CAPSTONE PROJECT REPORT

Retail Store Analytics

Abstract

As retail market becomes extensively competitive, the ability to optimize on serving business processes while satisfying customer expectations has never been more important. Therefore, managing and channelizing data to work towards customer delight as well as generate healthy profits is crucial to survive prosperously. In the case of big retail players internationally as well as in India, data or rather big data analytics is now being applied at every stage of the retail process - tracking emerging popular products, forecasting sales and future demand through predictive simulation, optimizing product placements and offers via customer heat-mapping and many more. Alongside this, identifying the customers likely to be interested in particular product types based on their previous purchase behaviors, working out the best way to approach them through targeted marketing efforts and finally working out what to sell them next is what forms the core of data analytics. This article is the outcome of a descriptive research on the past, present and future of retail industry and the application of business analytics in shaping appropriate marketing strategies. The extracted features from page view data kept track during the visit along with some session and user information are fed to machine learning methods to build a model. We first built a Linear regression model as the base model and followed it by using various feature selection techniques. The results of linear models show very poor R square score. Hence, we proceeded with classification models. In classification models, we made use of decision tree, random forest and Ensemble methods bagging and boosting (Adaboost, Gradient boosting, XGboost). For hyper parameter tuning we adopted grid search. After building all the models and comparing the results, we found that random forest gave us the best score.

INDEX

Executive Summary

Big Data is all about the non-traditional ways of dealing with the modern digital data. Big data refers to datasets which are difficult to capture, store, manage and analyze effectively using current data base management software and concepts. It includes data stored in piles of wellstructured databases residing with organizations, streams of data generated from the dynamic social networks, various understandable and intangible signals generated by all kinds of digital equipment all over the place.

The uses of big data analytics are not exclusive to one industry. In retail we can use big data to make predictions about sales and merchandising. Retail Big data analytics has a massive impact on retail industries, improving the customer experience and reducing fraud. The retail sector is of major importance in modern society, as almost everyone nowadays must buy their basic needs. Predicting sales and demand for items allows retailers to offer better services to customers, and retailers can use customers‘ billing data to gather information for business intelligence. Big data analytics provides these organizations with more information on market decisions and help in segmenting customer based on their characteristics

We import all data files i.e. features, store, sale and we merge all fills. After processing the dataset and cleaning the inconsistencies, the numerical and categorical features used in the purchasing intention prediction model is generated. Various regression algorithms are used to predict Weekly Sales intent based on set of independent variables. The predictive models are also used to identify the variables that strongly influence the conversion using variable importance approaches. The models are evaluated using relevant model performance measures to arrive at the most robust models for prediction. The high-level recommendations for the project focus on bottom level of departments apply different strategic how to boost sales and find out problem on this department is there problems of Sales Executive, price, promotion.

Problem Statement

The challenge of modeling retail data is the need to make decisions based on limited history. Holidays and select major events come once a year, and so does the chance to see how strategic decisions impacted the bottom line. In addition, markdowns are known to affect sales the challenge is to predict which departments will be affected and to what extent.

1. Predict the department-wide sales for each store for the following year
2. Model the effects of markdowns on holiday weeks
3. Provide recommended actions based on the insights drawn, with prioritization placed on largest business impact.

Industry Review

Retailers differ from industrial companies in that they do not manufacture products. They purchase goods from manufacturers in large quantities for resale to consumers at a profit. The domestic Retail Store industry is mature and highly competitive. Retail is usually measured by sales.

These companies must provide desirable products, while managing inventory and controlling costs, to succeed. From an investment perspective, the sector generally tracks the broader stock market, on average. Some retail stocks can be volatile, though, making them best suited for short-term accounts. However, there are a few well-established companies suitable for the conservative investors. The retail industry provides products such as food, apparel, furniture, jewelry, retail banking, tourism, insurance, private healthcare, private education, private security firms, legal firms, publishers, public transport and others. Apart from this, the stores can be classified into the convenience store, specialty retailer, internet retailing, and various others. Consumer spending, which typically accounts for around two-thirds of the GDP, has been a key indicator of the health of the retail market.

Analytics in Retail Industry

Big Data is all about the non-traditional ways of dealing with the modern digital data. Big data refers to datasets which are difficult to capture, store, manage and analyze effectively using current data base management software and concepts

In retail we can use big data to make predictions about sales and merchandising. Retail Big data analytics has a massive impact on retail industries, improving the customer experience and reducing fraud. The retail sector is of major importance in modern society, as almost everyone nowadays must buy their basic needs. Predicting sales and demand for items allows retailers to offer better services to customers, and retailers can use customers‘ billing data to gather information for business intelligence. Big data analytics provides these organizations with more information on market decisions and help in segmenting customer based on their characteristics Social media analytics can also be used to inform companies about what their customers prefer. Applying sentiment analysis to such data provides the organization with early warnings when the customer turns to different products, allowing action to be taken by the organization.

Organizations have used segmentation of customers for many years, but this is now assisted by complex big data techniques such as real-time micro-segmentation which offers better-targeted. Organizations can also gain better targets for social marketing by understanding customer behaviors and predicting market sentiment trends Retailers are thus using data analytics in order to address new challenges and find opportunities based on increases in market expectations, competition, and volatility. In many companies, additional accuracy, clarity, and insight can be provided by the adoption of data analytics techniques, and such intelligence can be extended toward industry supply chains.

Problem Solving Methodology

- Understanding of Problem Statement
- Data Reading
- Literature Survey
- Knowing more about Features
- EDA

1. Data Preprocessing
2. Experiment with null values
3. Conversion of datatypes
4. Visualization
5. Scaling and Transformation

- Base model
- Feature Selection

1. VIF
2. RFE
3. Backward
4. Step Forward
5. Embedded System

- Linear model
- Non- Linear model
- Result

Dataset and Description

Data Dictionary

Data Sources – The given dataset contains historical sales data for 45 stores located in different regions - each store contains a number of departments. The company also runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of which are the Super Bowl, Labor Day, Thanksgiving, and Christmas. The weeks including these holidays are weighted five times higher in the evaluation than nonholiday weeks.

Dataset Description

Within this dataset we are provided with 3 tables namely features, sales and stores. The table _Stores‘ contains information about the 45 states, indicating the type and size of store. _Features‘ table contains additional data related to the store, department, and regional activity for the given dates. The features in this particular table include store number, dates, average temperature, fuel price, MarkDown1-5 these are anonymized data related to promotional markdowns. (Markdown data is only available after Nov 2011, and is not available for all stores all the time). So, the missing values were marked with an NA, Consumer Price Index (CPI), Unemployment rate and whether the week is a special holiday week. And the last table _Sales‘ contains historical sales data which covers from Feb 2010 to November 2012. This table contains fields such as the store number, department number, date, weekly sales and whether the week is a special holiday week.

Categorization of Features

Three tables were merged using an _inner merge‘ condition, to obtain a single table with all the necessary features for carrying out our analysis. The features extracted are as follows.

Numerical Features

Table 1 – Numerical Features in the Dataset

Feature Name	Feature Description	Max	Min
Temperature	Average temperature in the region	3.25	-28.29
Fuel_Price	Cost of fuel in the region	4.46	2.47
MarkDown1	Anonymized data related to promotional markdowns-1	88646.76	0.00
MarkDown2	Anonymized data related to promotional markdowns-2	104519.54	-265.76
MarkDown3	Anonymized data related to promotional markdowns-3	141630.61	-29.10
MarkDown4	Anonymized data related to promotional markdowns-4	67474.85	0.00
MarkDown5	Anonymized data related to promotional markdowns-5	108519.28	0.00
CPI	Consumer Price Index	227.23	126.06
Unemployment	Unemployment rate	14.31	3.87
Size	Size of the store	219622.00	34875.00
Weekly Sales	Sales for the given department in the given store	693099.36	0.00

Categorical Features

Table 2 – Categorical Features in the Dataset

Feature Name	Feature Description	Number of Categorical Values
IsHoliday	Whether the week is a special holiday week	2
Type	Store type	3
Store	Store number	45
Dept	Department number	99

Pre-Processing Data Analysis

It is a data mining technique that transforms raw data into an understandable format. Raw data (real world data) is always incomplete and that data cannot be sent through a model. That would cause certain errors. That is why we need to preprocess data before sending through a model.

