Problem Statement

The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. The aim is to build a predictive model and find out the sales of each product at a particular store.

Using this model, BigMart will try to understand the properties of products and stores which play a key role in increasing sales.

Hypothesis Generation

Make it a practice to do this before solving any ML problem. Ideally,before seeing the data or else, you might end up with biased hypotheses.

What could affect the target variable (sales)?

1. Time of week : Weekends usually are more busy
2. Time of day : Higher sales in the mornings and late evenings
3. Time of year : Higher sales at end of the year
4. Store size and location
5. Items with more shelf space

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt

%matplotlib inline
```

In [2]:
```
train = pd.read_csv('bigmart_train.csv')
```

In [3]:
```
train.head(10)
```

Out[3]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	C
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	
5	FDP36	10.395	Regular	0.000000	Baking Goods	51.4008	OUT018	2009	Medium	
6	FDO10	13.650	Regular	0.012741	Snack Foods	57.6588	OUT013	1987	High	
7	FDP10	NaN	Low Fat	0.127470	Snack Foods	107.7622	OUT027	1985	Medium	
8	FDH17	16.200	Regular	0.016687	Frozen Foods	96.9726	OUT045	2002	NaN	
9	FDU28	19.200	Regular	0.094450	Frozen Foods	187.8214	OUT017	2007	NaN	

In [4]:
```
train.shape
```

Out[4]:
```
(8523, 12)
```

In [5]:
```
train.isnull().sum()
```

Out[5]:
```
Item_Identifier               0
Item_Weight                1463
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                2410
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
dtype: int64
```

learner tasks (intentionally skipped)

Exploratory Data Analysis

1. Univariate analysis on
   A. Target variable - Item outlet sales (histogram)
   B. Independent variables (numeric and categorical) - histograms
2. Bivariate analysis
   A. Explore IV's with respect to the target variable - scatterplots
3. Correlation matrix

In [6]:
```
train['Item_Fat_Content'].unique()
#notice Low fat, Low Fat, LF are all the same variable
```

Out[6]:
```
array(['Low Fat', 'Regular', 'low fat', 'LF', 'reg'], dtype=object)
```

In [7]:
```
train['Outlet_Establishment_Year'].unique()
```

Out[7]:
```
array([1999, 2009, 1998, 1987, 1985, 2002, 2007, 1997, 2004], dtype=int64)
```

In [8]:
```
train['Outlet_Age'] = 2020 - train['Outlet_Establishment_Year']
train.head()
```

Out[8]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	C
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	

In [9]:
```
train['Outlet_Size'].unique()
```

Out[9]:
```
array(['Medium', nan, 'High', 'Small'], dtype=object)
```

In [10]:
```
train.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
Item_Weight	7060.0	12.857645	4.643456	4.555	8.773750	12.600000	16.850000	21.350000
Item_Visibility	8523.0	0.066132	0.051598	0.000	0.026989	0.053931	0.094585	0.328391
Item_MRP	8523.0	140.992782	62.275067	31.290	93.826500	143.012800	185.643700	266.888400
Outlet_Establishment_Year	8523.0	1997.831867	8.371760	1985.000	1987.000000	1999.000000	2004.000000	2009.000000
Item_Outlet_Sales	8523.0	2181.288914	1706.499616	33.290	834.247400	1794.331000	3101.296400	13086.964800
Outlet_Age	8523.0	22.168133	8.371760	11.000	16.000000	21.000000	33.000000	35.000000

In [51]:
```
train['Item_Fat_Content'].value_counts()
```

Out[51]:
```
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64
```

In [52]:
```
train['Outlet_Size'].value_counts()
```

Out[52]:
```
Medium    2793
Small     2388
High       932
Name: Outlet_Size, dtype: int64
```

