# Regression - Retail Data Analysis

## I. Problem Statement

Sales forecasting is one important aspect for the management of retail stores. By pulling data from Kaggle, we have obtained three historical sales data, which covers 2010-02-05 to 2012-11-01. Given multiple numerical and categorical features, our analysis will be selecting a best model for predicting the weekly sales of retail stores. The goal of this project is to predict the weekly sales as accurately as possible, and we will be choosing RMSE as our key performance metric, and finally, choose the best prediction model. By looking at our trained models, we could get a better sense of what features are of more importance in determining the sales performance of a retail store, and we could use those results to make suggestions on what good action or strategies retail stores could use to increase sales.

## II. Data Preparation & Exploratory Data Analysis

Our three datasets are all historical sales data and contain information for 45 stores located in different regions, and each store has several departments. Let's first look at the raw data sets and see how we could utilize and combine information from all these three datasets.

Features Data:

The first dataset is called feature_df, which contains 8190 rows and 12 columns related to the store, department, promotional markdowns, and other features for the given dates. By checking the null values, we noticed that CPI and Unemployment columns contain 585 missing values, approximately 7%, which is not a very big number compared to the data size. We could easily use imputation or just drop these rows for handling this issue, but we will go back to this solution and explanation soon with our combined data. On the other hand, we are losing a large chunk of MarkDown1 - MarkDown5 data. As described on Kaggle, those columns measure several promotional markdown events throughout the year, such as the Super Bowl, Labor Day, Thanksgiving, and Christmas, but they are only available after November 2011. This brings us the question of how we should handle large amounts of missing data, and same as CPI and Unemployment we will explain this later in the report after we merge the three datasets. Also, we have checked that there are 45 stores in total.

| | Store | Date | Temperature | Fuel_Price | MarkDown1 | MarkDown2 | MarkDown3 | MarkDown4 | MarkDown5 | CPI | Unemployment | IsHoliday |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2010-05-02 | 42.31 | 2.572 | NaN | NaN | NaN | NaN | NaN | 211.096358 | 8.106 | False |
| **1** | 1 | 2010-12-02 | 38.51 | 2.548 | NaN | NaN | NaN | NaN | NaN | 211.242170 | 8.106 | True |
| **2** | 1 | 2010-02-19 | 39.93 | 2.514 | NaN | NaN | NaN | NaN | NaN | 211.289143 | 8.106 | False |
| **3** | 1 | 2010-02-26 | 46.63 | 2.561 | NaN | NaN | NaN | NaN | NaN | 211.319643 | 8.106 | False |
| **4** | 1 | 2010-05-03 | 46.50 | 2.625 | NaN | NaN | NaN | NaN | NaN | 211.350143 | 8.106 | False |

Sales Data:

The second dataset is sales_df with 421570 rows and 5 columns. This includes our target column - Weekly_Sales and 4 other features, i.e., store number, department number, date, and a boolean field describing if it is a holiday or not. There is no missing data in this dataset. Again, we checked that the store number is 45, and there are 81 unique departments for all those stores. Then, by looking at the descriptive statistics, we noticed that the mean of weekly sales is 15981, and its standard deviation is 22711, which indicates that there is a large volatility in weekly sales.

| | Store | Dept | Date | Weekly_Sales | IsHoliday |
|---|---|---|---|---|---|
| **0** | 1 | 1 | 2010-05-02 | 24924.50 | False |
| **1** | 1 | 1 | 2010-12-02 | 46039.49 | True |
| **2** | 1 | 1 | 2010-02-19 | 41595.55 | False |
| **3** | 1 | 1 | 2010-02-26 | 19403.54 | False |
| **4** | 1 | 1 | 2010-05-03 | 21827.90 | False |

Stores Data:

Our third dataset is Stores_df, which gives us information about the store type and each store's size. It contains 45 rows and 3 columns. There is also no missing value in this dataset. By using the groupby aggregation method in Pandas, there are more type A stores overall and type A stores also have the largest average size.

| | Store | Type | Size |
|---|---|---|---|
| **0** | 1 | A | 151315 |
| **1** | 2 | A | 202307 |
| **2** | 3 | B | 37392 |
| **3** | 4 | A | 205863 |
| **4** | 5 | B | 34875 |

Merge Three Datasets:

