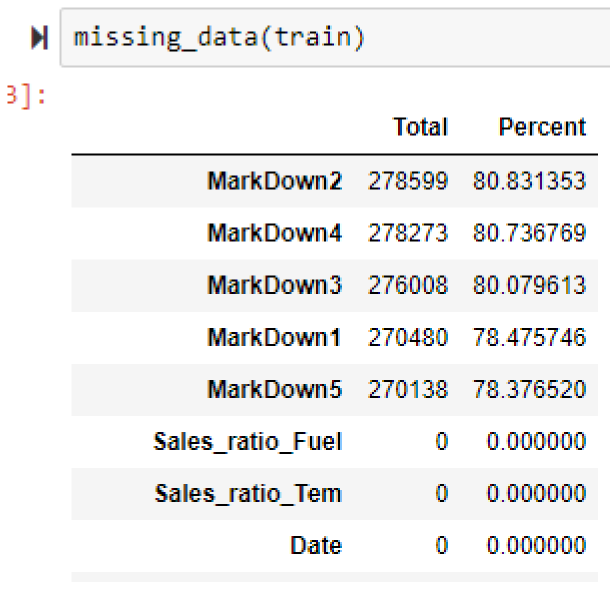
DSBA 6156 – Fall 2019
Individual Project

Topic: Regression, Classification and Clustering implementation in Walmart sales data

We are provided with historical data of 45 Walmart stores and asked to conduct both supervised and unsupervised modelling techniques in manipulating data effectively to discover new insights out of it. The data covers the period between 2010-02-05 and 2012-11-01. There are overall 12 variables for 45 anonymized stores ranging from different size and departments. The data divided in to master and test data so there is no need to split data for modelling later on though we can make sampling for cross-validation out of master data as well

We start our analysis by getting big picture of data and examine relationship among variables. In this regard we perform Exploratory Data Analysis (EDA), which is an open-ended process where we calculate statistics and make figures to find trends, anomalies, patterns, or relationships within the data. The goal of EDA is to learn what our data can tell us. It generally starts out with a high level overview, then narrows in to specific areas as we find intriguing areas of the data. The findings may be interesting in their own right, or they can be used to inform our modeling choices, such as by helping us decide which features to use.

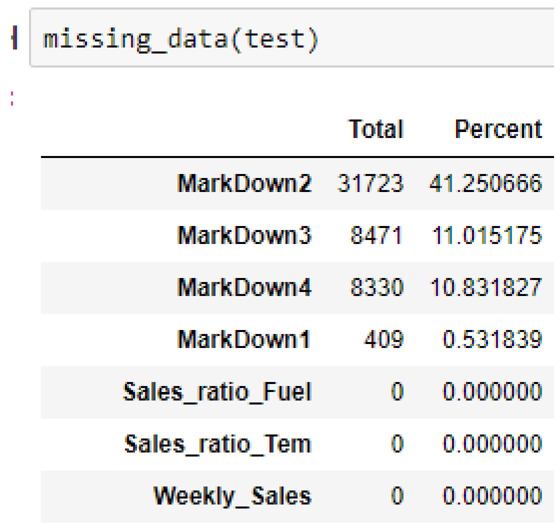
First of all, we check for the shape of data and as we can see Master data contains 344667 observations and 15 variables while test data also contains 15 variables and 76903 observations. Please refer to my jupyter notebook for more detailed analysis. Also, I have created some new variables out of given data so that ease and enhance predictive power of our analysis. I have divided Date column into separate columns like Week, Month, Day and number of days so that individually observe effect of each these variables. Also I created different ratios that can be seen as ratio of sales to see more derived effect. Before moving forward, I checked whether the data contains duplicate values and found out that the data is free from duplicates. However, I have discovered that there are a number of missing values on both master and test data only columns with promotional markdown events have the highest percentage of missing values.



```
missing_data(train)
```

3]:

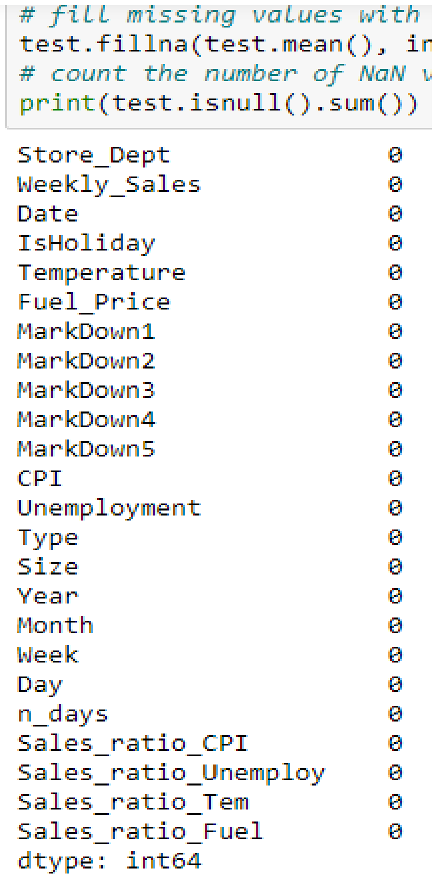
	Total	Percent
MarkDown2	278599	80.831353
MarkDown4	278273	80.736769
MarkDown3	276008	80.079613
MarkDown1	270480	78.475746
MarkDown5	270138	78.376520
Sales_ratio_Fuel	0	0.000000
Sales_ratio_Tem	0	0.000000
Date	0	0.000000



```
missing_data(test)
```