In [53]:
```
train['Outlet_Size'].mode()[0]
```

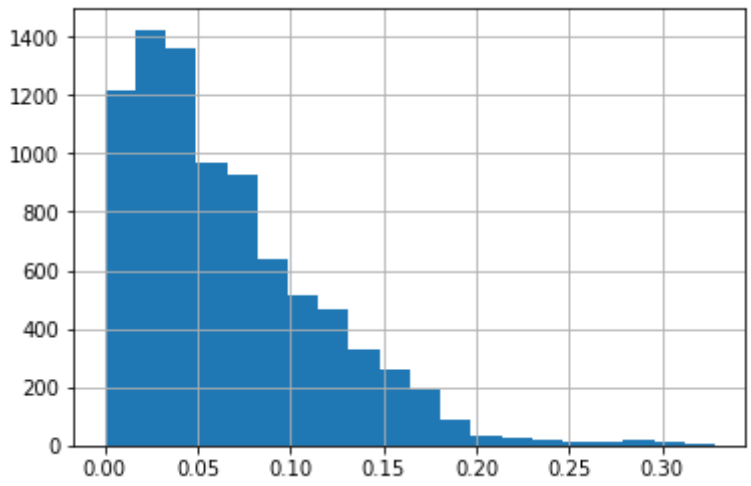
Out[53]:
```
'Medium'
```

In [54]:
```
# fill the na for outlet size with medium
train['Outlet_Size'] = train['Outlet_Size'].fillna(train['Outlet_Size'].mode()[0])
```

In [55]:
```
# fill the na for item weight with the mean of weights
train['Item_Weight'] = train['Item_Weight'].fillna(train['Item_Weight'].mean())
```

In [56]:
```
train['Item_Visibility'].hist(bins=20)
```

Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x211c4feb898>



In [11]:
```
# delete the observations

Q1 = train['Item_Visibility'].quantile(0.25)
Q3 = train['Item_Visibility'].quantile(0.75)
IQR = Q3 - Q1
filt_train = train.query('(@Q1 - 1.5 * @IQR) <= Item_Visibility <= (@Q3 + 1.5 * @IQR)')
```

In [12]:
```
filt_train
```

Out[12]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	High
5	FDP36	10.395	Regular	0.000000	Baking Goods	51.4008	OUT018	2009	Medium
6	FDO10	13.650	Regular	0.012741	Snack Foods	57.6588	OUT013	1987	High
7	FDP10	NaN	Low Fat	0.127470	Snack Foods	107.7622	OUT027	1985	Medium
8	FDH17	16.200	Regular	0.016687	Frozen Foods	96.9726	OUT045	2002	NaN
9	FDU28	19.200	Regular	0.094450	Frozen Foods	187.8214	OUT017	2007	NaN
10	FDY07	11.800	Low Fat	0.000000	Fruits and Vegetables	45.5402	OUT049	1999	Medium
11	FDA03	18.500	Regular	0.045464	Dairy	144.1102	OUT046	1997	Small
12	FDX32	15.100	Regular	0.100014	Fruits and Vegetables	145.4786	OUT049	1999	Medium
13	FDS46	17.600	Regular	0.047257	Snack Foods	119.6782	OUT046	1997	Small
14	FDF32	16.350	Low Fat	0.068024	Fruits and Vegetables	196.4426	OUT013	1987	High
15	FDP49	9.000	Regular	0.069089	Breakfast	56.3614	OUT046	1997	Small
16	NCB42	11.800	Low Fat	0.008596	Health and Hygiene	115.3492	OUT018	2009	Medium
17	FDP49	9.000	Regular	0.069196	Breakfast	54.3614	OUT049	1999	Medium
18	DRI11	NaN	Low Fat	0.034238	Hard Drinks	113.2834	OUT027	1985	Medium
19	FDU02	13.350	Low Fat	0.102492	Dairy	230.5352	OUT035	2004	Small
20	FDN22	18.850	Regular	0.138190	Snack Foods	250.8724	OUT013	1987	High
21	FDW12	NaN	Regular	0.035400	Baking Goods	144.5444	OUT027	1985	Medium
22	NCB30	14.600	Low Fat	0.025698	Household	196.5084	OUT035	2004	Small
23	FDC37	NaN	Low Fat	0.057557	Baking Goods	107.6938	OUT019	1985	Small
24	FDR28	13.850	Regular	0.025896	Frozen Foods	165.0210	OUT046	1997	Small
25	NCD06	13.000	Low Fat	0.099887	Household	45.9060	OUT017	2007	NaN
26	FDV10	7.645	Regular	0.066693	Snack Foods	42.3112	OUT035	2004	Small
27	DRJ59	11.650	low fat	0.019356	Hard Drinks	39.1164	OUT013	1987	High
28	FDE51	5.925	Regular	0.161467	Dairy	45.5086	OUT010	1998	NaN
29	FDC14	NaN	Regular	0.072222	Canned	43.6454	OUT019	1985	Small
...	...	...	...	...	...	...	...	...	...
8492	FDT34	9.300	Low Fat	0.174350	Snack Foods	104.4964	OUT046	1997	Small
8493	FDP21	7.420	Regular	0.025886	Snack Foods	189.1872	OUT017	2007	NaN
8494	NCI54	15.200	Low Fat	0.000000	Household	110.4912	OUT017	2007	NaN
8495	FDE22	9.695	Low Fat	0.029567	Snack Foods	160.4920	OUT035	2004	Small
8496	FDJ57	7.420	Regular	0.021696	Seafood	185.3582	OUT017	2007	NaN
8497	FDT08	13.650	Low Fat	0.049209	Fruits and Vegetables	150.0050	OUT035	2004	Small

