Big Mart Sales Dataset

Objective: To find out the properties of a product, and store which impacts the sales of a product.

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5
```

```
1 train = pd.read_csv("train.csv")
2 test = pd.read_csv("test.csv")
```

```
1 #Combine test and train into one file
2 train['source']='train'
3 test['source']='test'
4 data = pd.concat([train, test],ignore_index=True)
5 print(train.shape, test.shape, data.shape)
```

```
(8523, 13) (5681, 12) (14204, 13)
```

```
1 data.head()
```

	Item_Fat_Content	Item_Identifier	Item_MRP	Item_Outlet_Sales	Item_Type
0	Low Fat	FDA15	249.8092	3735.1380	Dairy
1	Regular	DRC01	48.2692	443.4228	Soft Drinks
2	Low Fat	FDN15	141.6180	2097.2700	Meat
3	Regular	FDX07	182.0950	732.3800	Fruits and Vegetables
4	Low Fat	NCD19	53.8614	994.7052	Household

```
1 #Numerical data summary:
2 data.describe()
```

	Item_MRP	Item_Outlet_Sales	Item_Visibility	Item_Weight	Outlet_Est
count	14204.000000	8523.000000	14204.000000	11765.000000	
mean	141.004977	2181.288914	0.065953	12.792854	
std	62.086938	1706.499616	0.051459	4.652502	
min	31.290000	33.290000	0.000000	4.555000	
25%	94.012000	834.247400	0.027036	8.710000	
50%	142.247000	1794.331000	0.054021	12.600000	
75%	185.855600	3101.296400	0.094037	16.750000	

Data Cleaning

```
1 #Check missing values:
2 data.apply(lambda x: sum(x.isnull()))
```

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

Filling missing values

```
1 data.Item_Outlet_Sales = data.Item_Outlet_Sales.fillna(data.Item_Outlet_Sales.me
```

```
1 data.Item_Weight = data.Item_Weight.fillna(data.Item_Weight.mean())
```

```
1 data['Outlet_Size'].value_counts()
```

```
Medium    4655
Small     3980
High      1553
Name: Outlet_Size, dtype: int64
```

```
1 data.Outlet_Size = data.Outlet_Size.fillna('Medium')
```

```
1 data.apply(lambda x: sum(x.isnull()))
```

```
Item_Fat_Content             0
Item_Identifier              0
Item_MRP                     0
Item_Outlet_Sales            0
Item_Type                    0
Item_Visibility              0
Item_Weight                  0
Outlet_Establishment_Year    0
Outlet_Identifier            0
Outlet_Location_Type         0
Outlet_Size                  0
Outlet_Type                  0
source                       0
dtype: int64
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14204 entries, 0 to 14203
Data columns (total 13 columns):
Item_Fat_Content             14204 non-null object
Item_Identifier              14204 non-null object
Item_MRP                     14204 non-null float64
Item_Outlet_Sales            14204 non-null float64
Item_Type                    14204 non-null object
Item_Visibility              14204 non-null float64
Item_Weight                  14204 non-null float64
Outlet_Establishment_Year    14204 non-null int64
Outlet_Identifier            14204 non-null object
Outlet_Location_Type         14204 non-null object
Outlet_Size                  14204 non-null object
Outlet_Type                  14204 non-null object
source                       14204 non-null object
dtypes: float64(4), int64(1), object(8)
memory usage: 1.4+ MB
```

```
#Item type combine:
data['Item_Identifier'].value_counts()
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-Consumab
                                                             'DR':'Drinks'})
data['Item_Type_Combined'].value_counts()
```

```
Food              10201
Non-Consumable     2686
Drinks             1317
Name: Item_Type_Combined, dtype: int64
```

Numerical and One-Hot Coding of Categorical variables

```
#Import library:
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
```

```
#New variable for outlet
data['Outlet'] = le.fit_transform(data['Outlet_Identifier'])
var_mod = ['Item_Fat_Content','Outlet_Location_Type','Outlet_Size','Item_Type_C
le = LabelEncoder()
for i in var_mod:
    data[i] = le.fit_transform(data[i])
```

```
#One Hot Coding:
data = pd.get_dummies(data, columns=['Item_Fat_Content','Outlet_Location_Type',
```