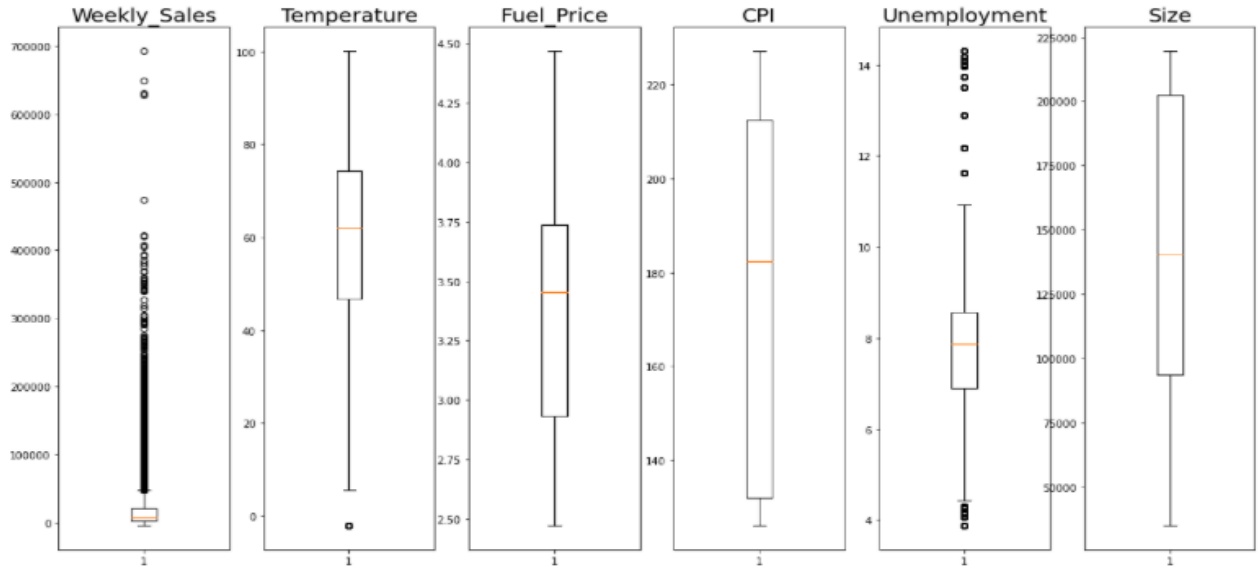
Now, with all these three datasets being verified and ready, we decided to merge the three data together. One common field for all these three data is the Store column. Hence, we first inner join the sales_df with feature_df on shared columns i.e. Store, Date, and IsHoliday, then we used inner join again to combine store_df to this. After the merging, our new dataset retail_df has 421570 rows and 16 fields.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 421570 entries, 0 to 421569
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  datetime64[ns]
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool
 5   Temperature   421570 non-null  float64
 6   Fuel_Price    421570 non-null  float64
 7   MarkDown1     150681 non-null  float64
 8   MarkDown2     111248 non-null  float64
 9   MarkDown3     137091 non-null  float64
 10  MarkDown4     134967 non-null  float64
 11  MarkDown5     151432 non-null  float64
 12  CPI           421570 non-null  float64
 13  Unemployment  421570 non-null  float64
 14  Type          421570 non-null  object
 15  Size          421570 non-null  int64
dtypes: bool(1), datetime64[ns](1), float64(10), int64(3), object(1)
memory usage: 51.9+ MB
```

Data Cleaning Pipeline & Handling Missing Values:

There are no missing values for most columns, but MarkDown1-5 columns all contain more than 270000 null values. To deal with this, we first defined a function - missing_col() that will return a list of column names for columns containing missing values. As expected, it returns ['MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5'] this specific data retail_df. The way how we handled this is to remove the column if it has more than 0.5 missing values. On the other hand, if the missing values are fewer than half, we choose to use median imputation.

Next, we used boxplots to see if there are outliers in some numerical field that would potentially hurt our following prediction. The graph is shown below:




From the graph, we can see that Weekly_Sales and Unemployment columns seem to have large outliers. There are many commonly used ways of handling outliers. For example, we could

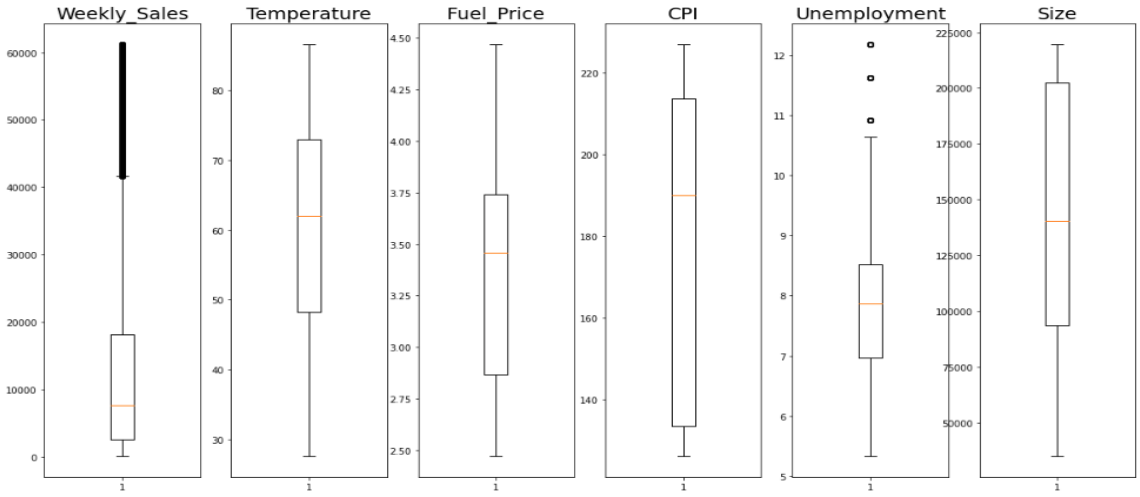
include data with only three standard deviations, or we could keep data only between Q1-1.5 IQR and Q3+1.5 IQR. But here, we decided to drop values below 0.05 quantile or above 0.95 quantile.

Besides, for categorical columns that contain limited unique values, we wanted to change the “object” data type to the “category” data type in order to save memory spaces.

After defining all these cleaning functions, we used a pipeline to do all cleaning steps simultaneously and returned a new dataframe - “retail”. This pipeline is quoted from https://towardsdatascience.com/cleaner-data-analysis-with-pandas-using-pipes-4d73770fbf3c And we modified and edited this pipeline according to what we need for this data. We also extracted another four columns - Week_of_Year, Month_of_Year, Year, and Day.