1. Import libraries

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
%matplotlib inline
warnings.filterwarnings('ignore')
import warnings
import seaborn as sns
from IPython.display import Image
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.feature_selection import RFE
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostRegressor
from sklearn.linear_model import  LassoCV,Ridge
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor
from xgboost import XGBRegressor
```

Figure 1 – Importing libraries

2. Read data

```
stores=pd.read_csv('stores data-set.csv')
sales=pd.read_csv('sales data-set.csv')
```

```
features['Date'] = pd.to_datetime(features['Date'])
sales['Date'] = pd.to_datetime(sales['Date'])
df=pd.merge(sales,features, on=['Store','Date', 'IsHoliday'], how='inner')
df=pd.merge(df,stores, on=['Store'], how='inner')
```

```
df.head()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unempl
0	1	1	2010-05-02	24924.50	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	
1	1	2	2010-05-02	50605.27	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	
2	1	3	2010-05-02	13740.12	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	
3	1	4	2010-05-02	39954.04	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	
4	1	5	2010-05-02	32229.38	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	

Figure 2 – Merging 3. Checking for missing values

```
df.isnull().sum()
```

```
Store                0
Dept                 0
Date                 0
Weekly_Sales         0
IsHoliday            0
Temperature          0
Fuel_Price           0
MarkDown1       270889
MarkDown2       310322
MarkDown3       284479
MarkDown4       286603
MarkDown5       270138
CPI                  0
Unemployment         0
Type                 0
Size                 0
dtype: int64
```

Figure 3 – Checking for missing values

4. Missing value Treatment

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analyzed the behavior and relationship with other variables correctly.

It can lead to wrong prediction or classification. In Markdown 1-5 columns, as mentioned earlier, data was only available after Nov 2011, and was not available for all stores all the time, so these missing values were tackled by imputing 0.

5. Checking for categorical data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 421570 entries, 0 to 421569
Data columns (total 16 columns):
Store           421570 non-null int64
Dept            421570 non-null int64
Date            421570 non-null datetime64[ns]
Weekly_Sales    421570 non-null float64
IsHoliday       421570 non-null bool
Temperature     421570 non-null float64
Fuel_Price      421570 non-null float64
MarkDown1       150681 non-null float64
MarkDown2       111248 non-null float64
MarkDown3       137091 non-null float64
MarkDown4       134967 non-null float64
MarkDown5       151432 non-null float64
CPI             421570 non-null float64
Unemployment    421570 non-null float64
Type            421570 non-null object
Size            421570 non-null int64
dtypes: bool(1), datetime64[ns](1), float64(10), int64(3), object(1)
memory usage: 51.9+ MB
```

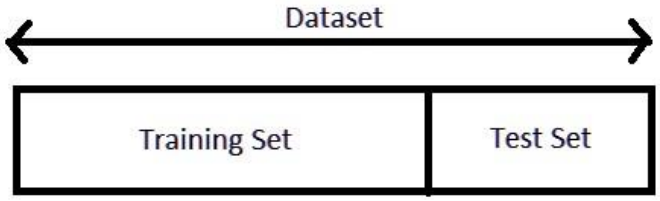
Figure 4: Checking for categorical data

Label Encoding – Label encoding was employed for two of the categorical columns namely, IsHoliday and Type columns. IsHoliday column is converted to 0s and 1s whereas Type column is converted to its dummy variables. Conversion of Date Column data type – Date column was converted from string object to date-time object using appropriate function.

6. Data splitting

Training and Test Data

As we work with datasets, a machine learning algorithm works in two stages. We usually split the data around 20%-80% between testing and training stages. Under supervised learning, we split a dataset into a training data and test data.



Insight on Salient Features of the Data

- Data is nonlinear
- Left Skew
- Markdown 1-year data not present
- CPI and Fuel has no relationship with Weekly Sales

Algorithms and Combined Techniques Used

1. Linear Model

Linear regression

2. Non-Linear Model

Decision Tree

Random Forest

3. Ensemble Techniques

Bagging

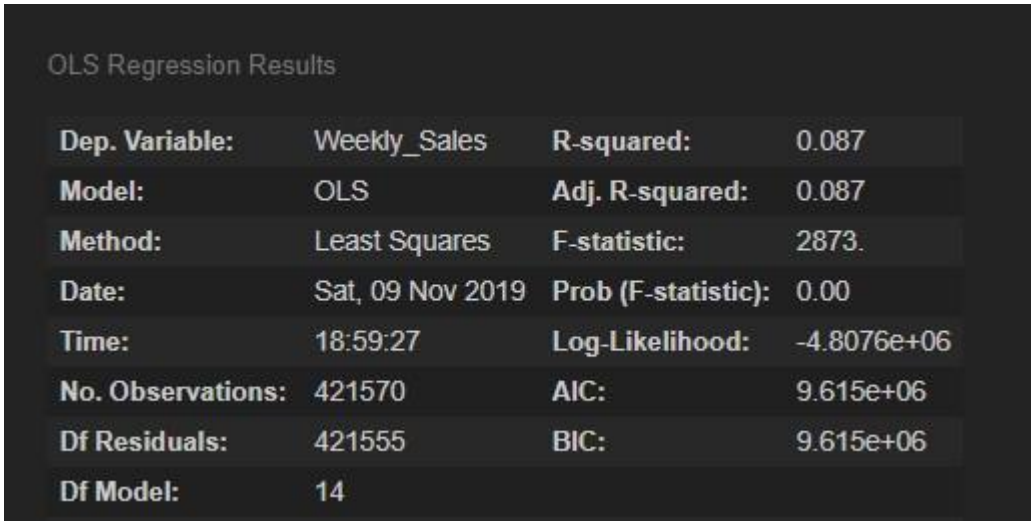
Boosting

Step by step walk through towards Model Building

1. Analyzing the Base Model

A linear regression model was built as a base model. This model is useful for finding relationship between two continuous variables. One is predictor or independent variable and other is response or dependent variable. It looks for statistical relationship but not deterministic relationship. Relationship between two variables is said to be deterministic if one variable can be accurately expressed by the other.

Ordinary least squares (OLS) is a type of linear least squares method for estimating the unknown parameters in a linear regression model. So OLS model was built in this section.



OLS Regression Results

Dep. Variable:	Weekly_Sales	**R-squared:**	0.087
Model:	OLS	**Adj. R-squared:**	0.087
Method:	Least Squares	**F-statistic:**	2873.
Date:	Sat, 09 Nov 2019	**Prob (F-statistic):**	0.00
Time:	18:59:27	**Log-Likelihood:**	-4.8076e+06
No. Observations:	421570	**AIC:**	9.615e+06
Df Residuals:	421555	**BIC:**	9.615e+06
Df Model:	14		

Figure 5 – OLS Regression Results Description of the summary R-Squared

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

Adj. R-Squared

The adjusted R-squared is a modified version of R-squared that has been adjusted for the number of predictors in the model. The adjusted R-squared increases only if the new term improves the model more than would be expected by chance. It decreases when a predictor improves the model by less than expected by chance.

Skewness

Skewness is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution.

Kurtosis

Kurtosis is a statistical measure that defines how heavily the tails of a distribution differ from the tails of a normal distribution. In other words, kurtosis identifies whether the tails of a given distribution contain extreme values.

F-Statistic

An F statistic is a value you get when you run an ANOVA test or a regression analysis to find out if the means between two populations are significantly different.

Durbin-Watson

The Durbin Watson Test is a measure of autocorrelation (also called serial correlation) in residuals from regression analysis. Autocorrelation is the similarity of a time series over successive time intervals.

Jarque Bera

The Jarque–Bera test is a goodness-of-fit test of whether sample data have the skewness and kurtosis matching a normal distribution.

Prob(JB)

It shows the probability of Jarque Bera.

Different scores obtained for this kind of model were –

Mean Squared Error (MSE) = 4.729834e+08

Root Mean Squared Error (RMSE) = 21748.1815

Mean Absolute Error (MAE) = 14575.9432

Test Accuracy Score = 0.0865

Train Accuracy Score = 0.0873

The accuracy score is very low. So, feature selection techniques are employed, to consider only the best features and thereby, reduce the dimension in the dataset.

2. Feature Selection Techniques

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of the model. The data features that are used to train machine learning models have a huge influence on the performance you can achieve.

Multiple feature selection techniques like Variance Inflation Factor (VIF), Recursive Feature Elimination (RFE), Backward Elimination, Forward Selection methods were employed to observe whether there is any significant improvement on the accuracy score obtained on the base obtained.

2(a) Variance Inflation Factor (VIF) Method

Variance inflation factor (VIF) is a technique to estimate the severity of multi-collinearity among independent variables within the context of a regression. It is calculated as the ratio of all the variances in a model with multiple terms, divided by the variance of a model with one term alone. However, the implementation of this method is straight-forward.