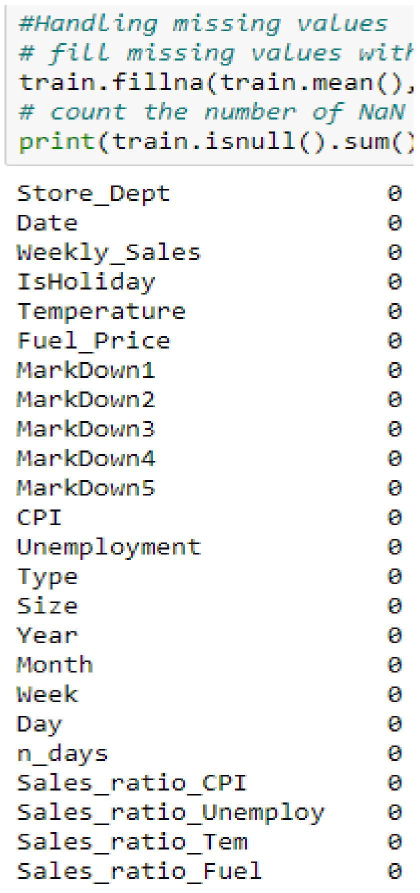
	Total	Percent
MarkDown2	31723	41.250666
MarkDown3	8471	11.015175
MarkDown4	8330	10.831827
MarkDown1	409	0.531839
Sales_ratio_Fuel	0	0.000000
Sales_ratio_Tem	0	0.000000
Weekly_Sales	0	0.000000

But then I simply imputed missing values with their mean as unfortunately I was not able to find right package in python that could help me to use Amelia or MICE package substitutes for R. The overall imputation was solved to get rid of empty values in data. Also, I used label encoding from Sklearn to transform some of the categorical variables in to dummy. I did not use one hot encoding as it would expand my data horizontally which in turn would affect my modest laptop computer memory efficiency. Other than that, I created a loop function that could enhance label encoding for multiple selected columns as I could not find right way of doing it given LabelEncoder() function.



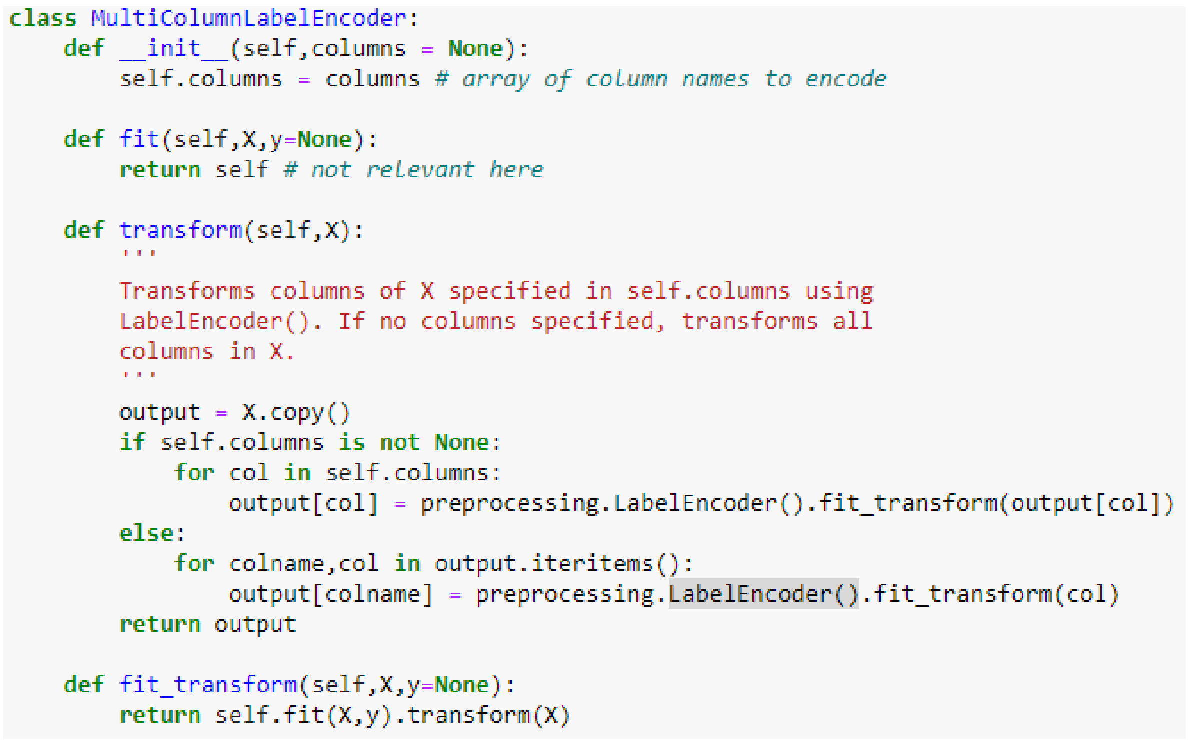
```
# fill missing values with
test.fillna(test.mean(), ir
# count the number of NaN v
print(test.isnull().sum())
```

```
Store_Dept             0
Weekly_Sales           0
Date                   0
IsHoliday              0
Temperature            0
Fuel_Price             0
MarkDown1              0
MarkDown2              0
MarkDown3              0
MarkDown4              0
MarkDown5              0
CPI                    0
Unemployment           0
Type                   0
Size                   0
Year                   0
Month                  0
Week                   0
Day                    0
n_days                 0
Sales_ratio_CPI        0
Sales_ratio_Unemploy   0
Sales_ratio_Tem        0
Sales_ratio_Fuel       0
dtype: int64
```



