```
import numpy as np
import pandas as pd
import matplotlib.pyplot
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
```

```
train_cpy=pd.read_csv("bigmartsales1.train.csv")
test_cpy=pd.read_csv("bigmartsales1.test.csv")
```

```
train_cpy.shape
```

```
(8523, 12)
```

```
test_cpy.shape
```

```
(5681, 11)
```

```
train_dtypes, test_dtypes
```

```
(Item_Identifier                 object
 Item_Weight                    float64
 Item_Fat_Content               object
 Item_Visibility                float64
 Item_Type                      object
 Outlet_Establishment_Year       int64
 Outlet_Size                    object
 Outlet_Location_Type           object
 Outlet_Type                    object
 Item_Outlet_Sales              float64
 dtype: object,
 Item_Identifier                 object
 Item_Weight                    float64
 Item_Fat_Content               object
 Item_Visibility                float64
 Item_Type                      object
 Outlet_Establishment_Year       int64
 Outlet_Size                    object
 Outlet_Location_Type           object
 Outlet_Type                    object
 dtype: object)
```

```
train_cpy.dtypes
```

```
Item_Identifier                 object
Item_Weight                    float64
Item_Fat_Content               object
Item_Visibility                float64
Item_Type                      object
Outlet_Establishment_Year       int64
Outlet_Size                    object
Outlet_Location_Type           object
Outlet_Type                    object
Item_Outlet_Sales              float64
dtype: object
```

```
train_cpy.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Small	Tier 3	Grocery Store	732.3800
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

```
test_dtypes
```

```
Item_Identifier                 object
Item_Weight                    float64
Item_Fat_Content               object
Item_Visibility                float64
Item_Type                      object
Outlet_Establishment_Year       int64
Outlet_Size                    object
Outlet_Location_Type           object
Outlet_Type                    object
dtype: object
```

```
test_cpy.dtypes
```

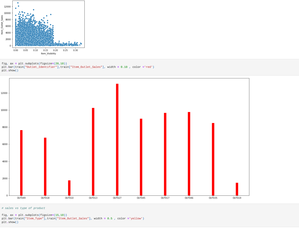
```
Item_Identifier                 object
Item_Weight                    float64
Item_Fat_Content               object
Item_Visibility                float64
Item_Type                      object
Outlet_Establishment_Year       int64
Outlet_Size                    object
Outlet_Location_Type           object
Outlet_Type                    object
dtype: object
```

```
train_cpy.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247540
50%	12.600000	0.054021	143.647800	1999.000000	1794.331000
75%	16.850000	0.094037	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.884800	2009.000000	13086.964800

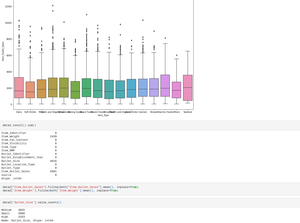
```
sns.scatterplot(x="Item_Visibility", y="Item_Outlet_Sales")
```

```
C:\Program Files\Anaconda3\envs\jupyterenv\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argum
ent will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
<AxesSubplot:xlabel='Item_Visibility', ylabel='Item_Outlet_Sales'>
```

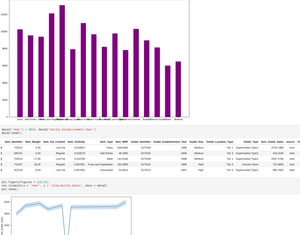



```
fig, ax = plt.subplots(figsize=(15,10))
sns.boxplot(x="Item_Type", y="Item_Outlet_Sales", width = 0.8 , color ='red')
plt.show()
```

```
# sales in type of product
fig, ax = plt.subplots(figsize=(15,10))
plt.boxplot("Item_Type", data=("Item_Outlet_Sales"), width = 0.8 , color ='yellow')
plt.show()
```




```
data = pd.concat([train["Item_Type"], train["Item_Outlet_Sales"]], axis=1)
fig, ax = plt.subplots(figsize=(15,10))
fig = sns.boxplot(x="Item_Type", y="Item_Outlet_Sales", data=data)
fig.axis(ymin=0, ymax=15000)
plt.show()
```




```
#BOXPLOT ANALYSIS
sns.boxplot(x = 'Outlet_Size', y = 'Item_Outlet_Sales', data = data)
plt.show()
```

```
sns.boxplot(x = 'Outlet_Location_Type', y = 'Item_Outlet_Sales', data = data)
plt.show()
```



```
plt.figure(figsize = (15,20))
sns.boxplot(x = 'Outlet_Identifier', y = 'Item_Outlet_Sales', data =data)
plt.show()
```



```
sns.boxplot(x = 'Item_Fat_Content', y = 'Item_Outlet_Sales', data = data)
plt.show()
```



```
plt.figure(figsize = (15,20))
sns.boxplot(x = 'Item_Type', y = 'Item_Outlet_Sales', data = data)
plt.show()
```



```
train.isnull().sum()
```

```
Item_Identifier                 0
Item_Weight                  1463
Item_Fat_Content             1400
Item_Visibility                 0
Item_Type                       0
Outlet_Establishment_Year       0
Outlet_Size                  2410
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales               0
dtype: int64
```

```
train["Item_Outlet_Sales"].fillna(train["Item_Outlet_Sales"].mean(), inplace=True)
train["Item_Weight"].fillna(train["Item_Weight"].mean(), inplace=True)
```

```
train["Outlet_Size"].value_counts()
```

```
Medium    2793
Small     2388
High      1553
Name: Outlet_Size, dtype: int64
```

```
train["Outlet_Size"].fillna("Medium", inplace = True)
```

```
train.isnull().sum()
```

```
Item_Identifier                 0
Item_Weight                     0
Item_Fat_Content             1400
Item_Visibility                 0
Item_Type                       0
Outlet_Establishment_Year       0
Outlet_Size                     0
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales               0
dtype: int64
```

```
#Replacing 0 with NaN
train["Item_Visibility"].replace(0, np.nan, inplace=True)
```

```
#Check missing values in numeric variables
train["Item_Visibility"].fillna(train["Item_Visibility"].median(), inplace=True)
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380	train
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228	train
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700	train
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Small	Tier 3	Grocery Store	732.3800	train
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052	train

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