| | Store | Dept | Date | Weekly_Sales | IsHoliday | Temperature | Fuel_Price | CPI | Unemployment | Type | Size | Week_of_Year | Month_of_Year | Year | Day |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 2010-05-02 | 24924.50 | False | 42.31 | 2.572 | 211.096358 | 8.106 | A | 151315 | 17 | 5 | 2010 | 2 |
| **1** | 1 | 2 | 2010-05-02 | 50605.27 | False | 42.31 | 2.572 | 211.096358 | 8.106 | A | 151315 | 17 | 5 | 2010 | 2 |
| **2** | 1 | 3 | 2010-05-02 | 13740.12 | False | 42.31 | 2.572 | 211.096358 | 8.106 | A | 151315 | 17 | 5 | 2010 | 2 |
| **3** | 1 | 4 | 2010-05-02 | 39954.04 | False | 42.31 | 2.572 | 211.096358 | 8.106 | A | 151315 | 17 | 5 | 2010 | 2 |
| **4** | 1 | 5 | 2010-05-02 | 32229.38 | False | 42.31 | 2.572 | 211.096358 | 8.106 | A | 151315 | 17 | 5 | 2010 | 2 |

Now, let’s look at the boxplots again and see how well our function worked in terms of removing outliers:




On top of that, it turned out that this step removed the MarkDown columns because they have more than 50% NaN values. This step makes sense when the data is really missing or not measurable. However, to ensure the quality of our prediction, we decided to create two separate sections, one uses the above retail data without MarkDown from 2010 February to 2012 November, and the other one uses complete data with MarkDown values after 2011 November.

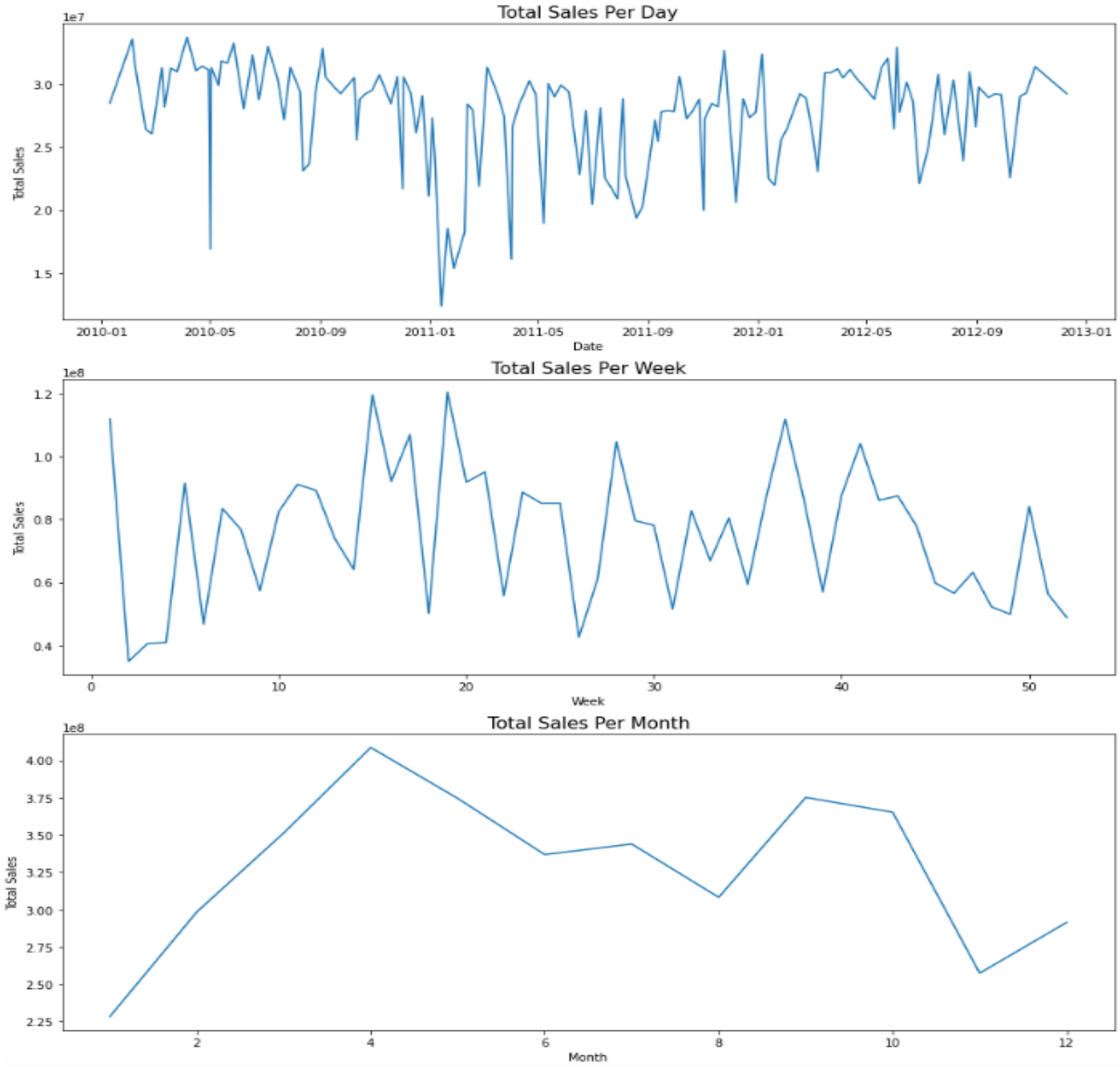
We will train and test our models separately on these two datasets with and without MarkDown values, and we will discuss this again later in this paper.

# III. Exploratory Data Analysis

With our new dataframe ready, we are going to create visualizations to display characteristics and patterns in the data.

## 1). Total Sales Grouped by Date, Week, and Month:

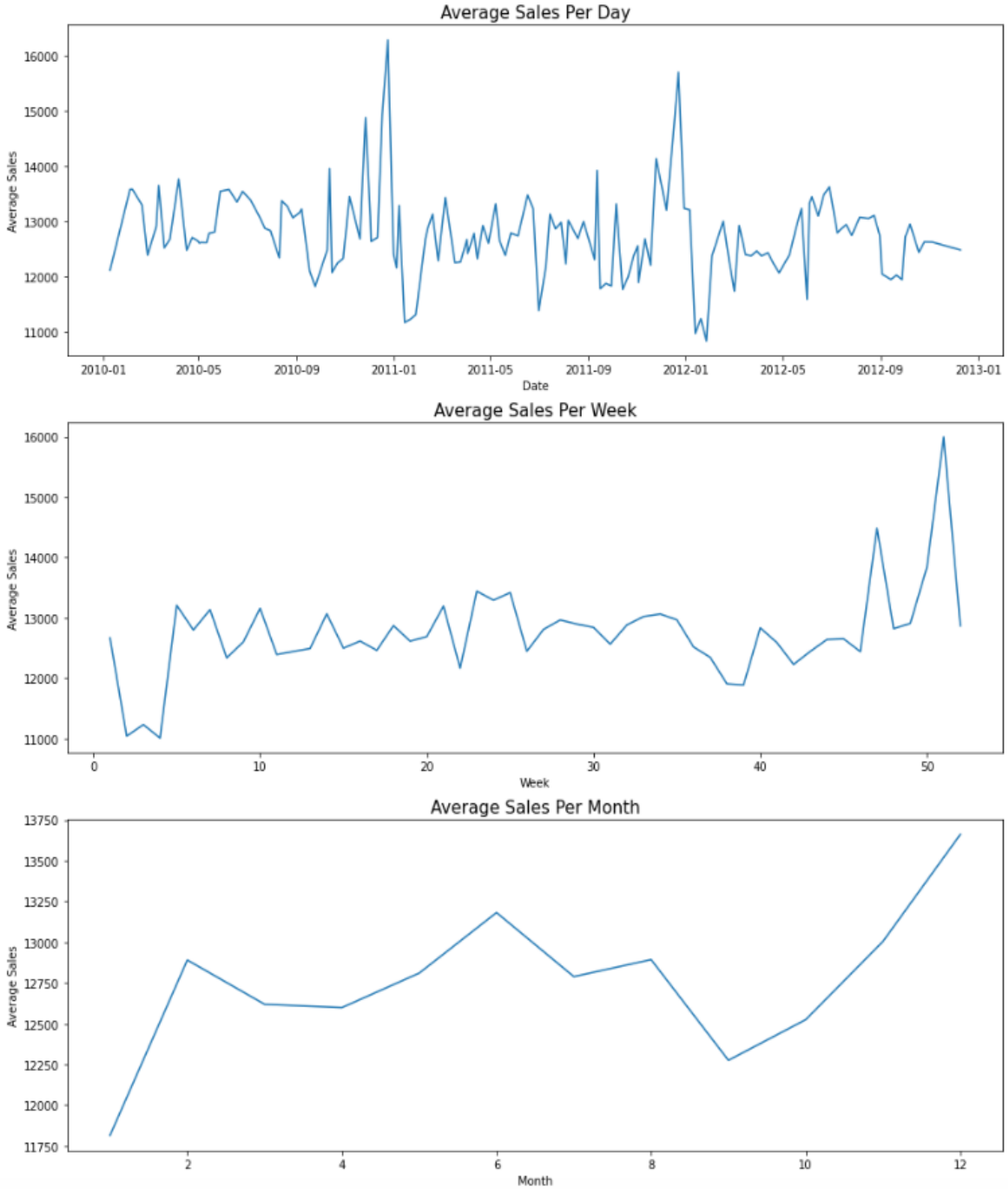
Firstly, the plots of total sales aggregated by Date, Week_of_Year and Month_of_Year respectively are shown below:



By looking at the first time series plot, we found that no clear trend exists, but it shows some seasonality occurring. As what we guessed from the standard deviation of Weekly_Sales, this time series does reveal fluctuation and volatility.

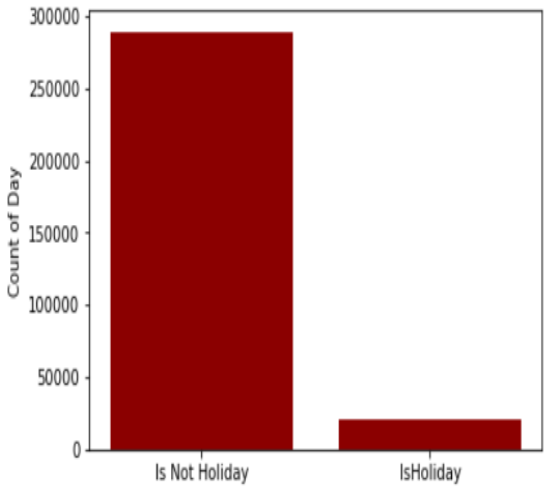
## 2). Average Sales Grouped by Date, Week, and Month:

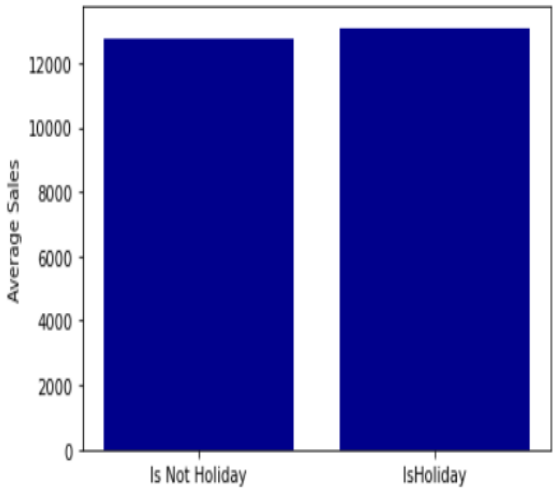
Similarly, we could show the average sales aggregate by Data, Week, and Month as well:



## 3). Holiday:

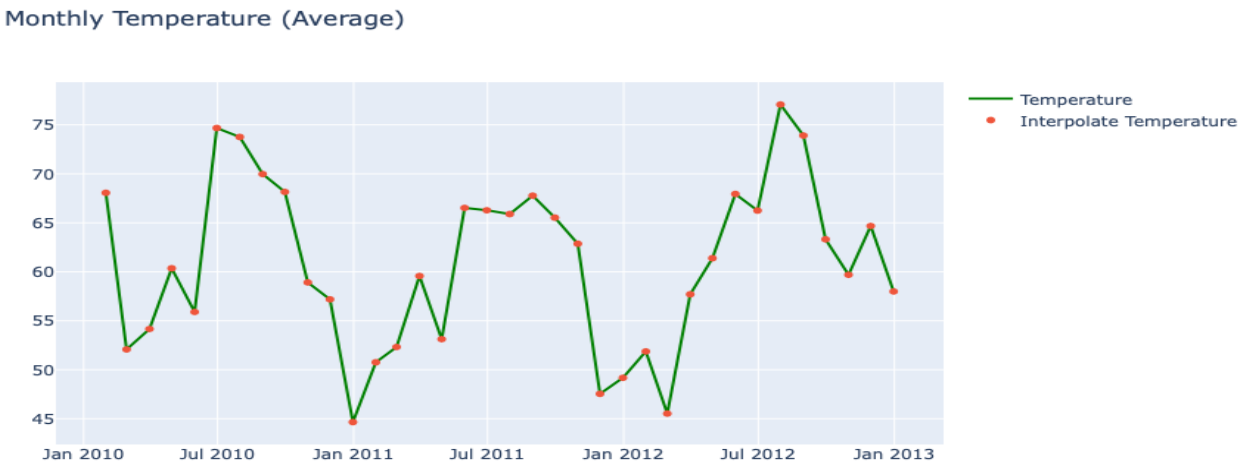
Next, we are going to use bar charts to compare categorical data. The below first figure displays the count of days of holidays and non-holidays, and the second figure displays the average sales comparison for holidays and non-holidays. We could observe that there are a lot more days that are not holidays, which makes total sense, but the average sales are pretty close for holidays and non-holidays. The average sale for holidays is a little bit higher than the one for non-holidays.



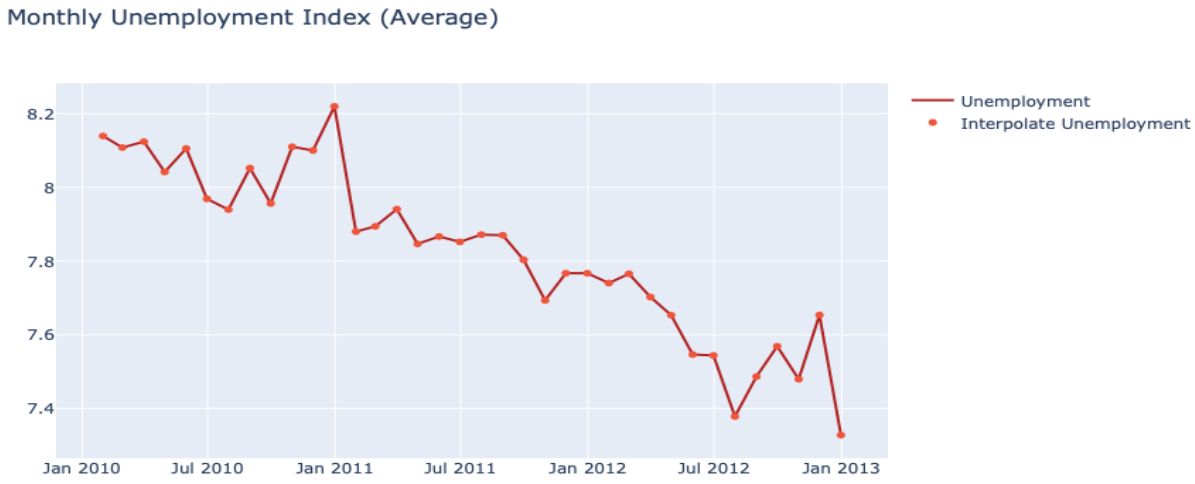

The below graphs are quoted from Kaggle Feri Haldi Tanjung's work. Details could be found here: https://www.kaggle.com/ferihalditanjung/retail-analytic-did-sales-increased-on-x-mas

## 4). Monthly Average Temperature:



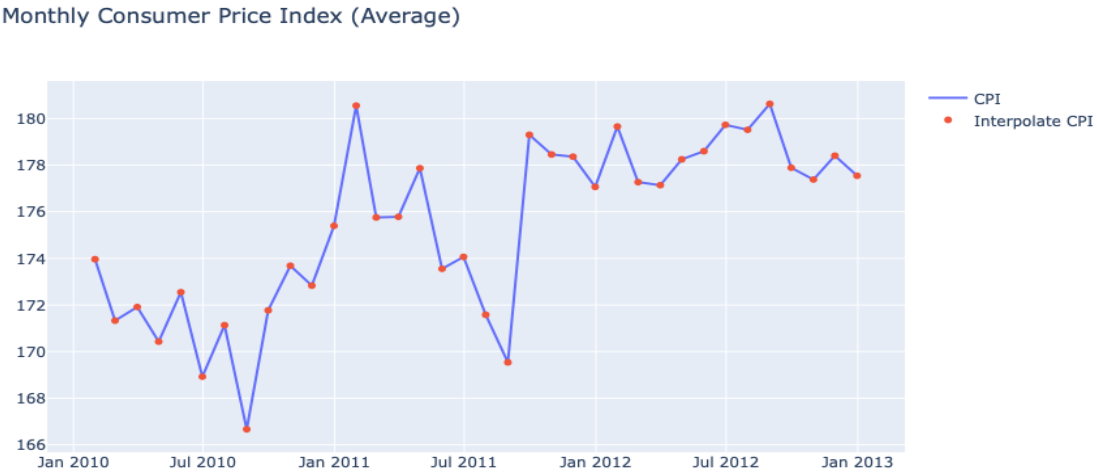