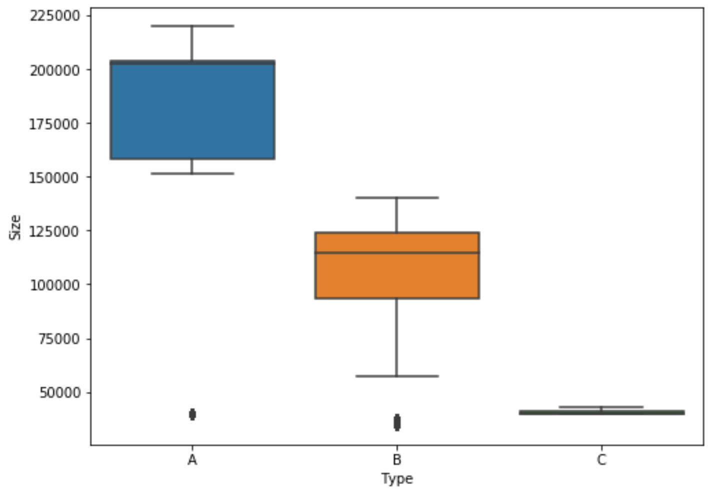
```
#Handling missing values
# fill missing values with
train.fillna(train.mean(),
# count the number of NaN
print(train.isnull().sum()
```

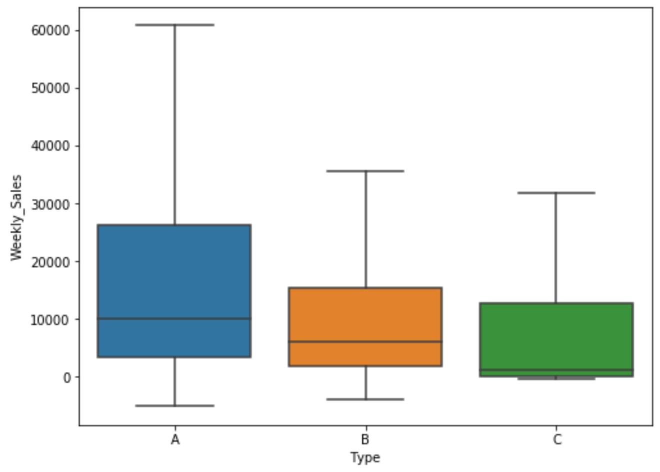
```
Store_Dept             0
Date                   0
Weekly_Sales           0
IsHoliday              0
Temperature            0
Fuel_Price             0
MarkDown1              0
MarkDown2              0
MarkDown3              0
MarkDown4              0
MarkDown5              0
CPI                    0
Unemployment           0
Type                   0
Size                   0
Year                   0
Month                  0
Week                   0
Day                    0
n_days                 0
Sales_ratio_CPI        0
Sales_ratio_Unemploy   0
Sales_ratio_Tem        0
Sales_ratio_Fuel       0
```