```
vif["VIF Factor"] = [variance_inflation_factor(predictors.values, i)
for i in range(predictors.shape[1])]
```

```
Iteration no  1
[4.506062183878648, 3.0758055996929587, 5.04961524849476, 15.917893436448516, 4.771518991694485, 1.1571000164977143, 1.0976347108329048,
3.759097302338094, 1.4561873083854493, 14.475390698248454, 3.60493098601358, 34.67428126118675, 18.138461761306328, 1.2425751433593784]
Max vif is for variable no :  11
Iteration no  2
[4.385204848296714, 3.0438039312995144, 4.4688738294125185, 13.602272634626276, 4.535720416460024, 1.156410952384596, 1.0963439349043076,
3.6963307158463943, 1.4476024767671747, 11.736014021392048, 3.574992205261479, 15.70361990972485, 1.238897364487619]
Max vif is for variable no :  11
Iteration no  3
[4.011635041132327, 3.0121978955374717, 3.451674196960917, 9.891868974657024, 4.533694534046664, 1.155660825558047, 1.0959594893797349, 3.
696281222749155, 1.438885575885046, 11.229670883482363, 3.572419853336726, 1.23764384515995]
Max vif is for variable no :  9
Iteration no  4
[4.007952271618233, 2.924250110916076, 2.31088131885725, 4.849041280967638, 4.532662845359753, 1.1546126063041635, 1.0954845730076288, 3.6
92700819062026, 1.4327672890527117, 3.567132652661396, 1.2359309690095632]
Max vif is for variable no :  3
```

	Store	Dept	Type	Size	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	Temperature	IsHoliday
0	1	1	0	151315	0.0	0.0	0.0	0.0	0.0	-14.595679	0
1	1	2	0	151315	0.0	0.0	0.0	0.0	0.0	-14.595679	0
2	1	3	0	151315	0.0	0.0	0.0	0.0	0.0	-14.595679	0
3	1	4	0	151315	0.0	0.0	0.0	0.0	0.0	-14.595679	0
4	1	5	0	151315	0.0	0.0	0.0	0.0	0.0	-14.595679	0

Figure 6 – Depiction of VIF iterations and Features Obtained

After VIF method of feature selection was performed, the above figure shows the best features obtained from this algorithm. 11 best features were obtained from this algorithm.

When these features were fed to Linear Regression (Base) model, the different scores obtained for this model were –

Mean Squared Error (MSE) = 4.734783e+08

Root Mean Squared Error (RMSE) = 21759.5563

Mean Absolute Error (MAE) = 14585.9958

Test Accuracy Score = 0.0855

Train Accuracy Score = 0.0859

Clearly, there is no improvement in the accuracy scores for this model.

2(b) Recursive Feature Elimination (RFE) Method

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are ranked by the model's coefficient attributes, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model.

```
cols = list(x.columns)
model = LinearRegression()
#Initializing RFE model
rfe = RFE(model, 9)
#Transforming data using RFE
X_rfe = rfe.fit_transform(x,y)
#Fitting the data to model
model.fit(X_rfe,y)
temp = pd.Series(rfe.support_,index = cols)
selected_features_rfe = temp[temp==True].index
print(selected_features_rfe)
```

```
Index(['Store', 'Dept', 'Type', 'Size', 'MarkDown3', 'MarkDown4', 'MarkDown5',
       'Temperature', 'IsHoliday'],
      dtype='object')
```

Figure 7 – RFE Technique and Features Obtained

After RFE method of feature selection was performed, the above figure shows the best features obtained from this algorithm. 9 best features were obtained from this algorithm.

When these features were fed to Linear Regression (Base) model, the different scores obtained for this model were –

Mean Squared Error (MSE) = 4.866161e+08

Root Mean Squared Error (RMSE) = 22059.3774

Mean Absolute Error (MAE) = 14834.9719

Test Accuracy Score = 0.0601

Train Accuracy Score = 0.0609

Using this model, the accuracy scores decreased, so this method is certainly not considered.

2(c) Backward Elimination Technique

Backward elimination (or backward deletion) is the reverse process. All the independent variables are entered into the equation first and each one is deleted one at a time if they do not contribute to the regression equation.

```
cols=list(x.columns)
pmax=1
while (len(cols)>0):
    p=[]
    x_1=x[cols]
    x_1=sm.add_constant(x_1)
    model=sm.OLS(y,x_1).fit()
    p=pd.Series(model.pvalues.values[1:],index=cols)
    pmax=max(p)
    features_with_p_max=p.idxmax()
    if(pmax>0.05):
        cols.remove(features_with_p_max)
    else:
        break
selected_features=cols
print(selected_features)
```

```
['Store', 'Dept', 'Type', 'Size', 'MarkDown1', 'MarkDown3', 'MarkDown4', 'MarkDown5', 'Temperature', 'IsHoliday']
```

Figure 8 – Backward Elimination Technique and Features Obtained

After backward elimination method of feature selection was performed, the above figure shows the best features obtained from this algorithm. 10 best features were obtained from this algorithm. When these features were fed to Linear Regression (Base) model, the different scores obtained for this model were –

Mean Squared Error (MSE) = 4.734783e+08

Root Mean Squared Error (RMSE) = 21759.5563

Mean Absolute Error (MAE) = 14585.9958

Test Accuracy Score = 0.0855

Train Accuracy Score = 0.0859

Clearly, there is no improvement in the accuracy scores for this model.

2(d) Step Forward Selection Technique

Forward selection is a type of stepwise regression which begins with an empty model and adds in variables one by one. In each forward step, one variable is added that gives the single best improvement to the model.

