 10, 'learning_rate': 0.25, 'gamma': 0.0, 'colsample_bytree': 0.5}
Best parameters with the minimum Mean Square Error are: {'min_child_weight': 1, 'max_depth': 10, 'learning_rate': 0.25, 'gamm
a': 0.0, 'colsample_bytree': 0.5}
-1211915.592490 with: {'min_child_weight': 5, 'max_depth': 3, 'learning_rate': 0.05, 'gamma': 0.4, 'colsample_bytree': 0.4}
```

In [54]:
```
random_search
```

Out[54]:



In [55]:
```
random_search.best_params_
```

Out[55]:
```
{'min_child_weight': 5,
 'max_depth': 3,
 'learning_rate': 0.3,
 'gamma': 0.0,
 'colsample_bytree': 0.3}
```

In [56]:
```
model = xgboost.XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0.6, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.4, max_delta_step=0, max_depth=15,
             min_child_weight=1, monotone_constraints='()',
             n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

In [57]:
```
model.fit(X,y)
```

Out[57]:

XGBRegressor

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             device=None, early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, feature_types=None, gamma=0.6, gpu_id=-1,
             grow_policy=None, importance_type='gain',
             interaction_constraints='', learning_rate=0.4, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=0,
             max_depth=15, max_leaves=None, min_child_weight=1, missing=nan,
             monotone_constraints='()', multi_strategy=None, n_estimators=100,
```

In [58]:
```
y_pred = model.predict(X)
```

In [59]:
```
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
score = r2_score(y,y_pred)
print("Score of Training:",score)
print("MAE : %.4g" % np.sqrt(mean_absolute_error(y,y_pred)))
print("RMSE : %.4g" % np.sqrt(mean_squared_error(y,y_pred)))
```

```
Score of Training: 0.9999443021821429
MAE : 2.659
RMSE : 12.74
```

In [60]:
```
from sklearn.metrics import mean_squared_error,make_scorer
from sklearn.model_selection import cross_val_score
cv_score = cross_val_score(model,X, y, cv=20, scoring='neg_mean_squared_error')
cv_score = np.sqrt(np.abs(cv_score))
print("\nModel Report")
print("RMSE : %.4g" % np.sqrt(mean_squared_error(y,y_pred)))
print("CV Score : Mean - %.4g | Std - %.4g | Min - %.4g | Max - %.4g" % (np.mean(cv_score),np.std(cv_score),np.min(cv_score),n
```

```
Model Report
RMSE : 12.74
CV Score : Mean - 1251 | Std - 69.53 | Min - 1113 | Max - 1382
```

In [61]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y ,test_size = 0.2,random_state = 2 )
```

In [62]:
```
print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)
```

```
(6815, 25) (1704, 25) (6815,) (1704,)
```

In [63]:
```
y_pred = model.predict(X_train)
```

In [64]:
```
from sklearn.metrics import r2_score,mean_squared_error
score = r2_score(y_train,y_pred)
print("Score of Training:",score)
print("RMSE : %.4g" % np.sqrt(mean_squared_error(y_train,y_pred)))
```

```
Score of Training: 0.9999459502391606
RMSE : 12.56
```

In [65]:
```
from sklearn.metrics import mean_squared_error,make_scorer
from sklearn.model_selection import cross_val_score
cv_score = cross_val_score(model,X_train, y_train, cv=20, scoring = make_scorer(mean_squared_error))
cv_score = np.sqrt(np.abs(cv_score))
print("\nModel Report")
print("RMSE : %.4g" % np.sqrt(mean_squared_error(y_train,y_pred)))
print("CV Score : Mean - %.4g | Std - %.4g | Min - %.4g | Max - %.4g" % (np.mean(cv_score),np.std(cv_score),np.min(cv_score),n
```

```
Model Report
RMSE : 12.56
CV Score : Mean - 1264 | Std - 55.41 | Min - 1176 | Max - 1371
```

In [66]:
```
y_test_pred = model.predict(X_test)
```

In [67]:
```
score = r2_score(y_test,y_test_pred)
print("Score of Testing:",score)
print("RMSE : %.4g" % np.sqrt(mean_squared_error(y_test,y_test_pred)))
```

```
Score of Testing: 0.999937543123367
RMSE : 13.42
```

In [68]:
```
cv_score = cross_val_score(model,X_test, y_test, cv=20, scoring = make_scorer(mean_squared_error))
cv_score = np.sqrt(np.abs(cv_score))
print("\nModel Report")
```

```
print("RMSE : %.4g" % np.sqrt(mean_squared_error(y_test,y_test_pred)))
print("CV Score : Mean - %.4g | Std - %.4g | Min - %.4g | Max - %.4g" % (np.mean(cv_score),np.std(cv_score),np.min(cv_score),n
```

```
Model Report
RMSE : 13.42
CV Score : Mean - 1235 | Std - 125.1 | Min - 1017 | Max - 1488
```