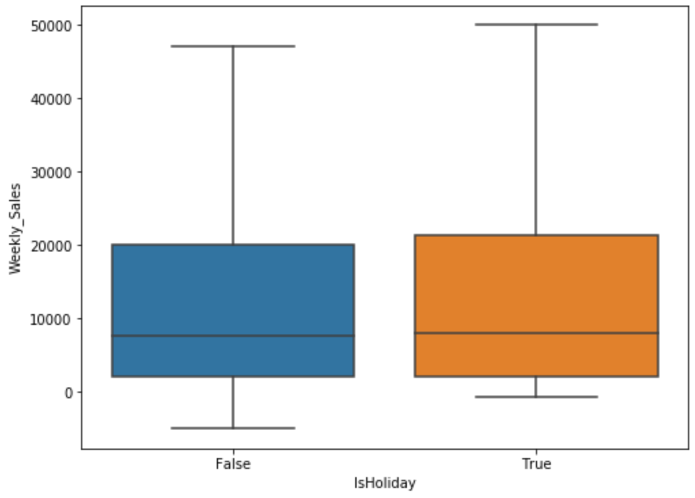


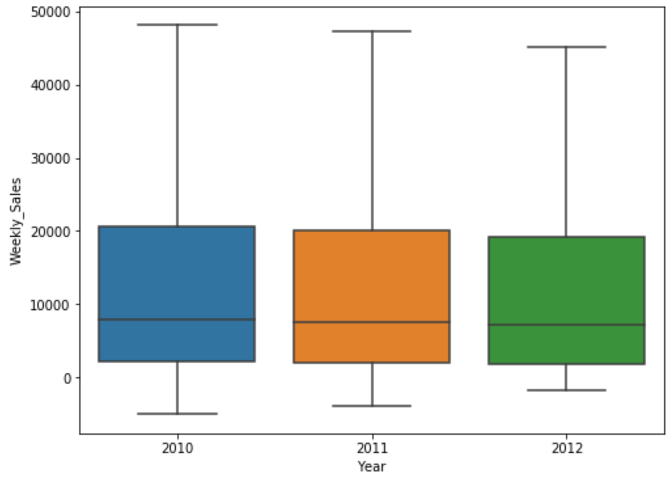
```
class MultiColumnLabelEncoder:
    def __init__(self,columns = None):
        self.columns = columns # array of column names to encode

    def fit(self,X,y=None):
        return self # not relevant here

    def transform(self,X):
        '''
        Transforms columns of X specified in self.columns using
        LabelEncoder(). If no columns specified, transforms all
        columns in X.
        '''
        output = X.copy()
        if self.columns is not None:
            for col in self.columns:
                output[col] = preprocessing.LabelEncoder().fit_transform(output[col])
        else:
            for colname,col in output.iteritems():
                output[colname] = preprocessing.LabelEncoder().fit_transform(col)
        return output

    def fit_transform(self,X,y=None):
        return self.fit(X,y).transform(X)
```

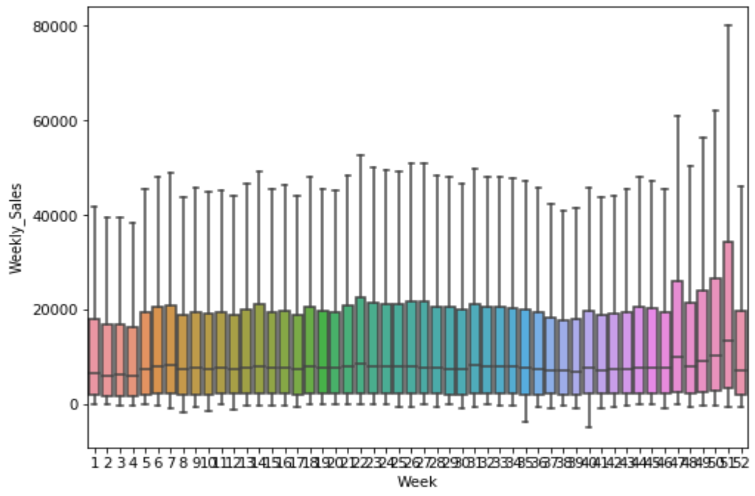
As there are several plots that I have plotted to understand the distribution of variables I will introduce here only the most significant and compact ones to keep the straightforward approach. Please refer below to grasp overview of data at glance. As it is visible I plotted distribution individually and also as the relationship among different variables as well.