```
LR = LinearRegression()
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state = 0)



# Build step forward feature selection
sfs1 = sfs(LR,k_features = 5,forward=True,floating=False, scoring='r2',verbose=2,cv=5)

# Perform SFFS
sfs1 = sfs1.fit(X_train, y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    1.3s finished

[2019-11-13 12:36:38] Features: 1/5 -- score: 0.03306518645356136[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.2s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    2.6s finished

[2019-11-13 12:36:41] Features: 2/5 -- score: 0.05537136151298747[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.3s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    2.9s finished

[2019-11-13 12:36:44] Features: 3/5 -- score: 0.05780477327811693[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.4s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    3.0s finished

[2019-11-13 12:36:47] Features: 4/5 -- score: 0.0597716642668481[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wo
rkers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.5s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    3.0s finished

[2019-11-13 12:36:50] Features: 5/5 -- score: 0.060737245578236 79
```

```
feat_cols = list(sfs1.k_feature_idx_)
print(feat_cols)
```

```
[0, 1, 2, 3, 4]
```

Figure 9 – Step Forward Selection Technique and Features Obtained

After forward selection method of feature selection was performed, the above figure shows the best features obtained from this algorithm. 5 best features were obtained from this algorithm. When these features were fed to Linear Regression (Base) model, the different scores obtained for this model were –

Mean Squared Error (MSE) = 4.739344e+08

Root Mean Squared Error (RMSE) = 21770.0337

Mean Absolute Error (MAE) = 14587.7719

Test Accuracy Score = 0.0846

Train Accuracy Score = 0.0856

Clearly, there is no improvement in the accuracy scores for this model.

2(e) Embedded Method of Feature Selection

In embedded techniques, the feature selection algorithm is integrated as part of the learning algorithm. ... Decision tree algorithms select a feature in each recursive step of the tree growth process and divide the sample set into smaller subsets. In embedded techniques, the feature selection algorithm is integrated as part of the learning algorithm.

The most typical embedded technique is decision tree algorithm. Decision tree algorithms select a feature in each recursive step of the tree growth process and divide the sample set into smaller subsets. The more child nodes in a subset are in the same class, the more informative the features are. The process of decision tree generation is also the process of feature selection.

In this over target variable remain same weekly sales, but we have dropped Date from the dataset.

Lasso

In statistics and machine learning, lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces.

Best alpha using built-in LassoCV: **446404.847387**
Best score using built-in LassoCV: **0.060492**

3. Comparing the Feature Selection Techniques

Table 3 – Comparative Analysis on Feature Selection Techniques

Sr No	Model	Accuracy Train	Accuracy Test	RMSE	MSE	MAE
1	Linear Regression	0.087310	0.086518	21748.181502	4.729834e+08	14575.943217
2	VIF + Linear Regression	0.085963	0.085562	21759.556367	4.734783e+08	14585.995829
3	Backward Elimination + Linear Regression	0.085963	0.085562	21759.556367	4.734783e+08	14585.995829
4	RFE + Linear Regression	0.060977	0.060189	22059.377436	4.866161e+08	14834.971963
5	Step Forward Selection + Linear Regression	0.085657	0.084681	21770.033740	4.739344e+08	14587.771945

None of the feature selection methods significantly improved our accuracy scores, and also the accuracy of the base model is too low to even consider the model. Since, we found that this data doesn‘t follow a linear relationship. We shifted our focus on building non-linear models for the data.

Non-Linear Models

Non-linear regression is a form of regression analysis in which observational data are modeled by a function which is a non-linear combination of the model parameters and depends on one or more independent variables. The data are fitted by a method of successive approximations.

Non-linear regression is a regression in which the dependent or criterion variables are modeled as a non-linear function of model parameters and one or more independent variables. There are several common models, such as Asymptotic Regression/Growth Model, which is given by: b1 + b2 * exp (b3 * x).

4. Decision Tree

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. Tree based learning algorithms are considered to be one of the best and mostly used supervised learning methods. Tree based methods empower predictive models with high accuracy, stability and ease of interpretation. Unlike linear models, they map non-linear relationships quite well. They are adaptable at solving any kind of problem at hand (classification or regression). Decision Tree algorithms are referred to as CART (Classification and Regression Trees).

The accuracy score for Train with DecisionTreeRegressor was **100%.**

The accuracy score for Test with DecisionTreeRegressor was **91.23%.**

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=0)
```

```
from sklearn.tree import DecisionTreeRegressor
tree=DecisionTreeRegressor()
tree.fit(x_train,y_train)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
tree.score(x_test,y_test)
```

```
0.9102479589134521
```

```
tree.score(x_train,y_train)
```

```
1.0
```

Figure 10 – Implementing Decision Tree Classifier

GridSearchCV combines an estimator with a grid search preamble to tune hyper-parameters. The method picks the optimal parameter from the grid search and uses it with the estimator selected by the user. GridSearchCV inherits the methods from the classifier. Grid search is an approach to parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

On applying GridSearchCV, gave a maximum depth of 5 parameters with an accuracy of 54.27% on test data.

The accuracy score for Train with GridSearchCV was **54%.**

The accuracy score for Test with GridSearchCV was **54.27%.**

Mean absolute error is **9503.156357777561**

Mean squared error is **236780535.61545244**

Root mean squared error is**15387.674795610039**

```
parms ={'max_depth':[1,2,3,4,5]}
```

```
grid_search=GridSearchCV(DecisionTreeRegressor(),param_grid=parms,n_jobs=-1,verbose=0)
grid_search.fit(x_train,y_train)
```

```
C:\Users\Sangita\Anaconda1\lib\site-packages\sklearn\model_selection\_split.py:1978: FutureWarning: The default value of cv will change fr
om 3 to 5 in version 0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)


GridSearchCV(cv='warn', error_score='raise-deprecating',
            estimator=DecisionTreeRegressor(criterion='mse', max_depth=None,
                                            max_features=None,
                                            max_leaf_nodes=None,
                                            min_impurity_decrease=0.0,
                                            min_impurity_split=None,
                                            min_samples_leaf=1,
                                            min_samples_split=2,
                                            min_weight_fraction_leaf=0.0,
                                            presort=False, random_state=None,
                                            splitter='best'),
            iid='warn', n_jobs=-1, param_grid={'max_depth': [1, 2, 3, 4, 5]},
            pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
            scoring=None, verbose=0)
```

```
grid_search.best_params_
```

```
{'max_depth': 5}
```

Figure 11- Applying GridSearchCV to the Decision Tree

5. Random Forest

```
from sklearn.ensemble import RandomForestRegressor
RF=RandomForestRegressor()
RF.fit(x_train,y_train)
```

```
C:\Users\Sangita\Anaconda1\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators will change
from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

```
RF.score(x_test,y_test)
```

```
0.9467178497468123
```

Figure 12 – Implementing Random Forest Classifier

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes the model‘s prediction. The input of each tree is sampled data from the original dataset. In addition, a subset of features is randomly selected from the optional features to grow the tree at each node. Each tree is grown without pruning. Essentially, random forest enables a large number of weak or weakly-correlated classifiers to form a strong classifier.

The accuracy score for Train with RandomForest was **99%.**

The accuracy score for Test with RandomForest was **94.4%.**

Mean absolute error is **1959.2983999652097**

Mean squared error is **28864485.625835557**

Root mean squared error is **5372.567880058432**

An accuracy of 94.67% was obtained on test data upon building the random forest classifier.

This is by far the best accuracy score obtained for this dataset.

To see if the accuracy of test data can be slightly improved we implement ensemble techniques to our non-linear models.

6. Ensemble Techniques

6(a) Bagging

A parallel approach, where in multiple classifier models are built and average scores of models are considered which is more robust than a single decision tree classifier. Bagging is done to reduce the variance of a decision tree classifier, handles higher dimensionality data very well and also maintains accuracy for missing data.

An accuracy of 85.9% was obtained on test data upon building the decision tree classifier with bagging.