```
Low Fat    5089
Regular    2889
Name: Item_Fat_Content, dtype: int64
```

```
fig, ax = plt.subplots(figsize=(15,10))
sns.boxplot("Item_Type", data=("Outlet_Sales"), width = 0.8 , color ='purple')
plt.show()
```



```
train["Year"] = 2013 - train["Outlet_Establishment_Year"]
train.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales	source	Year
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380	train	14
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228	train	4
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700	train	14
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Small	Tier 3	Grocery Store	732.3800	train	15
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052	train	26

```
train.Item_Identifier = pd.Series(
    [x[:2] + str(len(x.split('\t'))) for x in
     list(train['Item_Identifier'])], dtype=train['Item_Identifier'].dtype)
train['Item_Identifier'].value_counts()
```

```
NC    1599
FD    6125
DR     799
Name: Item_Identifier, dtype: int64
```

```
# ENCODING LABEL
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
train['Outlet_Size'] = le.fit_transform(train['Outlet_Identifier'])
train.ix[:,('Item_Fat_Content','Outlet_Location_Type','Outlet_Type')].head()
# ONE HOT ENCODING
for i in cat_var:
    train = pd.get_dummies(train,columns=[i])
```

	Item_Fat_Content	Outlet_Location_Type	Outlet_Type
0	Low Fat	Tier 1	Supermarket Type1
1	Regular	Tier 3	Supermarket Type2
2	Low Fat	Tier 1	Supermarket Type1
3	Regular	Tier 3	Grocery Store
4	Low Fat	Tier 3	Supermarket Type1

```
train1 = train[['Item_Identifier', 'Outlet_Identifier', 'Outlet_Establishment_Year', 'Outlet_Size', 'Outlet_Type', 'Item_Type_Combined']]
```

	Item_Identifier	Item_Weight	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Item_Outlet_Sales	source	Year	_	Outlet_Size_0	Outlet_Size_1	Outlet_Size_2	Outlet_Type_0	Outlet_Type_1	Outlet_Type_2
0	FDA15	9.30	0.016047	Dairy	249.8092	OUT049	1999	3735.1380	train	14	1	0	1	0	1	0	0
1	DRC01	5.92	0.019278	Soft Drinks	48.2692	OUT018	2009	443.4228	train	4	1	0	1	0	0	1	0
2	FDN15	17.50	0.016760	Meat	141.6180	OUT049	1999	2097.2700	train	14	1	0	1	0	1	0	0
3	FDX07	19.20	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	732.3800	train	15	1	1	0	0	0	0	1
4	NCD19	8.93	0.000000	Household	53.8614	OUT013	1987	994.7052	train	26	1	0	0	1	1	0	0

5 rows × 17 columns

```
train1 = train[['Item_Identifier', 'Outlet_Identifier', 'Item_Identifier']]
train1 = train1.iloc[:5,:]
```

	Item_Identifier	Outlet_Identifier	Item_Identifier
0	FDA15	OUT049	FDA15
1	DRC01	OUT018	DRC01
2	FDN15	OUT049	FDN15
3	FDX07	OUT010	FDX07
4	NCD19	OUT013	NCD19

```
train1 = train[['Item_Identifier', 'Outlet_Identifier', 'Item_Identifier']]
train1 = train1.iloc[:5,:]
train1.shape
```

```
(5, 3)
```

```
#BUILDING MODELS
# Linear regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
lr = LinearRegression()
lr.fit(x_train, y_train)
```

```
LinearRegression()
```

```
[r_squared = round(regressor.score(X_train,y_train) * 100,2)
print(r_squared)
```

```
56.49
```

```
from sklearn.metrics import mean_squared_error
print("RMSE : %.4g",%(np.sqrt(ridge_mse(y_pred,y_train))))
```

```
RMSE : 1147.1842638177824
```

```
#DECISION TREE
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(max_depth=15,min_samples_leaf=100)
regressor.fit(x_train,y_train)
```

```
DecisionTreeRegressor(max_depth=15, min_samples_leaf=100)
```

```
[rmse_array = round(regressor.score(X_train,y_train),2)
print(rmse_array)
```

```
0.73
```

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

```
0.6853254430922
```

```
print("RMSE : %.4g",%(np.sqrt(mean_squared_error(y_train, regressor.predict(X_train))))
```

```
RMSE : 1054.5441666111
```

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=100,max_depth=15, min_samples_leaf=50,n_jobs=4)
regressor.fit(x_train,y_train)
```

```
RandomForestRegressor(max_depth=15, min_samples_leaf=50, n_jobs=4)
```

```
[r_squared = round(regressor.score(X_train,y_train),2)
print(r_squared)
```

```
0.73
```

```
print("RMSE : %.4g",%(np.sqrt(mean_squared_error(y_train, regressor.predict(X_train))))
```

```
RMSE : 1041.3410000111
```