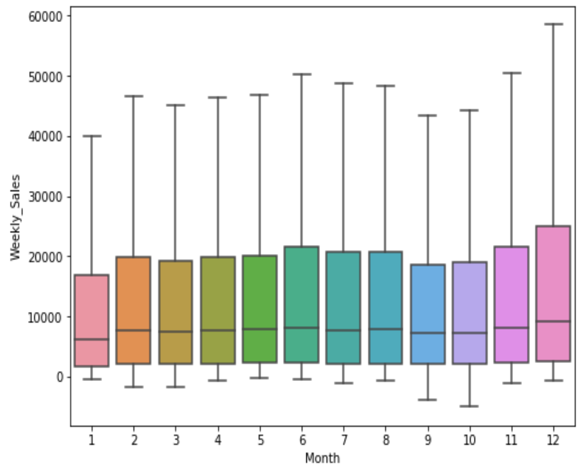

225000
200000
175000
150000
125000
100000
75000
50000
Size
A
B
C
Type

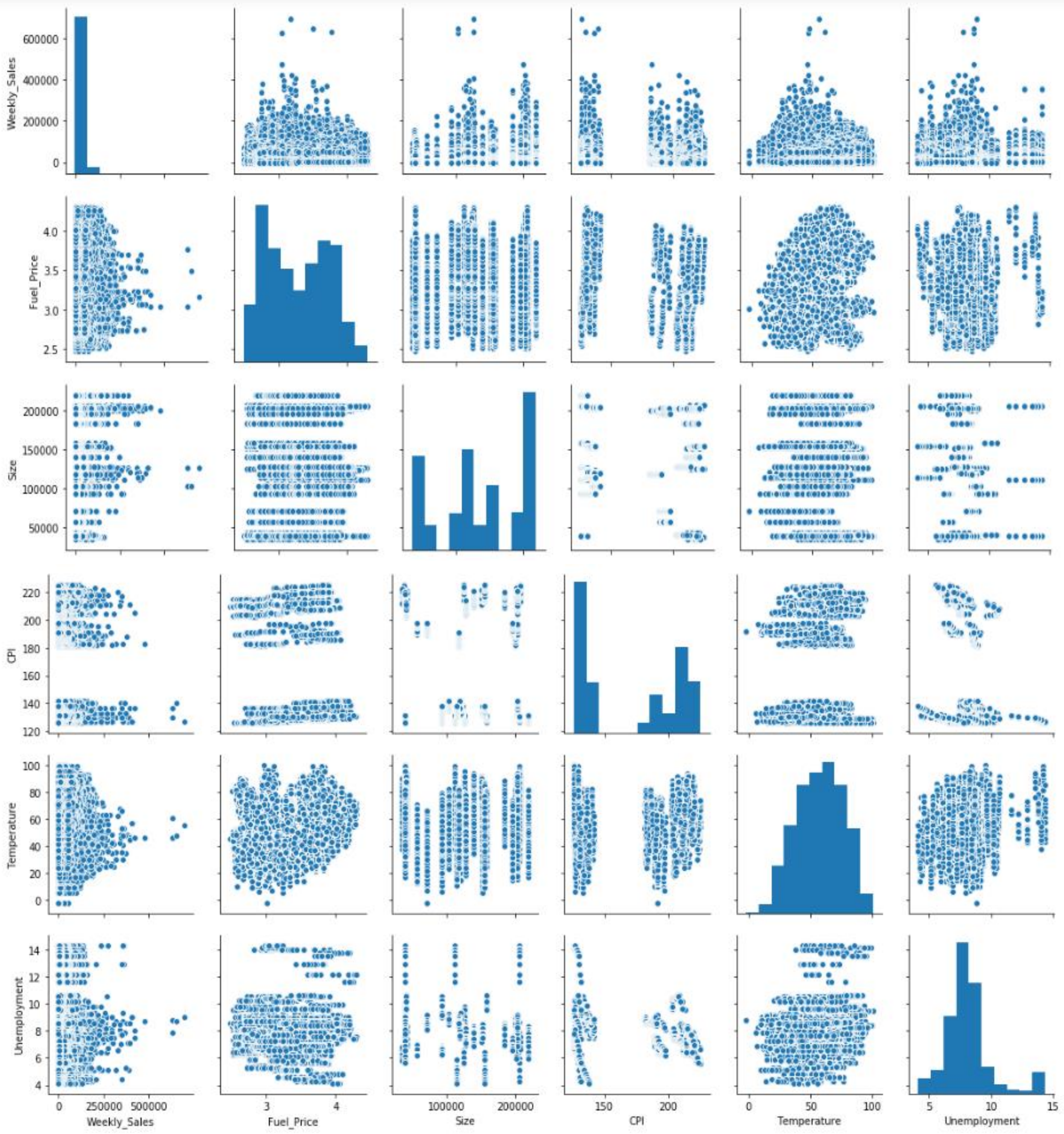

60000
50000
40000
30000
20000
10000
0
Weekly_Sales
A
B
C
Type


50000
40000
30000
20000
10000
0
Weekly_Sales
False
True
IsHoliday

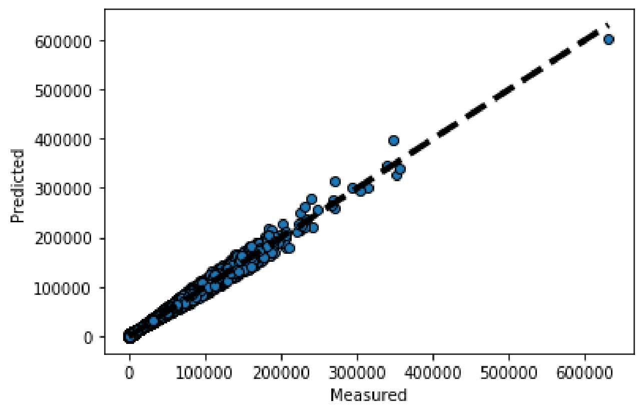

50000
40000
30000
20000
10000
0
Weekly_Sales
2010
2011
2012
Year

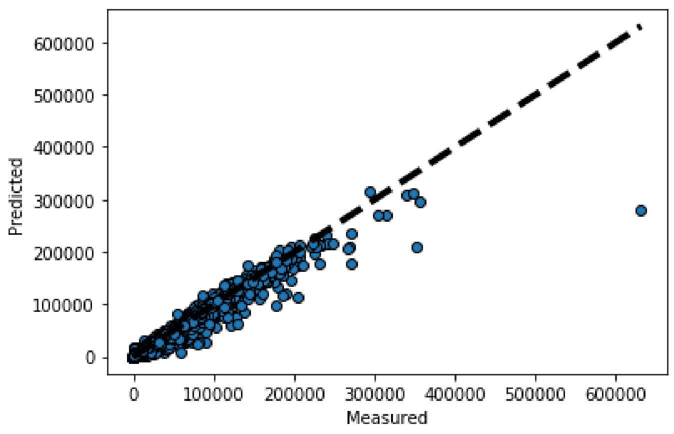

80000
60000
40000
20000
0
Weekly_Sales
Week


60000
50000
40000
30000
20000
10000
0
Weekly_Sales
1
2
3
4
5
6
7
8
9
10
11
12
Month

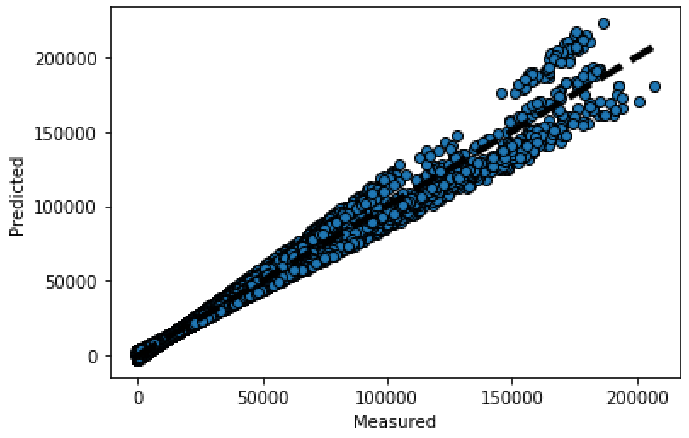


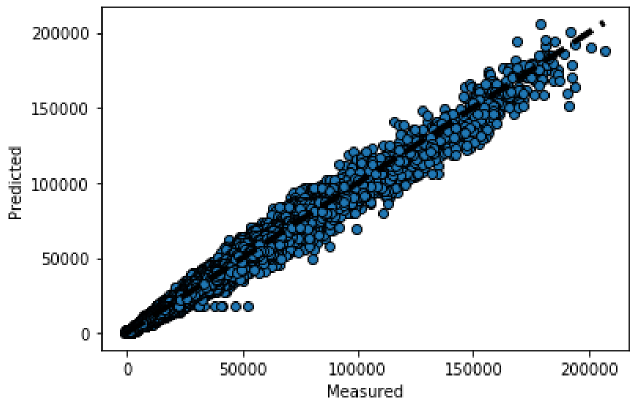
In the first part of our analysis we are asked to conduct Regression analysis with respect to target variable, in this case we are using Weekly_Sales variable here as a dependent variable. We implemented two kinds of models such as Linear Regression and KNN regression analysis. I could say looking at MSE and R Square we can derive some conclusions here as Linear regression has better explanation power than KNN Regression. As both model performed well enough with very high score of R square, but Linear Regression R square value was 0.01 more than KNN Regression. However, while Considering MSE, KNN Regression had considerably lower error value than Linear regression. The left is Linear regression and the right one is KNN Regression. Both of the charts are plotted for cross validated predictions.