```
from sklearn.ensemble import BaggingRegressor
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=0)
```

```
DT_bag=DecisionTreeRegressor(max_depth=10,min_samples_leaf=10,min_samples_split=60)
```

```
DT_bag.fit(x_train,y_train)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=10, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=10,
                      min_samples_split=60, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

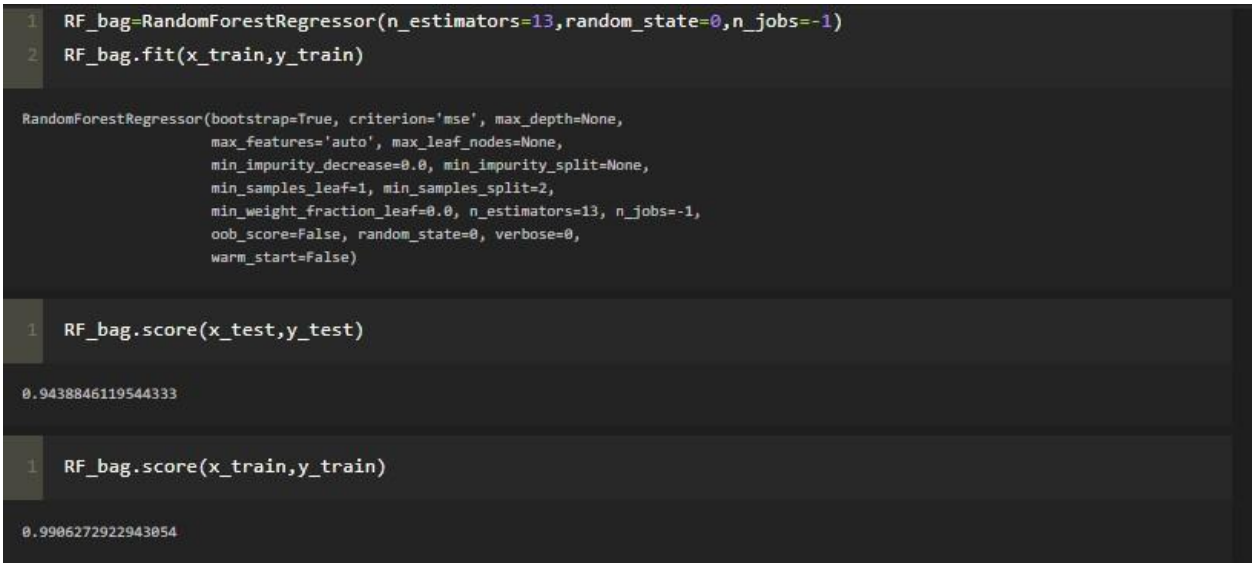
```
DT_bag.score(x_test,y_test)
```

```
0.8598304768083723
```

```
DT_bag.score(x_train,y_train)
```

```
0.861603045541637
```

Figure 13 – Decision Tree with Bagging



```
RF_bag=RandomForestRegressor(n_estimators=13,random_state=0,n_jobs=-1)
RF_bag.fit(x_train,y_train)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=13, n_jobs=-1,
                      oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
```

```
RF_bag.score(x_test,y_test)
```

```
0.9438846119544333
```

```
RF_bag.score(x_train,y_train)
```

```
0.9906272922943054
```

Figure 14 – Random Forest with Bagging

An accuracy of 94.3% was obtained on test data upon building the random forest classifier with bagging.

6(b) Boosting

Boosting is an ensemble technique in which the predictors are not made independently, but sequentially. Boosting attempts to create a strong classifier from a number of weak classifiers. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.

AdaBoost Technique

AdaBoost was the first really successful boosting algorithm developed for binary classification. It is the best starting point for understanding boosting. This method is best used to boost the performance of decision trees on binary classification problems. It may be referred to as discrete AdaBoost because it is used for classification rather than regression. AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners.

These are models that achieve accuracy just above random chance on a classification problem.

```
from sklearn.ensemble import AdaBoostRegressor
dt_Boost = AdaBoostRegressor(base_estimator=tree,n_estimators=10,learning_rate=0.01,random_state=1)
dt_Boost.fit(x_train,y_train)
```

```
AdaBoostRegressor(base_estimator=DecisionTreeRegressor(criterion='mse',
                                                       max_depth=5,
                                                       max_features=None,
                                                       max_leaf_nodes=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       presort=False,
                                                       random_state=None,
                                                       splitter='best'),
                  learning_rate=0.01, loss='linear', n_estimators=10,
                  random_state=1)
```

```
dt_Boost.score(x_test,y_test)
```

```
0.5432757508880157
```

```
dt_Boost.score(x_train,y_train)
```

```
0.5407005933472047
```

Figure 15 – Decision Tree with Ada Boost

An accuracy of 54.3% was obtained on test data upon building the decision tree classifier with AdaBoost technique.

```
RF_Boost = AdaBoostRegressor(base_estimator=RF,n_estimators=10,learning_rate=0.01,random_state=1)
RF_Boost.fit(x_train,y_train)
```

```
AdaBoostRegressor(base_estimator=RandomForestRegressor(bootstrap=True,
                                                       criterion='mse',
                                                       max_depth=4,
                                                       max_features=None,
                                                       max_leaf_nodes=None,
                                                       min_impurity_decrease=0.0,
                                                       min_impurity_split=None,
                                                       min_samples_leaf=1,
                                                       min_samples_split=2,
                                                       min_weight_fraction_leaf=0.0,
                                                       n_estimators=10,
                                                       n_jobs=None,
                                                       oob_score=False,
                                                       random_state=None,
                                                       verbose=0,
                                                       warm_start=False),
                  learning_rate=0.01, loss='linear', n_estimators=10,
                  random_state=1)
```

```
RF_Boost.score(x_test,y_test)
```

```
0.45225900002519215
```

```
RF_Boost.score(x_train,y_train)
```

```
0.44591062396617176
```

Figure 16 – Random Forest with Ada Boost

An accuracy of 45.2% was obtained on test data upon building the random forest classifier with AdaBoost technique.

Gradient Boost Technique

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. The gradient is used to minimize a loss function. In each round of training, the weak learner is built and its predictions are compared to the correct outcome that we expect. The distance between prediction and truth represents the error rate of our model. These errors can now be used to calculate the gradient. Gradient is basically the partial derivative of our loss function - so it describes the steepness of our error function. The gradient can be used to find the direction in which to change the model parameters in order to (maximally) reduce the error in the next round of training by ―descending the gradient‖. The accuracy obtained for the test data from this boosting technique was as low as 6.86%.

```
from sklearn.ensemble import GradientBoostingRegressor
Gradient_boost = GradientBoostingRegressor(n_estimators=10,learning_rate=0.01,random_state=1)
Gradient_boost.fit(x_train,y_train)
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                          learning_rate=0.01, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=10,
                          n_iter_no_change=None, presort='auto', random_state=1,
                          subsample=1.0, tol=0.0001, validation_fraction=0.1,
                          verbose=0, warm_start=False)
```

Figure 17 – Gradient Boost Algorithm

XGBoost Technique

XGBoost stands for Extreme Gradient Boosting; it is a specific implementation of the Gradient Boosting method which uses more accurate approximations to find the best tree model. It employs a number of nifty tricks that make it exceptionally successful, particularly with structured data. XGBoost uses the 2nd order derivative as an approximation and advanced regularization (L1 & L2), which improves model generalization. The training of this algorithm

is very fast and can be parallelized / distributed across clusters. The accuracy obtained for the test data from this boosting technique was 74.42%.

```
from xgboost import XGBRegressor
xgbr = XGBRegressor()
xgbr.fit(x_train,y_train, verbose=False)
```

```
C:\Users\Sangita\Anaconda1\lib\site-packages\xgboost\core.py:587: FutureWarning: Series.base is deprecated and will be removed in a future
version
  if getattr(data, 'base', None) is not None and \


[19:05:19] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objective/regression_obj.cu:152: reg:linear is now deprecated in f
avor of reg:squarederror.


XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
             silent=None, subsample=1, verbosity=1)
```