```
data.head()
```

	Item_Identifier	Item_MRP	Item_Outlet_Sales	Item_Type	Item_Visibility	I
0	FDA15	249.8092	3735.1380	Dairy	0.016047	
1	DRC01	48.2692	443.4228	Soft Drinks	0.019278	
2	FDN15	141.6180	2097.2700	Meat	0.016760	
3	FDX07	182.0950	732.3800	Fruits and Vegetables	0.000000	
4	NCD19	53.8614	994.7052	Household	0.000000	

5 rows × 37 columns

```
data.dtypes
```

```
Item_Identifier              object
Item_MRP                    float64
Item_Outlet_Sales           float64
Item_Type                    object
Item_Visibility             float64
Item_Weight                 float64
Outlet_Establishment_Year     int64
Outlet_Identifier            object
source                       object
Item_Fat_Content_0             uint8
Item_Fat_Content_1             uint8
Item_Fat_Content_2             uint8
Item_Fat_Content_3             uint8
Item_Fat_Content_4             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_0                  uint8
Outlet_Type_1                  uint8
Outlet_Type_2                  uint8
Outlet_Type_3                  uint8
Item_Type_Combined_0           uint8
Item_Type_Combined_1           uint8
Item_Type_Combined_2           uint8
```

```
Outlet_0                    uint8
Outlet_1                    uint8
Outlet_2                    uint8
Outlet_3                    uint8
Outlet_4                    uint8
Outlet_5                    uint8
Outlet_6                    uint8
Outlet_7                    uint8
Outlet_8                    uint8
Outlet_9                    uint8
dtype: object
```

Exporting Data

```
import warnings
warnings.filterwarnings('ignore')
#Drop the columns which have been converted to different types:
data.drop(['Item_Type','Outlet_Establishment_Year'],axis=1,inplace=True)

#Divide into test and train:
train = data.loc[data['source']=="train"]
test = data.loc[data['source']=="test"]

#Drop unnecessary columns:
test.drop(['Item_Outlet_Sales','source'],axis=1,inplace=True)
train.drop(['source'],axis=1,inplace=True)

#Export files as modified versions:
train.to_csv("train_modified.csv",index=False)
test.to_csv("test_modified.csv",index=False)
```

Model Building

```
# Reading modified data
train2 = pd.read_csv("train_modified.csv")
test2 = pd.read_csv("test_modified.csv")
```

```
train2.head()
```

	Item_Identifier	Item_MRP	Item_Outlet_Sales	Item_Visibility	Item_Weight

```
X_train = train2.drop(['Item_Outlet_Sales', 'Outlet_Identifier','Item_Identifie
y_train = train2.Item_Outlet_Sales
```

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

```
X_train.head()
```

	Item_MRP	Item_Visibility	Item_Weight	Item_Fat_Content_0	Item_Fat_Conten
0	249.8092	0.016047	9.30	0	
1	48.2692	0.019278	5.92	0	
2	141.6180	0.016760	17.50	0	
3	182.0950	0.000000	19.20	0	
4	53.8614	0.000000	8.93	0	

5 rows × 31 columns

```
y_train.head()
```

```
0    3735.1380
1     443.4228
2    2097.2700
3     732.3800
4     994.7052
Name: Item_Outlet_Sales, dtype: float64
```

Linear Regression Model:

```
# Fitting Multiple Linear Regression to the training set
from sklearn.linear_model import  LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
# Predicting the test set results
y_pred = regressor.predict(X_test)
```

```
y_pred
```

```
array([1848.53604783, 1472.81670435, 1875.65285894, ..., 1809.18796433,
       3565.6645235 , 1267.46171871])
```

```
import warnings
warnings.filterwarnings('ignore')
# Measuring Accuracy
from sklearn.metrics import accuracy_score, r2_score, mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn import cross_validation, metrics

```

```
lr_accuracy = round(regressor.score(X_train,y_train) * 100,2)
lr_accuracy
```

```
56.36
```

```
r2_score(y_train, regressor.predict(X_train))
```

```
0.563589277727048
```

```
import warnings
warnings.filterwarnings('ignore')
#Perform cross-validation:
cv_score = cross_val_score(regressor, X_train, y_train, cv=5, scoring='mean_squa

```