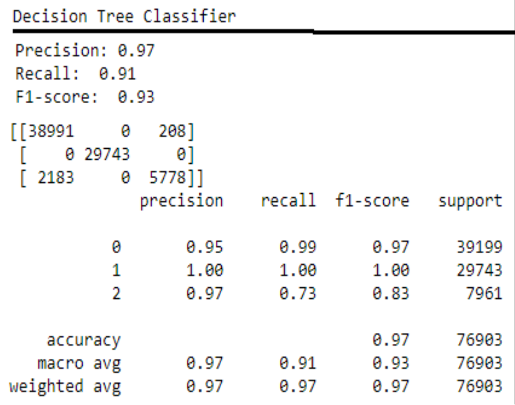


Now let’s visually plot residuals from predicted test data to see how our model is performing. Again left one is for Linear regression and the right chart for KNN regression.

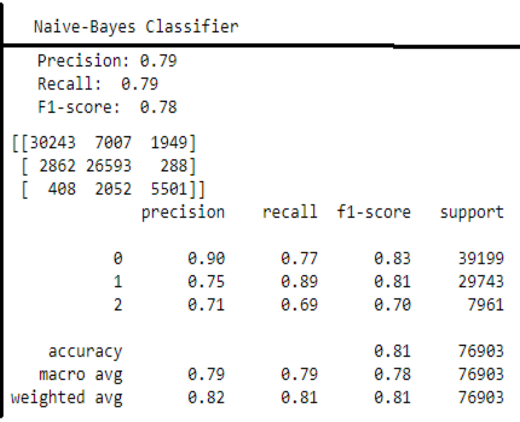




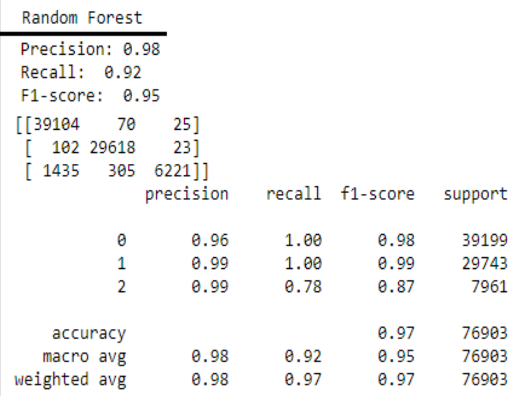
Regarding Classification problem which was our Part 2 Analysis, Naïve Bayes, Decision trees and Random Forest were implemented to conduct our analysis. Our target variable was here Type of the store, and we could easily define that it is multi class problem as our target is A, B and C types of stores. From our analysis it was clear that both decision tree and random forest performed clearly similarly with much considerable accuracy than Naïve Bayes Classification. Rather than using correlation matrix, I handcrafted a function which will automatically derive feature importance for our model and from where we can see which variables are more significant than others. For that I used Random Forest, Decision Tree, ExtratreeClassfier. I did not find necessary to rerun the model as by itself the results were pretty impressive.



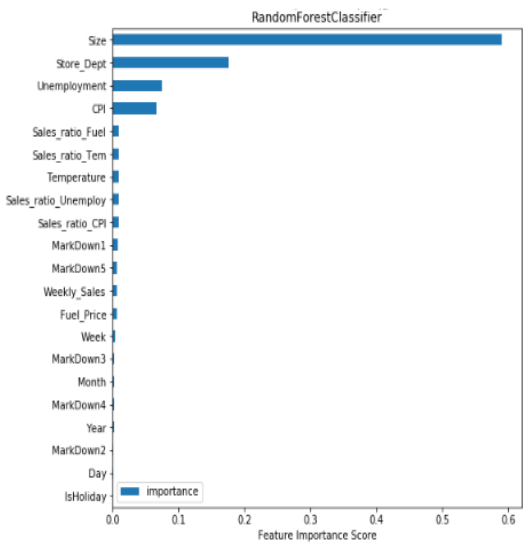
```
Decision Tree Classifier
Precision: 0.97
Recall:  0.91
F1-score:  0.93
[[38991     0   208]
 [    0 29743     0]
 [ 2183     0  5778]]
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     39199
           1       1.00      1.00      1.00     29743
           2       0.97      0.73      0.83      7961

    accuracy                           0.97     76903
   macro avg       0.97      0.91      0.93     76903
weighted avg       0.97      0.97      0.97     76903
```