Figure 18 – XG Boost Algorithm

Model Evaluation (Best Model)

```
from sklearn.ensemble import RandomForestRegressor
RF=RandomForestRegressor()
RF.fit(x_train,y_train)
```

```
C:\Users\Sangita\Anaconda1\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators will change
from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

```
RF.score(x_test,y_test)
```

```
0.9467178497468123
```

Figure 19 – Model Evaluation

Random Forest Regressor gave the best and maximum accuracy score with a minimum RMSE compared to all the other techniques.

The accuracy score for Train with RandomForest was **99%.**

The accuracy score for Test with RandomForest was **94.4%.**

Mean absolute error is **1959.2983999652097**

Mean squared error is **28864485.625835557**

Root mean squared error is **5372.567880058432**

Comparison to Benchmark

While it is important to benchmark the predictive performance of a machine learning model, there are also important operational constraints that require consideration. These include the training and runtime performance characteristics of a model. That is, how long it takes to train a model, how long it takes to score new data, and the compute resources required to accomplish both of these.

	Model	Accuracy	RMSE
0	Decision Tree	0.913862	6678.373776
1	RandomForest Tree	0.942752	5444.439244
2	DT_bag Tree	0.859830	8519.218067
3	DT_Boost	0.543276	15378.004097
4	RF_Boost	0.452259	16840.716395
5	Gradient_Boosting	0.068645	21959.905028
6	XGBR	0.744200	11508.620821

Upon comparing accuracy on test data and RMSE value for each of the above models it is seen that Random Forest set the highest benchmark in terms of the accuracy score, and lowest benchmark in terms of RMSE. Hence, it can be concluded that Random Forest model was the best model.

Data Visualization

Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize their main characteristics, often with visual methods. It is a way of visualizing, summarizing and interpreting the information that is hidden in rows and column format. EDA is performed in order to define and refine the selection of feature variables that will be used for machine learning. This process is