It makes sense that monthly average temperatures are high during summer and low during winter. We are guessing that these temperature patterns will not affect retail sales greatly.

## 5). Monthly Average Unemployment Rate:



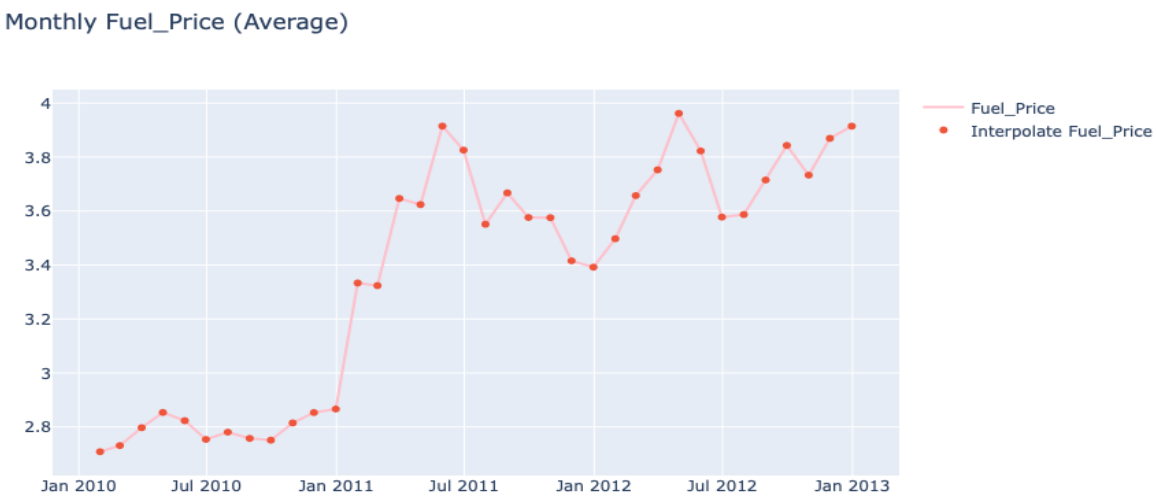

The above graph shows that the average unemployment rate is in general decreasing across time. There are two spikes around December in 2011 and 2013, which tends to be cyclical. This also implies that the economy is expanding, so the unemployment rate is dropping as evidence of economic growth. We are wondering if the decreasing unemployment rate brings more people being able to spend money in retail stores, hence increasing sales.

## 6). Monthly Average CPI:



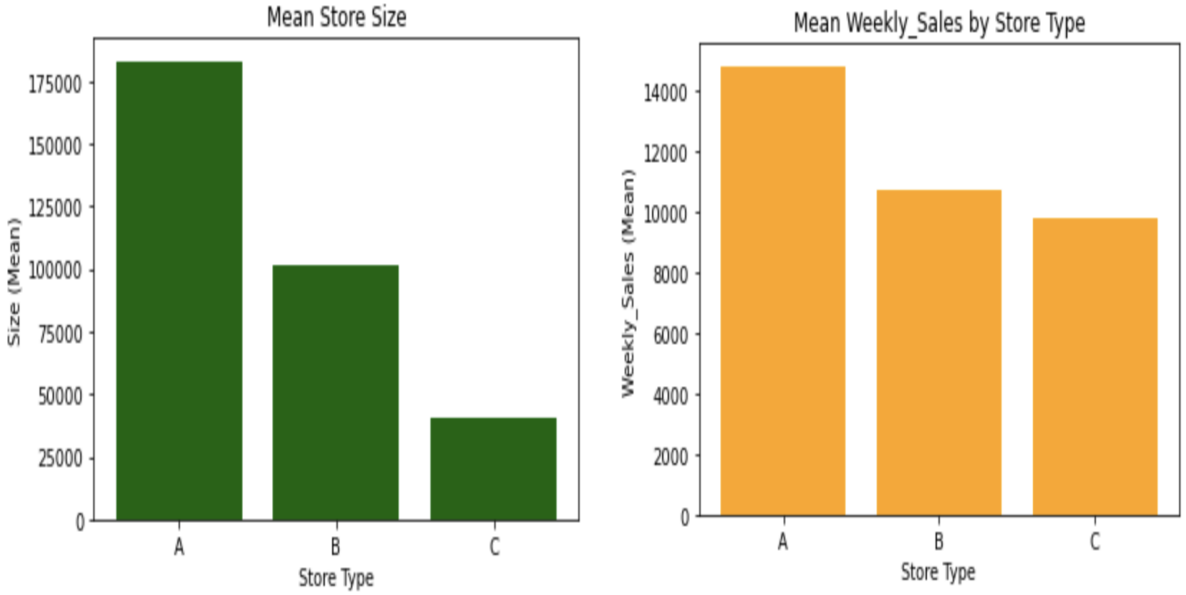

Overall, the average CPI is increasing, which implies an increasing inflation rate. Also, there was a spike occurring around January 2011. With inflation, the cost of goods and services is going up as well, which could force retail stores to increase their prices and could impact their sales too.

## 7). Monthly Average Fuel Price:



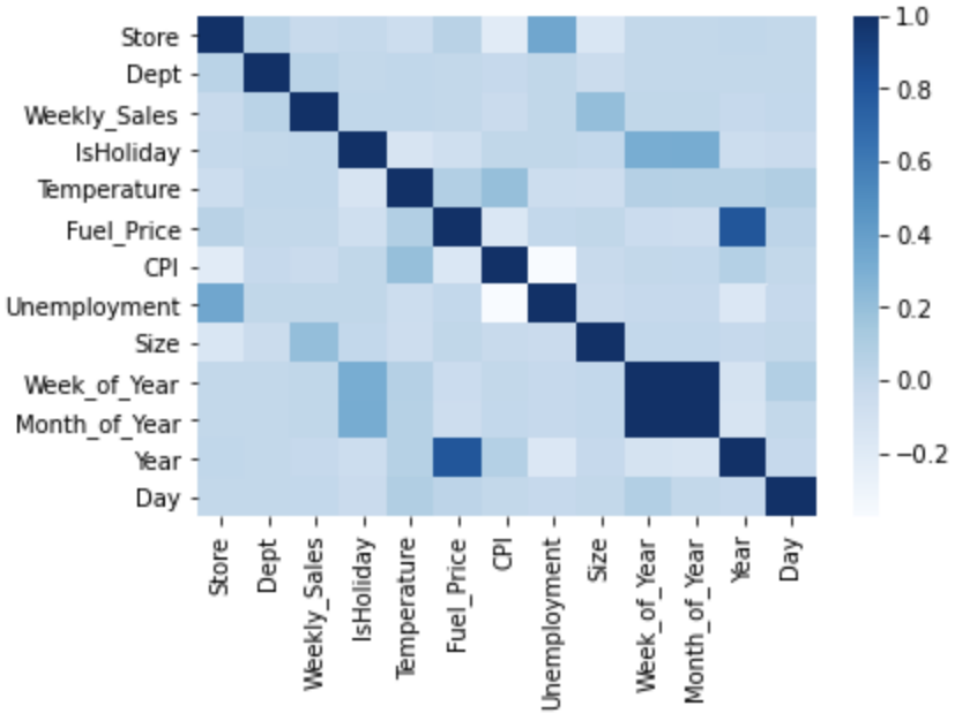

It seems that the average fuel price keeps increasing from 2010 to 2013. The reason for this might be that Gasoline prices tend to grow when the available supply of Gasoline decreases. But we don't think there is a direct correlation between fuel prices and retail sales.

## 8). Store Type:




As we have explored already, Type A stores have the highest mean store size, which also contributes to the largest amount of average sales as well. Therefore, there is a clear relationship between store type and sales, or between size and sales, i.e., Type A stores may generate higher sales, or large-size stores could generate higher sales.

9). Pairwise Relationship:




The above heatmap gives us a visualization of correlations between each two variables. It looks like Weekly_Sales is most positively related to the size of stores, but no other obvious correlations between Weekly_Sales and other variables. Note that Pearson's correlation coefficient only represents the linear relationship between two fields, there might be other relationships between variables that we could only find out using different methods.

## IV. Prediction Models

Now with data ready and explored, we can move onto our next stage - use machine learning models to make predictions. The models that we are selecting for this project will be Linear Regression, KNN, Decision Trees, Random Forests, and another ensemble learning model using Voting Regressor. We will train our training data using each of these methods, and at the end we will use RMSE to choose the best model. As we mentioned earlier, we are going to do similarly, but using two datasets (with MarkDown and without MarkDown information). There are things we wanted to explain before our prediction:

About Standardization:

Most models that we have chosen do not require standardization, such as Linear Regression, Decision Trees, and Random Forests. However, we will need to standardize for the KNN model, as standardization matters when calculating Euclidean distance. Therefore, we chose normalization, which also allows us to move the points centered at zero. Here, we standardize the whole dataset, while in practice, scaling only using the training set will be better.

About Principal Components Analysis (PCA):
Before jumping into our regression, we also wanted to first specify our decision on not using PCA for these models. From III 9), we have shown that the variables are not strongly correlated, hence PCA will not be really effective in reducing the dimension of data. Also, we would like to look at each variable's effect on Weekly_Sales separately, so we don't want to transform them to new components, which would be hard to interpret.

Performance Evaluation Metrics:
The metrics that we chose are RMSE, MAE, testing set R^2. For comparing models, we will be using RMSE as the key metric. R^2 is great for measuring how much variation is explained by our predicting models, but RMSE or MAE for quantifying the size of our prediction errors. However, we will use R^2 to compare the training and testing set performances, and to see if the model is overfitting.

## 1). Data Without MarkDown columns:

Since most markdowns are missing (before November 2011), we could first consider dropping the markdown columns for this section.

### 1.1 Select Features & Standardization:

Numerical features we selected are Store, Dept, Temperature, Fuel_Price, CPI, Unemployment, Size, Day, Year. Notice that Day, and Year columns are extracted in our earlier part, which are more doable than using the Date column. We also chose the boolean variable IsHoliday and a categorical variable Type and used the get_dummies function to create dummies for the Type feature. Lastly, our target will be the Weekly_Sales column.

We also use a ColumnTransformer function to standardize all the numerical features and leave the Boolean variables unchanged.

### 1.2 Train Test Split:

We will split the full dataset into 60% training data and 40% testing data. We are going to use the training set to train models, then use the testing set to evaluate the performance of models. We decided not to use cross validation for this because models like Decision Trees, Random Forests take longer time to train and cross validation will make the process really slow. However, we will use cross validation later for only on training data when tuning hyperparameters. This is equivalent to splitting the training set again into a "real" training set and a validation set. In addition, by creating two separate sets, the testing set will be treated as the new or unseen data that represents a robust set that could be used to evaluate our training models.

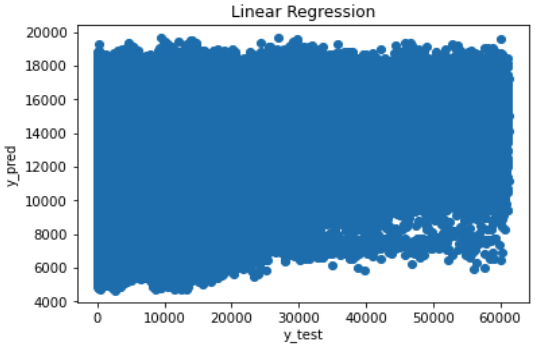
### 1.3 Train Models and Evaluate Performances:

#### *<i> Linear Regression:*

Training our data using linear regression, the results of coefficients and intercept is shown below. Linear regression produces RMSE = 13415.21, MAE = 10321.18, training accuracy