```
print(np.sqrt(np.abs(cv_score)))
```

```
[1150.93927648 1118.68414103 1112.89657923 1126.30724065 1140.59735737]
```

```
print("RMSE : %.4g" % np.sqrt(metrics.mean_squared_error(y_train, regressor.pred
```

```
RMSE : 1127
```

```
submission = pd.DataFrame({
'Item_Identifier':test2['Item_Identifier'],
'Outlet_Identifier':test2['Outlet_Identifier'],
'Item_Outlet_Sales': y_pred
},columns=['Item_Identifier','Outlet_Identifier','Item_Outlet_Sales'])
```

```
submission.to_csv('submission1.csv',index=False)
```

Decision Tree Model:

```
# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(max_depth=15,min_samples_leaf=300)
regressor.fit(X_train, y_train)

```

```
DecisionTreeRegressor(criterion='mse', max_depth=15, max_features=None,
```

```
                max_leaf_nodes=None, min_impurity_decrease=0.0,
                min_impurity_split=None, min_samples_leaf=300,
                min_samples_split=2, min_weight_fraction_leaf=0.0,
                presort=False, random_state=None, splitter='best')
```

```
1 # Predicting the test set results
2 y_pred = regressor.predict(X_test)
3 y_pred
```

```
   array([1673.98398729, 1349.51290433,  471.30684669, ..., 1892.06614452,
          3805.94860417, 1349.51290433])
```

```
1 tree_accuracy = round(regressor.score(X_train,y_train),2)
2 tree_accuracy
```

```
   0.59
```

```
1 r2_score(y_train, regressor.predict(X_train))
```

```
   0.5884050821570486
```

```
1 import warnings
2 warnings.filterwarnings('ignore')
3 cv_score = cross_val_score(regressor, X_train, y_train, cv=5, scoring='mean_squa
4 print(np.sqrt(np.abs(cv_score)))
```

```
   [1138.77137157 1109.42501179 1145.66395939 1113.2648073  1129.0816826 ]
```

```
1 print("RMSE : %.4g" % np.sqrt(metrics.mean_squared_error(y_train, regressor.pred
```

```
   RMSE : 1095
```

```
1 submission = pd.DataFrame({
2 'Item_Identifier':test2['Item_Identifier'],
3 'Outlet_Identifier':test2['Outlet_Identifier'],
4 'Item_Outlet_Sales': y_pred
5 },columns=['Item_Identifier','Outlet_Identifier','Item_Outlet_Sales'])
```

```
1 submission.to_csv('submission2.csv',index=False)
```

Random Forest Model:

```
1 # Fitting Random Forest Regression to the dataset
2 from sklearn.ensemble import RandomForestRegressor
3 regressor = RandomForestRegressor(n_estimators=100,max_depth=6, min_samples_lea
4 regressor.fit(X_train, y_train)
```

```
   RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=6,
                max_features='auto', max_leaf_nodes=None,
```

```
                min_impurity_decrease=0.0, min_impurity_split=None,
                min_samples_leaf=50, min_samples_split=2,
                min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=4,
                oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
1 # Predicting the test set results
2 y_pred = regressor.predict(X_test)
3 y_pred
```

```
   array([1643.87106725, 1364.24193091,  603.09113992, ..., 1957.62183676,
          3698.60040819, 1290.25320329])
```

```
1 rf_accuracy = round(regressor.score(X_train,y_train),2)
2 rf_accuracy
```

```
   0.61
```

```
1 r2_score(y_train, regressor.predict(X_train))
```

```
   0.6125814698282157
```

```
1 import warnings
2 warnings.filterwarnings('ignore')
3 cv_score = cross_val_score(regressor, X_train, y_train, cv=5, scoring='mean_squa
4 print(np.sqrt(np.abs(cv_score)))
```

```
   [1100.46298396 1077.70836131 1077.65325884 1069.0502564  1083.85364282]
```

```
1 print("RMSE : %.4g" % np.sqrt(metrics.mean_squared_error(y_train, regressor.pred
```

```
   RMSE : 1062
```

```
1 submission = pd.DataFrame({
2 'Item_Identifier':test2['Item_Identifier'],
3 'Outlet_Identifier':test2['Outlet_Identifier'],
4 'Item_Outlet_Sales': y_pred
5 },columns=['Item_Identifier','Outlet_Identifier','Item_Outlet_Sales'])
```

```
1 submission.to_csv('submission3.csv',index=False)
```