valuable to data science projects since it allows getting closer to the certainty that the future results will be *valid*, *correctly interpreted*, and *applicable* to the desired business contexts. Such level of certainty can be achieved only after raw data is validated and checked for anomalies, ensuring that the data set was collected without errors.

EDA also helps to find insights that were not evident or worth investigating to business stakeholders and data scientists but can be very informative about a particular business. The goal of this step is to become confident that the data set is ready to be used in a machine learning algorithm. Once EDA is complete and insights are drawn, its feature can be used for supervised and unsupervised machine learning modeling.

Exploratory Data Analysis is majorly performed using the following methods:

- Univariate visualization — provides summary statistics for each field in the raw data set
- Bivariate visualization — is performed to find the relationship between each variable in the dataset and the target variable of interest
- Multivariate visualization — is performed to understand interactions between different fields in the dataset
- Dimensionality reduction — helps to understand the fields in the data that account for the most variance between observations and allow for the processing of a reduced volume of data.

Univariate Analysis

The distributions of multiple features in the data source are studied in this section.

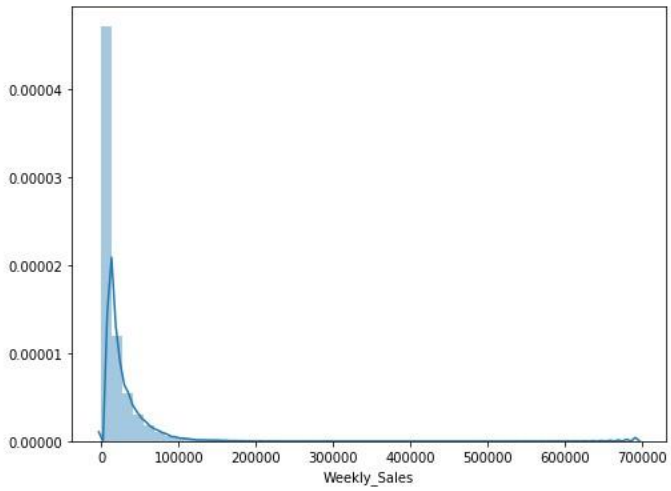


Figure 20 – Distribution of weekly sales

From the distribution of weekly sales it is observed that the sales are ranging from a minimum of $0 to a maximum of $0.693M in the source dataset. Clearly, the distribution is right skewed, with too many positive outliers in the weekly sales.