equals 4.97%, and testing accuracy equals 4.83%. We stored those values in a dictionary so that we can compare with other models for each measurement. By using scatterplots, we could see our prediction and the actual values are not very close to each other. Thus, a linear function might not be a good measurement of relationships between our features and target.

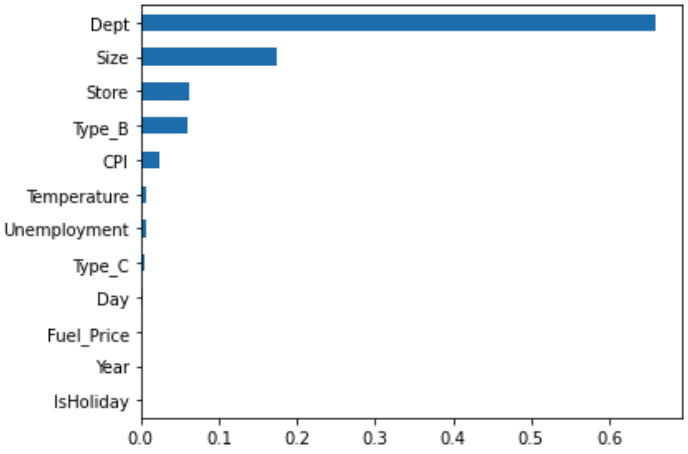
```
intercept   12321.086481323595
       Predictor   coefficient
0          Store   -275.332836
1           Dept    608.624792
2    Temperature    391.788209
3     Fuel_Price    124.866193
4            CPI   -486.873631
5   Unemployment    102.427654
6           Size   3396.556169
7            Day    -25.905855
8           Year   -219.199002
9      IsHoliday    591.254452
10        Type_B    425.658221
11        Type_C   2901.525575
```



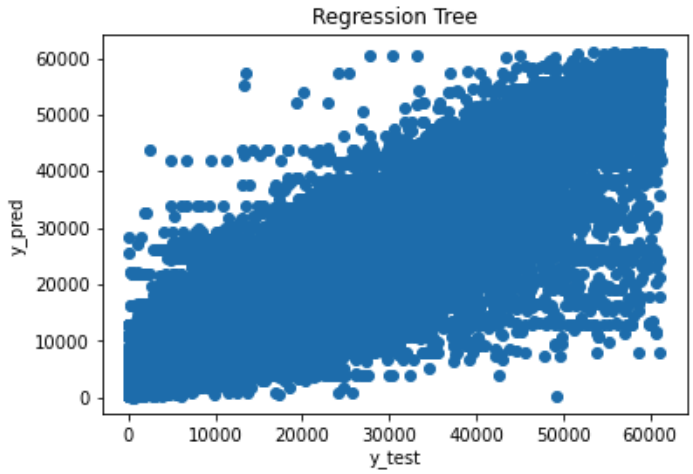

The above regression model uses all possible features. To modify our model, we tried it again with backward elimination. Backward elimination only removes the Day column, and it generates the results really similar to and not better than our previous model. Thus, we will still use our previous linear regression model. This again indicates that linear regression is not an appropriate model even with backward elimination.

### *<ii> Regression Tree Model:*

Next, we tried with a decision tree regressor. One crucial step for this is the hyperparameter tuning. A grid search method with 5-fold cross validation is used to avoid overfitting and gives us 12 max_depth is optimal. One good characteristic about decision tree models is that it is doing a feature selection automatically by comparing feature importance and provides us with information about what features are most critical in determining the retail sale. From the result below, we could easily observe that features like department, size, store, type_b, CPI are more important in determining the sales.

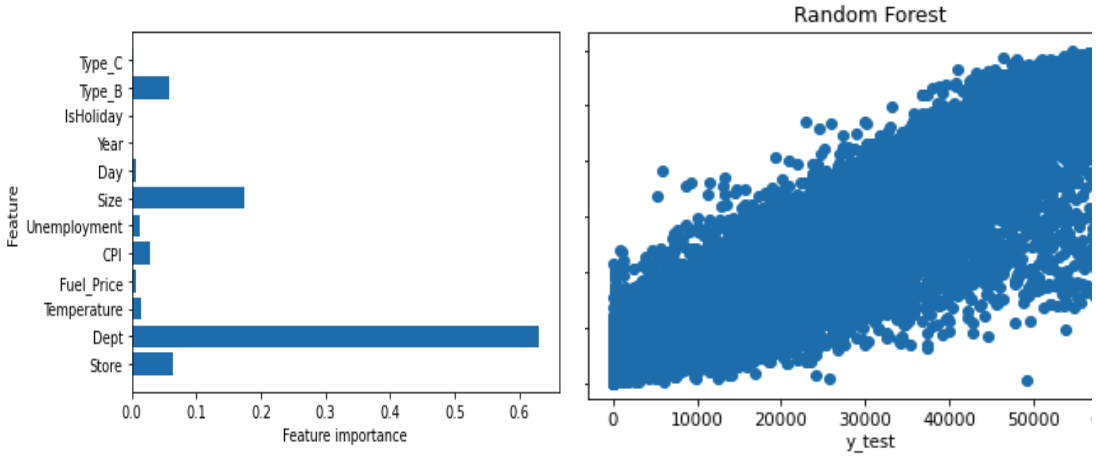


The prediction result shows that RMSE = 4529.143, MAE = 2589.040, training score = 90.16%, and testing score = 89.15%. This result is so much better compared to Linear Regression. Advantages of a regression tree model also include that it is intuitive and easy to explain and interpret. By using grid search, we also properly handle the potential overfitting problem for most tree models. One drawback is that it is relatively expensive and took us more time because our data is large. Overall, it produces a satisfying prediction result, and the predicting values and actual values are pretty close according to the scatter plot.



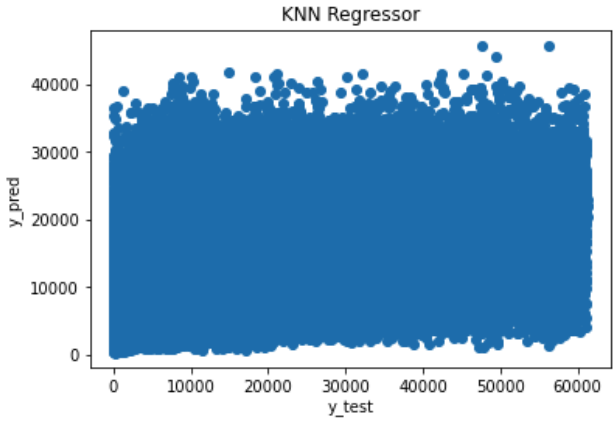

***<iii> Random Forest:***

On top of our decision tree model, random forest is one ensemble method that is more robust and versatile than trees in the sense that they aggregate many trees together and yield better results. Again, we have shown the feature importance for our random forest. The graph also visualizes what features contribute more, but features look more diversified than trees. It turns out that random forest generates a better result as we expected. We have RMSE = 3323.43, MAE = 1732.74, training accuracy = 96.1%, and testing accuracy = 94.2%, which are indeed better than using a single tree.