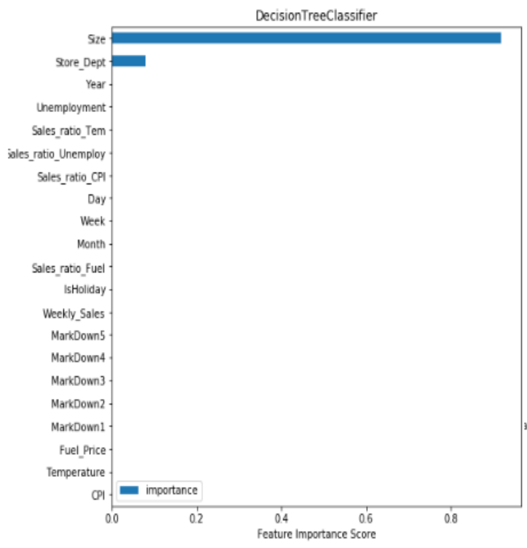


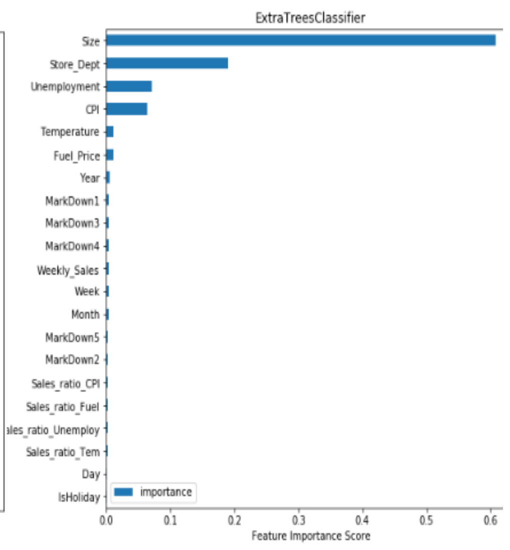
```
Naive-Bayes Classifier
Precision: 0.79
Recall:  0.79
F1-score:  0.78
[[30243  7007  1949]
 [ 2862 26593   288]
 [  408  2052  5501]]
              precision    recall  f1-score   support

           0       0.90      0.77      0.83     39199
           1       0.75      0.89      0.81     29743
           2       0.71      0.69      0.70      7961

    accuracy                           0.81     76903
   macro avg       0.79      0.79      0.78     76903
weighted avg       0.82      0.81      0.81     76903
```



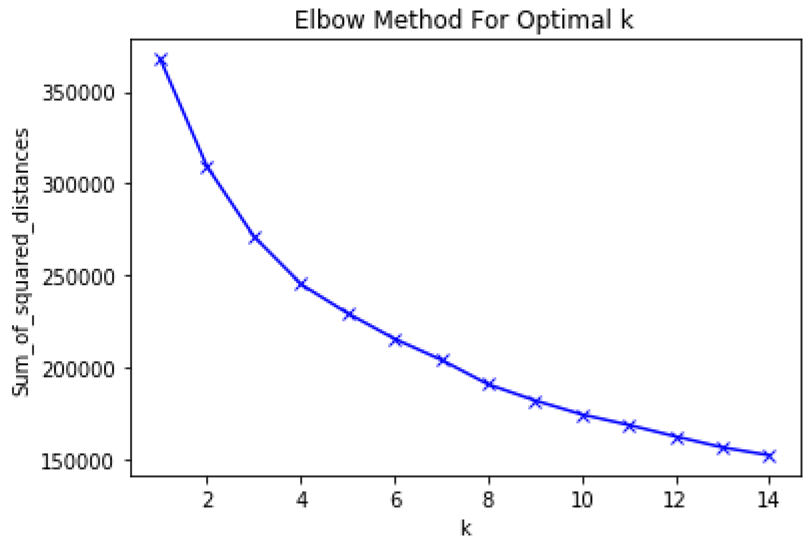
```
Random Forest
Precision: 0.98
Recall:  0.92
F1-score:  0.95
[[39104    70    25]
 [  102 29618    23]
 [ 1435   305  6221]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     39199
           1       0.99      1.00      0.99     29743
           2       0.99      0.78      0.87      7961

    accuracy                           0.97     76903
   macro avg       0.98      0.92      0.95     76903
weighted avg       0.98      0.97      0.97     76903
```







In the last section we used clustering for our classification to somehow segment Walmart stores. We were asked to choose applicable value of K that is number of cluster to conduct K-means clustering. To identify the most optimal k value, we used pre-defined function which will plot relationship of k and Sum of Squared distances. Here, will refer to a very good concept from Economics which is called Marginal Diminishing Returns. It means that we should find optimal value of the K which will produce less significant change while adding additional K value. Here, I came up with K value of 7 which I think is the optimal value.


Elbow Method For Optimal k
Sum_of_squared_distances
k