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size
8498	NCP54	15.350	Low Fat	0.035293	Household	124.5730	OUT018	2009	Medium
8499	NCK53	11.600	Low Fat	0.037574	Health and Hygiene	100.0042	OUT035	2004	Small
8500	NCQ42	20.350	Low Fat	0.000000	Household	125.1678	OUT017	2007	NaN
8501	FDW21	5.340	Regular	0.005998	Snack Foods	100.4358	OUT017	2007	NaN
8502	NCH43	8.420	Low Fat	0.070712	Household	216.4192	OUT045	2002	NaN
8503	FDQ44	20.500	Low Fat	0.036133	Fruits and Vegetables	120.1756	OUT035	2004	Small
8504	NCN18	NaN	Low Fat	0.124111	Household	111.7544	OUT027	1985	Medium
8505	FDB46	10.500	Regular	0.094146	Snack Foods	210.8244	OUT018	2009	Medium
8506	DRF37	17.250	Low Fat	0.084676	Soft Drinks	263.1910	OUT018	2009	Medium
8507	FDN28	5.880	Regular	0.030242	Frozen Foods	101.7990	OUT035	2004	Small
8508	FDW31	11.350	Regular	0.043246	Fruits and Vegetables	199.4742	OUT045	2002	NaN
8510	FDN58	13.800	Regular	0.056862	Snack Foods	231.5984	OUT035	2004	Small
8511	FDF05	17.500	Low Fat	0.026980	Frozen Foods	262.5910	OUT018	2009	Medium
8512	FDR26	20.700	Low Fat	0.042801	Dairy	178.3028	OUT013	1987	High
8513	FDH31	12.000	Regular	0.020407	Meat	99.9042	OUT035	2004	Small
8514	FDA01	15.000	Regular	0.054489	Canned	57.5904	OUT045	2002	NaN
8515	FDH24	20.700	Low Fat	0.021518	Baking Goods	157.5288	OUT018	2009	Medium
8516	NCJ19	18.600	Low Fat	0.118661	Others	58.7588	OUT018	2009	Medium
8517	FDF53	20.750	reg	0.083607	Frozen Foods	178.8318	OUT046	1997	Small
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	OUT013	1987	High
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	OUT045	2002	NaN
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	OUT035	2004	Small
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	OUT018	2009	Medium
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	OUT046	1997	Small

8379 rows × 13 columns

In [59]:
```
filt_train.shape, train.shape
```

Out[59]:
```
((8379, 13), (8523, 13))
```

In [60]:
```
train = filt_train
train.shape
```

Out[60]:
```
(8379, 13)
```

In [61]:
```
#train['Item_Visibility'].value_counts()
```

In [62]:
```
#creating a category
train['Item_Visibility_bins'] = pd.cut(train['Item_Visibility'], [0.000, 0.065, 0.13, 0.2], labels=['Low Viz', 'Viz',
'High Viz'])
```

In [63]:
```
train['Item_Visibility_bins'].value_counts()
```

Out[63]:
```
Low Viz     4403
Viz         2557
High Viz     893
Name: Item_Visibility_bins, dtype: int64
```

In [64]:
```
train['Item_Visibility_bins'] = train['Item_Visibility_bins'].replace(np.nan,'Low Viz',regex=True)
```

In [65]:
```
train['Item_Fat_Content'] = train['Item_Fat_Content'].replace(['low fat', 'LF'], 'Low Fat')
```

In [66]:
```
train['Item_Fat_Content'] = train['Item_Fat_Content'].replace('reg', 'Regular')
```

In [67]:
```
train.head()
```

Out[67]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	C
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	Medium	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	

Encoding Categorical Variables

In [68]:
```
le = LabelEncoder()
```

In [69]:
```
train['Item_Fat_Content'].unique()
```

Out[69]:
```
array(['Low Fat', 'Regular'], dtype=object)
```

In [70]:
```
train['Item_Fat_Content'] = le.fit_transform(train['Item_Fat_Content'])
```

In [71]:
```
train['Item_Visibility_bins'] = le.fit_transform(train['Item_Visibility_bins'])
```