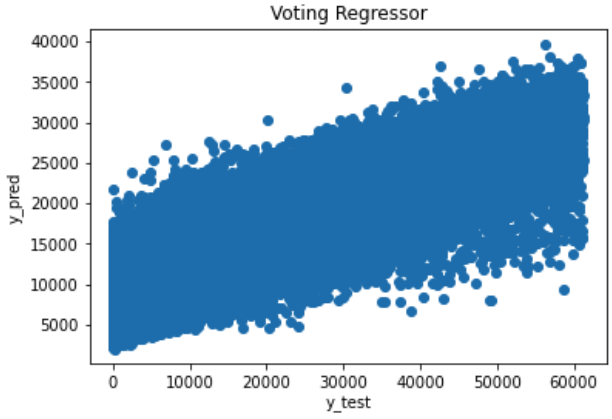
***<iv> KNN:***

Another popular method we could use is KNN regression. It is also a non-parametrics model, and is very easy to implement. Just as our decision tree model, we used grid search and cross validation again in order to find the best value for k. By selecting k = 12, our testing performance summary shows that RMSE = 12107.11, MAE = 8626.92, training score = 34.6% and testing score = 22.5%. This result is better than linear regression, but much worse than trees or random forests. And we could see that, even with hyperparameter tuning, the model is still overfitting that the training score is 10% higher than the testing score. Our understanding with these results is that KNN is based on feature similarity, but if many of our features are not relevant or important, then this is really prone to overfitting problems. Another shortcoming for this model is that KNN is not working well with a large dataset.




***<v> Voting Regressors:***

Our last model is a voting regressor. It is also an ensemble learning method that combines several diversified models together, and usually generates better results than individual models. We chose to combine linear regression, decision tree, and random forest together, which gives us RMSE = 8832.62, MAE 6571. 39, training accuracy = 62.0%, and testing accuracy = 58.7%. Obviously, it is more accurate than linear regression and KNN models, however, it does not surpass the decision tree prediction. Our understanding of this is that voting regressor also produces an averaging effect over each method.



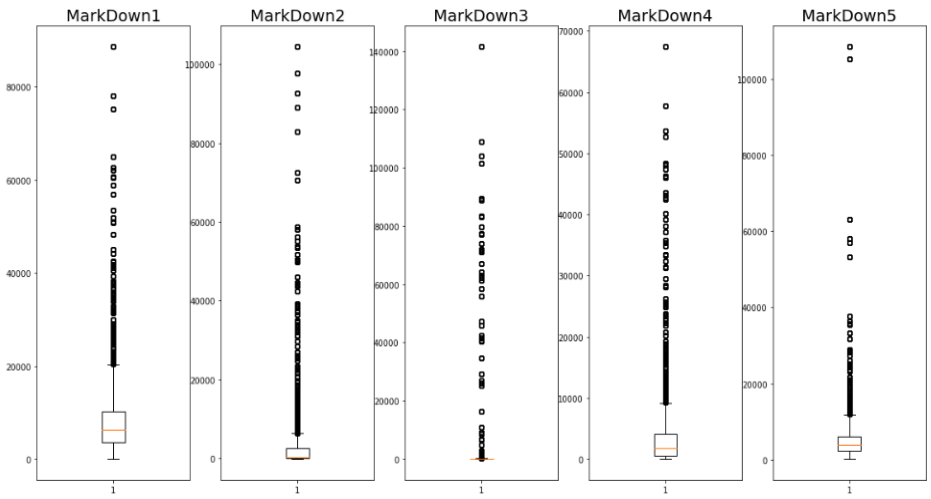
1.4 Model Comparison:

Since we have stored scores in the dictionary, we then converted it into a Pandas dataframe, and sorted them by RMSE ascendingly. From the sorted results, random forest is the best model with RMSE equals 3323.428. One thing about RMSE is that it penalizes more or gives higher weight to large errors. However, MAE is also a great metric that measures the average magnitude of prediction errors without taking their direction into account. Also, we could see that using RMSE, MAE or R-squares would give us the same ranking. By looking at the training and testing accuracy scores from earlier sections, most of our models are not overfitting too much, controlled by procedures like hyperparameter tuning, or backward propagation. Only the knn model overfits data a lot. In this problem, random forests and regression trees work really well in generating good prediction outcomes, but simple methods like knn or linear regression are not as effective as more complicated models.

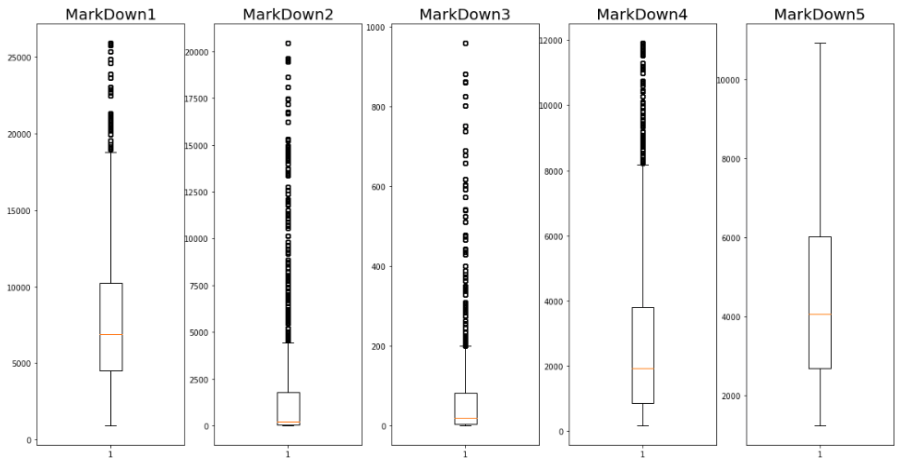
| | Model | RMSE | MAE | Test_Accuracy |
|---|---|---|---|---|
| 2 | Random Forest | 3323.428 | 1732.744 | 94.159% |
| 1 | Regression Tree | 4529.143 | 2589.040 | 89.152% |
| 4 | Voting Regressor | 8832.617 | 6571.393 | 58.743% |
| 3 | KNN Regressor | 12107.111 | 8636.921 | 22.483% |
| 0 | Linear Regression | 13415.207 | 10321.179 | 4.828% |