In [72]:
```
train['Outlet_Size'] = le.fit_transform(train['Outlet_Size'])
```

In [73]:
```
train['Outlet_Location_Type'] = le.fit_transform(train['Outlet_Location_Type'])
```

In [39]:
```
# create dummies for outlet type
```

In [75]:
```
dummy = pd.get_dummies(train['Outlet_Type'])
dummy.head()
```

Out[75]:

	Grocery Store	Supermarket Type1	Supermarket Type2	Supermarket Type3
0	0	1	0	0
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0
4	0	1	0	0

In [76]:
```
train = pd.concat([train, dummy], axis=1)
```

In [77]:
```
train.dtypes
```

Out[77]:
```
Item_Identifier              object
Item_Weight                 float64
Item_Fat_Content              int32
Item_Visibility             float64
Item_Type                    object
Item_MRP                    float64
Outlet_Identifier            object
Outlet_Establishment_Year     int64
Outlet_Size                   int32
Outlet_Location_Type          int32
Outlet_Type                  object
Item_Outlet_Sales           float64
Outlet_Age                    int64
Item_Visibility_bins          int32
Grocery Store                 uint8
Supermarket Type1             uint8
Supermarket Type2             uint8
Supermarket Type3             uint8
dtype: object
```

In [52]:
```
# got to drop all the object types features
train = train.drop(['Item_Identifier', 'Item_Type', 'Outlet_Identifier', 'Outlet_Type','Outlet_Establishment_Year'], a
xis=1)
```

In [53]:
```
train.columns
```

Out[53]:
```
Index(['Item_Weight', 'Item_Fat_Content', 'Item_Visibility', 'Item_MRP',
       'Outlet_Size', 'Outlet_Location_Type', 'Item_Outlet_Sales',
       'Outlet_Age', 'Item_Visibility_bins'],
      dtype='object')
```

In [54]:
```
train.head()
```

Out[54]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Size	Outlet_Location_Type	Item_Outlet_Sales	Outlet_Age	Item_Visibility_bins
0	9.30	0	0.016047	249.8092	1	0	3735.1380	19	1
1	5.92	1	0.019278	48.2692	1	2	443.4228	9	1
2	17.50	0	0.016760	141.6180	1	0	2097.2700	19	1
3	19.20	1	0.000000	182.0950	1	2	732.3800	20	1
4	8.93	0	0.000000	53.8614	0	2	994.7052	31	1

Linear Regression

In [78]:
```
# build the linear regression model
X = train.drop('Item_Outlet_Sales', axis=1)
y = train.Item_Outlet_Sales
```

In [79]:
```
test = pd.read_csv('bigmart_test.csv')
test['Outlet_Size'] = test['Outlet_Size'].fillna('Medium')
```

In [80]:
```
test['Item_Visibility_bins'] = pd.cut(test['Item_Visibility'], [0.000, 0.065, 0.13, 0.2], labels=['Low Viz', 'Viz', 'H
igh Viz'])
```

In [81]:
```
test['Item_Weight'] = test['Item_Weight'].fillna(test['Item_Weight'].mean())
```

In [82]:
```
test['Item_Visibility_bins'] = test['Item_Visibility_bins'].fillna('Low Viz')
test['Item_Visibility_bins'].head()
```

Out[82]:
```
0    Low Viz
1    Low Viz
2        Viz
3    Low Viz
4        Viz
Name: Item_Visibility_bins, dtype: category
Categories (3, object): [Low Viz < Viz < High Viz]
```

In [83]:
```
test['Item_Fat_Content'] = test['Item_Fat_Content'].replace(['low fat', 'LF'], 'Low Fat')
test['Item_Fat_Content'] = test['Item_Fat_Content'].replace('reg', 'Regular')
```

In [84]:
```
test['Item_Fat_Content'] = le.fit_transform(test['Item_Fat_Content'])
```

In [85]:
```
test['Item_Visibility_bins'] = le.fit_transform(test['Item_Visibility_bins'])
```

In [86]:
```
test['Outlet_Size'] = le.fit_transform(test['Outlet_Size'])
```

In [87]:
```
test['Outlet_Location_Type'] = le.fit_transform(test['Outlet_Location_Type'])
```

In [88]:
```
test['Outlet_Age'] = 2020 - test['Outlet_Establishment_Year']
```

In [89]:
```
dummy = pd.get_dummies(test['Outlet_Type'])
test = pd.concat([test, dummy], axis=1)
```

In [98]:
```
X_test = test.drop(['Item_Identifier', 'Item_Type', 'Outlet_Identifier', 'Outlet_Type','Outlet_Establishment_Year'], a
xis=1)
```

In [99]:
```
X.columns, X_test.columns
```

Out[99]:
```
(Index(['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', 'Outlet_Age', 'Item_Visibility_bins', 'Grocery Store',
        'Supermarket Type1', 'Supermarket Type2', 'Supermarket Type3'],
       dtype='object'),
 Index(['Item_Weight', 'Item_Fat_Content', 'Item_Visibility', 'Item_MRP',
        'Outlet_Size', 'Outlet_Location_Type', 'Item_Visibility_bins',
        'Outlet_Age', 'Grocery Store', 'Supermarket Type1', 'Supermarket Type2',
        'Supermarket Type3'],
       dtype='object'))
```

In [100]:
```
from sklearn import model_selection
xtrain,xtest,ytrain,ytest = model_selection.train_test_split(X,y,test_size=0.3,random_state=42)
```

In [101]:
```
lin = LinearRegression()
```

In [103]:
```
lin.fit(xtrain, ytrain)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-103-8faeb3f7dfba> in <module>
----> 1 lin.fit(xtrain, ytrain)

D:\Python\lib\site-packages\sklearn\linear_model\base.py in fit(self, X, y, sample_weight)
    456         n_jobs_ = self.n_jobs
    457         X, y = check_X_y(X, y, accept_sparse=['csr', 'csc', 'coo'],
--> 458                          y_numeric=True, multi_output=True)
    459
    460         if sample_weight is not None and np.atleast_1d(sample_weight).ndim > 1:

D:\Python\lib\site-packages\sklearn\utils\validation.py in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype,
order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric,
warn_on_dtype, estimator)
    754                     ensure_min_features=ensure_min_features,
    755                     warn_on_dtype=warn_on_dtype,
--> 756                     estimator=estimator)
    757     if multi_output:
    758         y = check_array(y, 'csr', force_all_finite=True, ensure_2d=False,

D:\Python\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse, accept_large_sparse, dty
pe, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, warn_on_dtype, estim
ator)
    565         # make sure we actually converted to numeric:
    566         if dtype_numeric and array.dtype.kind == "O":
--> 567             array = array.astype(np.float64)
    568         if not allow_nd and array.ndim >= 3:
    569             raise ValueError("Found array with dim %d. %s expected <= 2."

ValueError: could not convert string to float: 'DRE49'
```

In [72]:
```
predictions = lin.predict(xtest)
print(sqrt(mean_squared_error(ytest, predictions)))
```

```
1353.3824567853526
```

A good RMSE for this problem is atleast 1150.

Try using Ridge, Lasso, ElasticNet, and compare the RMSE scores. You can try Gradient boosting too.

Lesson 08: Ensemble learning is about xgboost

Once you have learnt the XGboost techniques, come back and re-work on this problem. *XGBOOST* will give the lowest RMSE.

In [59]:
```
from sklearn.linear_model import Ridge
ridgeReg = Ridge(alpha=0.001, normalize=True)
ridgeReg.fit(xtrain,ytrain)
print(sqrt(mean_squared_error(ytrain, ridgeReg.predict(xtrain))))
print(sqrt(mean_squared_error(ytest, ridgeReg.predict(xtest))))
print('R2 Value/Coefficient of Determination: {}'.format(ridgeReg.score(xtest, ytest)))
```

```
1139.5277714448032
1118.3593685856831
R2 Value/Coefficient of Determination: 0.548659756640925
```

In [73]:
```
from sklearn.linear_model import Lasso
lassoreg = Lasso(alpha=0.001, normalize=True)
lassoreg.fit(xtrain, ytrain)

print(sqrt(mean_squared_error(ytrain, lassoreg.predict(xtrain))))
print(sqrt(mean_squared_error(ytest, lassoreg.predict(xtest))))
print('R2 Value/Coefficient of Determination: {}'.format(lassoreg.score(xtest, ytest)))
```

```
1380.366905556028
1353.3760424120942
R2 Value/Coefficient of Determination: 0.3390352983965016
```

In [74]:
```
from sklearn.linear_model import ElasticNet
Elas = ElasticNet(alpha=0.001, normalize=True)
Elas.fit(xtrain, ytrain)

print(sqrt(mean_squared_error(ytrain, Elas.predict(xtrain))))
print(sqrt(mean_squared_error(ytest, Elas.predict(xtest))))
print('R2 Value/Coefficient of Determination: {}'.format(Elas.score(xtest, ytest)))
```

```
1576.152560123294
1529.8250146583991
R2 Value/Coefficient of Determination: 0.15545107363331978
```

In [ ]:
```

```