## 2). Data With MarkDown columns:

Similarly, in this section, we are going to repeat our training and testing process but using a different set of data. We will take the markdown information into account, but only look at a shorter period of time, i.e., after November 2011. We still need to first remove outliers in MarkDown1 to MarkDown5 columns by using the remove_outliers () function. It seems very effective as shown below:



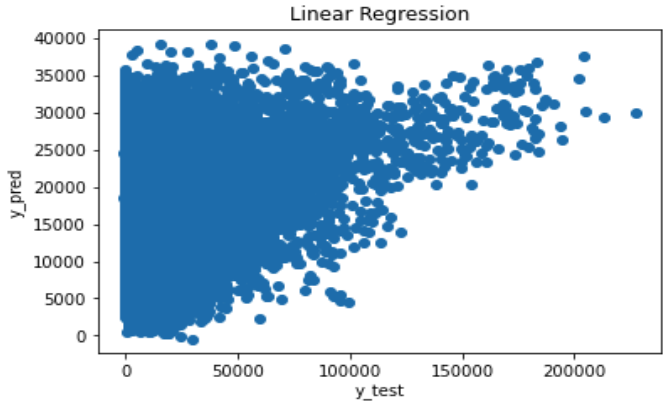

(Before removing outliers)



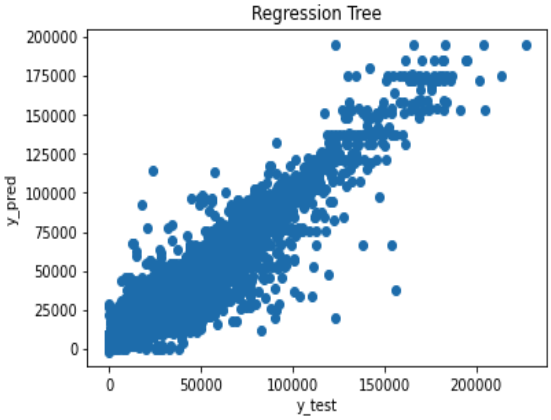

(After removing outliers)

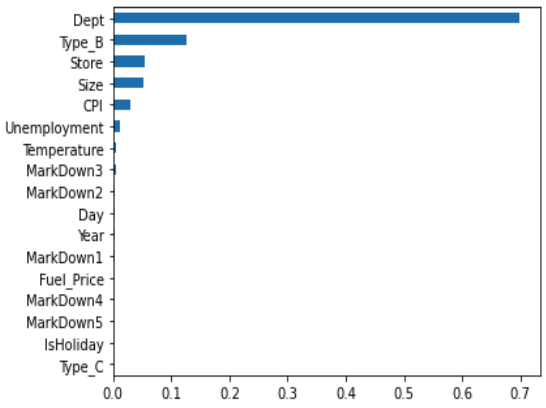
Then, we can add these columns to our numerical features, and do exactly the same training and prediction process as part 1). Below, I will only show the plots of our result as other things are not much different from part 1).

***<i> Linear Regression:***

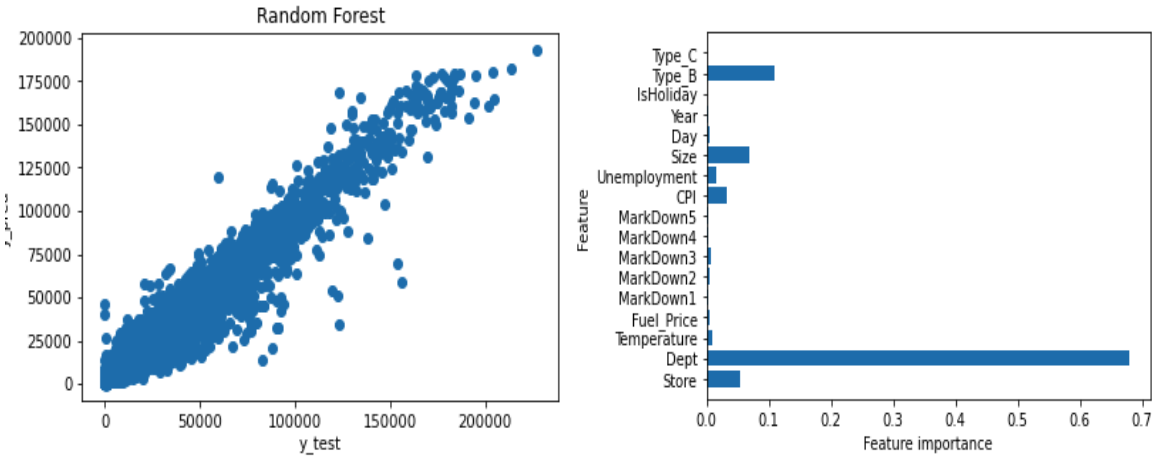



***<ii> Regression Tree:***

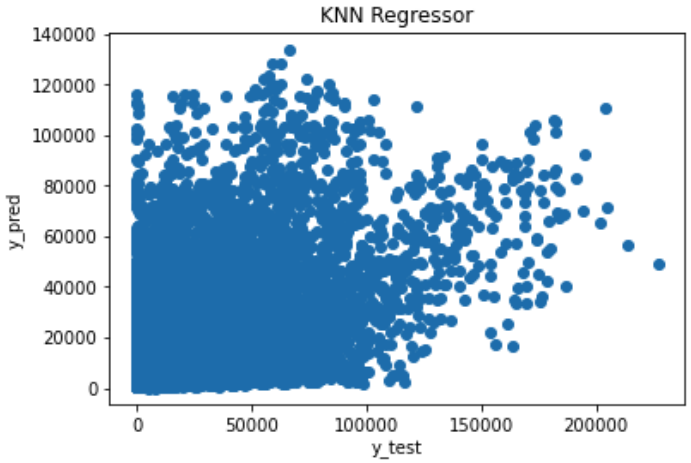


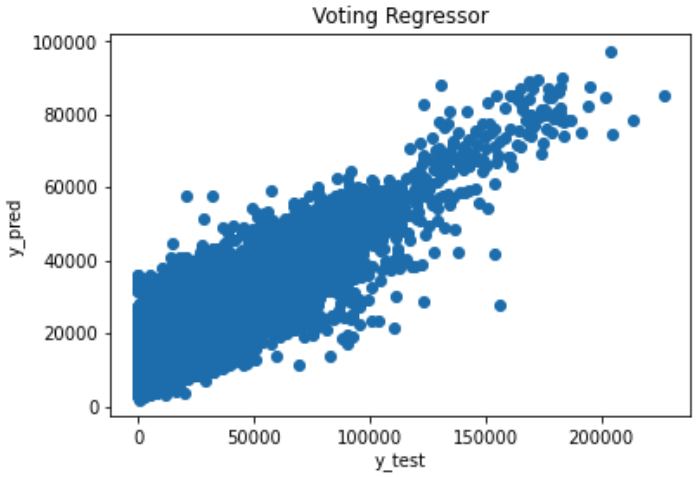
### *<iii> Random Forest:*




### *<iv> KNN Regressor:*




### *<v> Voting Regressor:*



**Model Comparison:**

| | Model | RMSE | MAE | Test_Accuracy |
|---|---|---|---|---|
| 2 | Random Forest | 4760.637 | 2441.699 | 95.925% |
| 1 | Regression Tree | 6456.744 | 3431.169 | 92.503% |
| 4 | Voting Regressor | 15168.416 | 10144.159 | 58.626% |
| 3 | KNN Regressor | 21537.542 | 13881.728 | 16.586% |
| 0 | Linear Regression | 22660.107 | 15748.605 | 7.664% |

(Part 2. with markdown data)

From above, the random forest model is still the best choice, and the ranking is the same as our part 1). To better compare these two scenarios, I listed our former chart again:

| | Model | RMSE | MAE | Test_Accuracy |
|---|---|---|---|---|
| 2 | Random Forest | 3323.428 | 1732.744 | 94.159% |
| 1 | Regression Tree | 4529.143 | 2589.040 | 89.152% |
| 4 | Voting Regressor | 8832.617 | 6571.393 | 58.743% |
| 3 | KNN Regressor | 12107.111 | 8636.921 | 22.483% |
| 0 | Linear Regression | 13415.207 | 10321.179 | 4.828% |

(Part 1. without markdown data)

The testing accuracy scores are higher than the ones in part 1), but RMSE and MAE are also higher. However, in general, the model rankings are exactly the same in these two scenarios.

## V. Conclusion

As we have described above, Random Forest is the best prediction model. Only based on the RMSE or MAE errors, we may decide not to include markdown information. However, data with markdown columns dropped have lower errors might also be due to the size of our dataset being large and generating a more stable model. Therefore, our choice of whether to include markdown or not really depends on the availability of data.

## VI. Reflection and Future Study

From earlier exploratory data analysis, Weekly_Sales is a time series. And in the above analysis, we selected five machine learning models for sales prediction. For future study, we could also consider time series forecasting methods such as ARIMA, Exponential Smoothing Models, Moving Average Models. Neural Network would also be a good choice as it tends to generate great predictions most of the time.