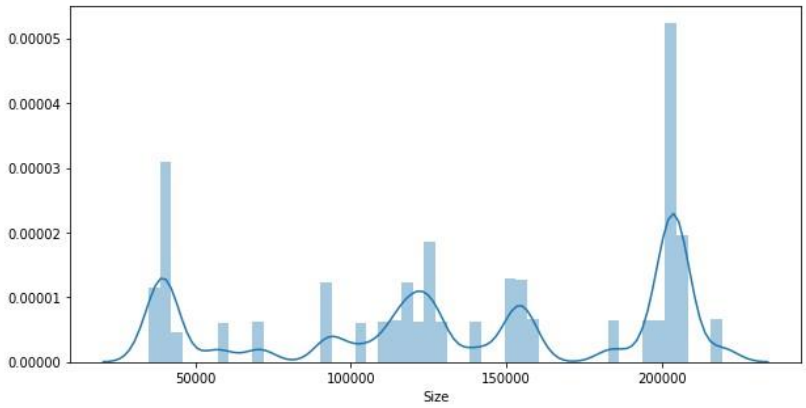


Figure 21 – Distribution of size of the stores.

The size of each type of stores varies there by leading to the multiple crests and troughs in the above plot.

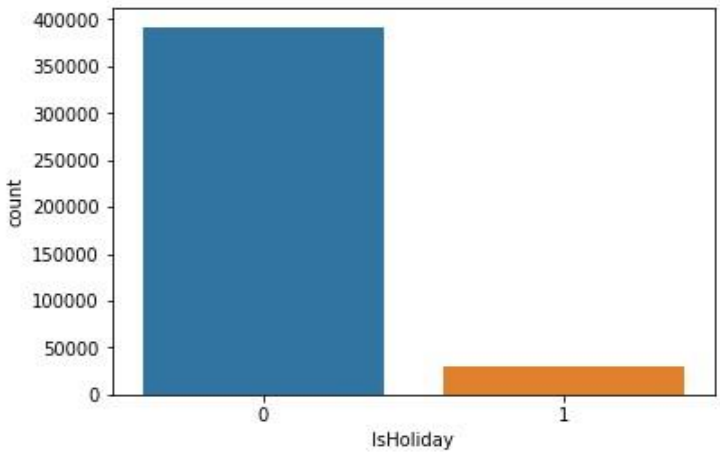


Figure 22 – Count plot for IsHoliday Field.

In the measure of the count plot for whether the week is a special holiday week, more than 90% of the records don‘t fall in the special holiday week category.

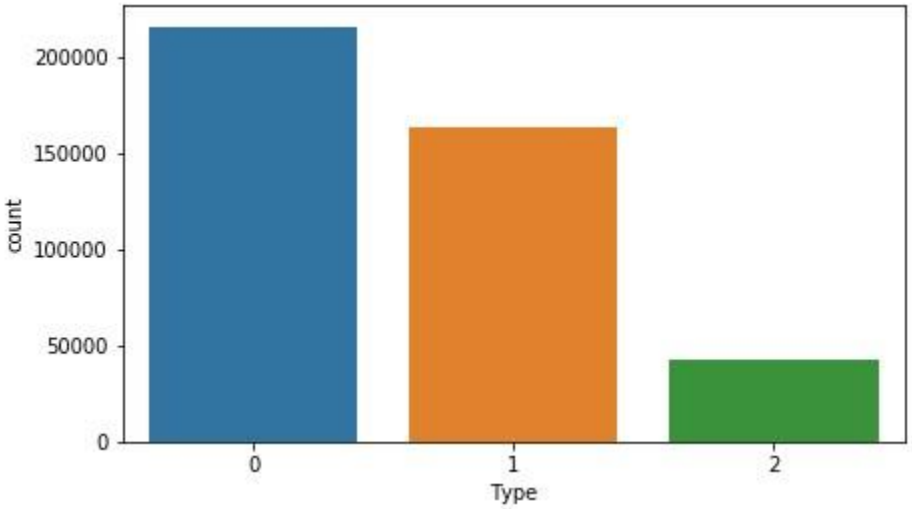


Figure 23 – Count plot for Type of the Store.

On observing the above count plot it is observed that many stores in the data source fall in the Type A category followed by Type B category. Fairly lesser proportion of the stores belongs to the Type C category.

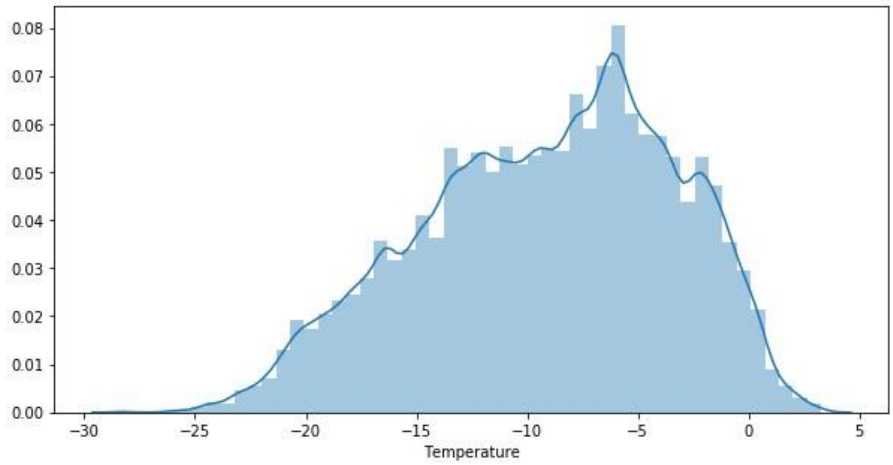


Figure 24 – Distribution of Temperature

In the distribution of temperature during the study tenure, most of the weeks were colder than the average temperature observed, leading to slight left skewness in the above distribution.

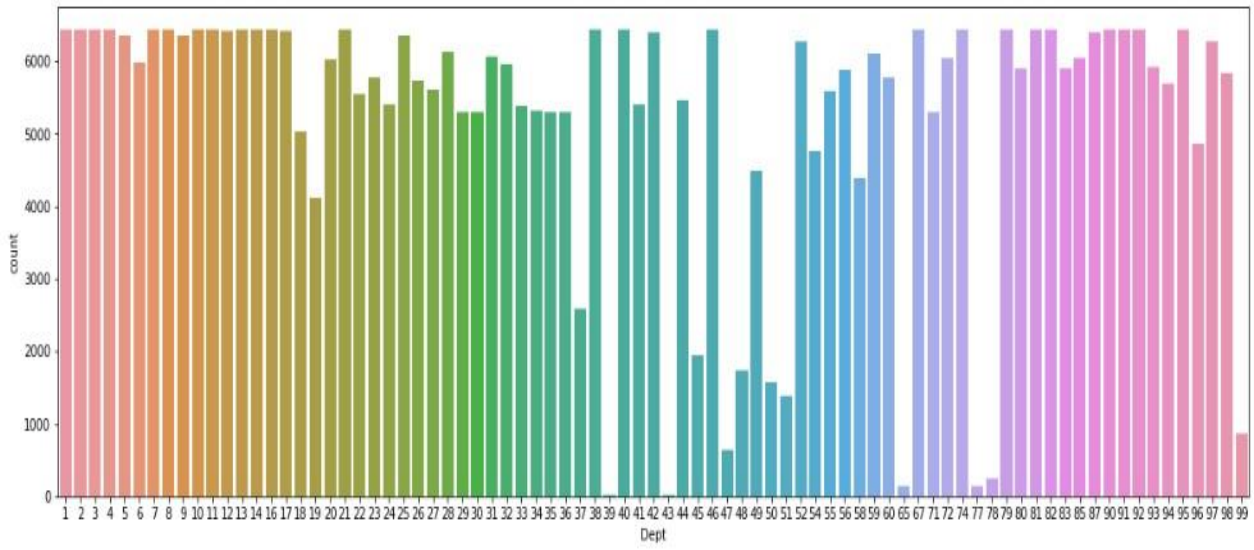


Figure 25 – Count Plot of the Departments

It is observed that department number 1,2,3 has the highest number stores and department number 39,43 had the lowest number of stores.

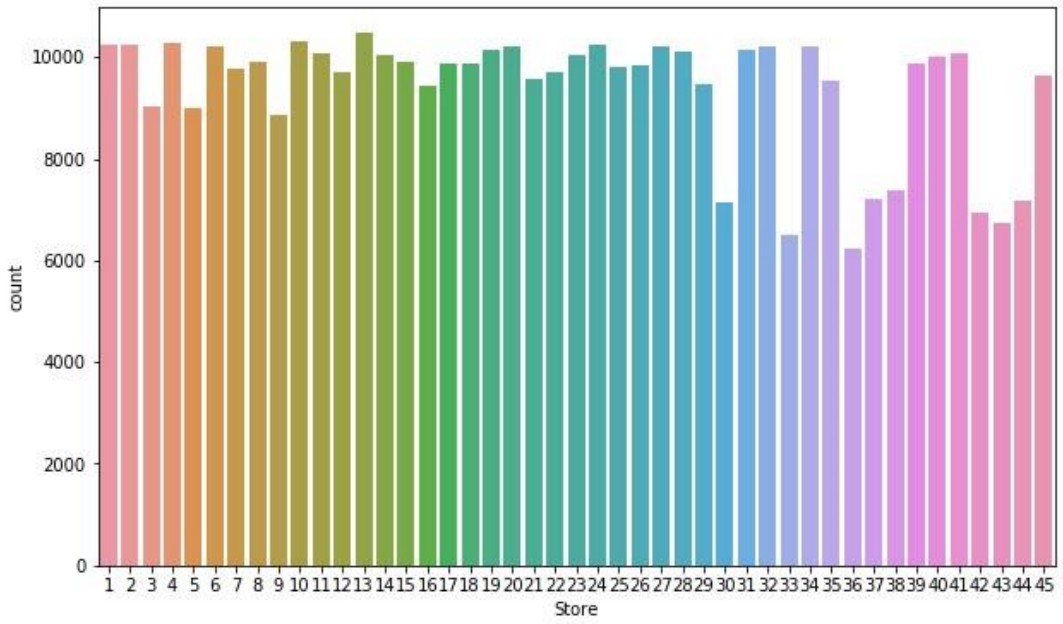


Figure 26 – Count Plot of Store Number

It is observed that Store number 13 has the highest number stores and store 33 had the lowest number of stores.

Bivariate Analysis

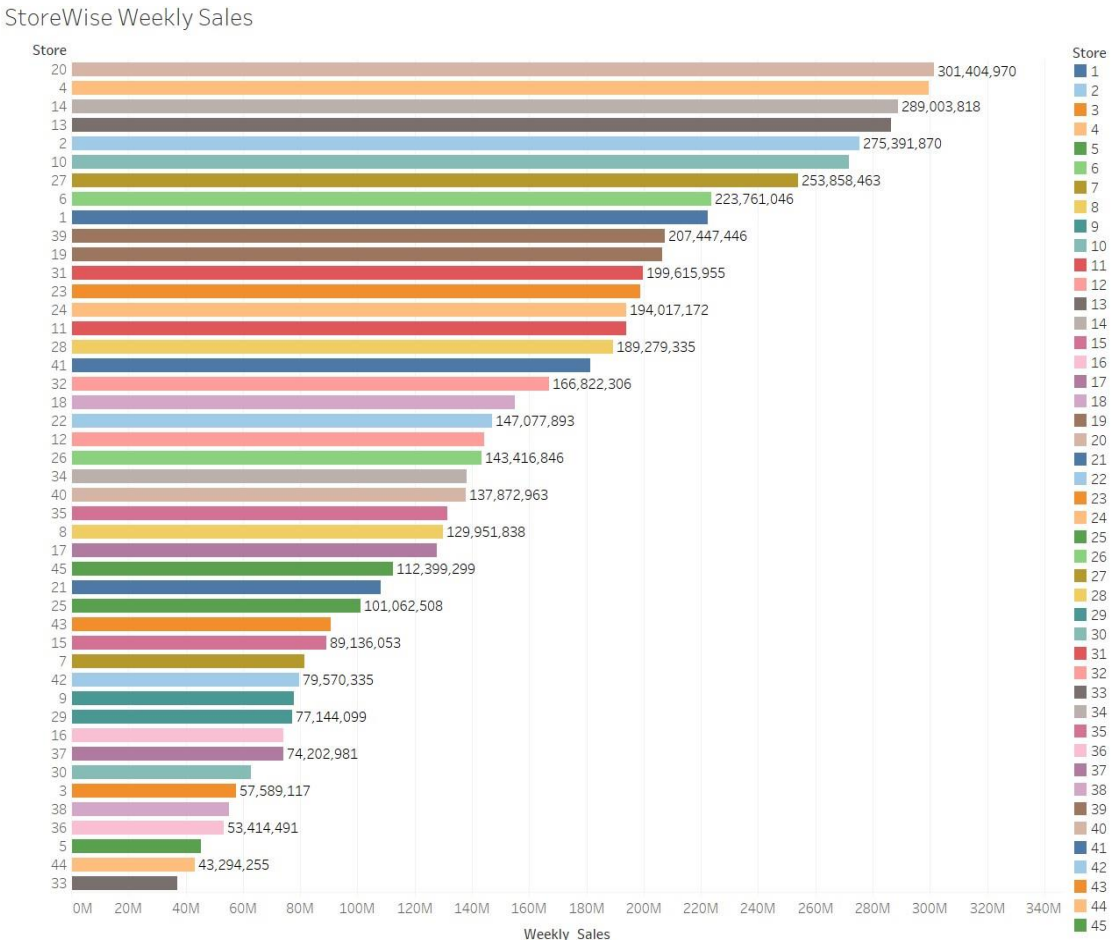


Figure 27 – Weekly Sales v/s Store Number

From the above figure store number 20 has the highest weekly sales at over 300 million, while store number 32 has the lowest weekly sales at fewer than 40 million.

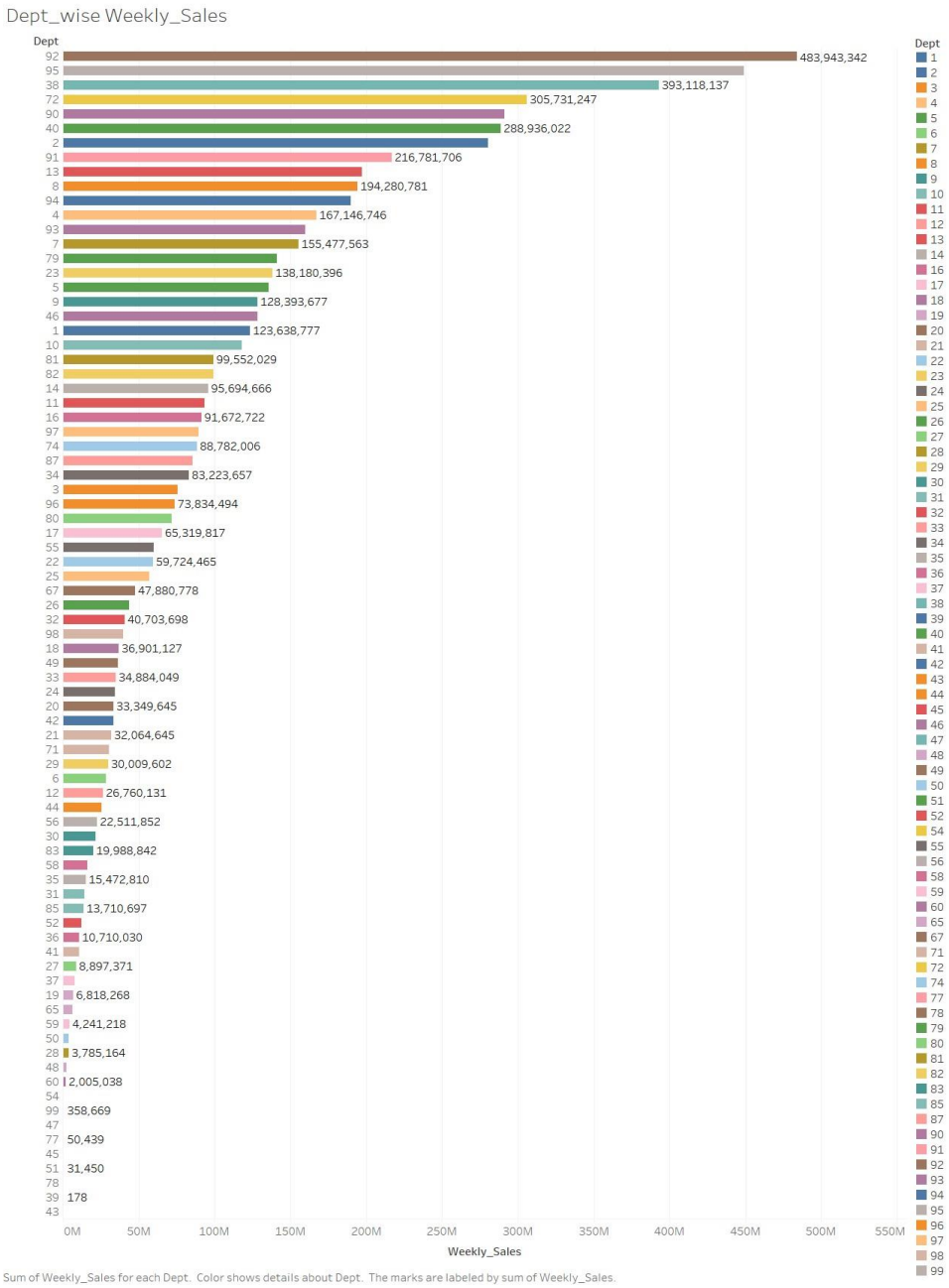


Figure 28 – Weekly Sales v/s Department Number

From the above figure department number 92 has the highest weekly sales at over 301 million, while store number 43 has the lowest weekly sales at fewer than 38 million.

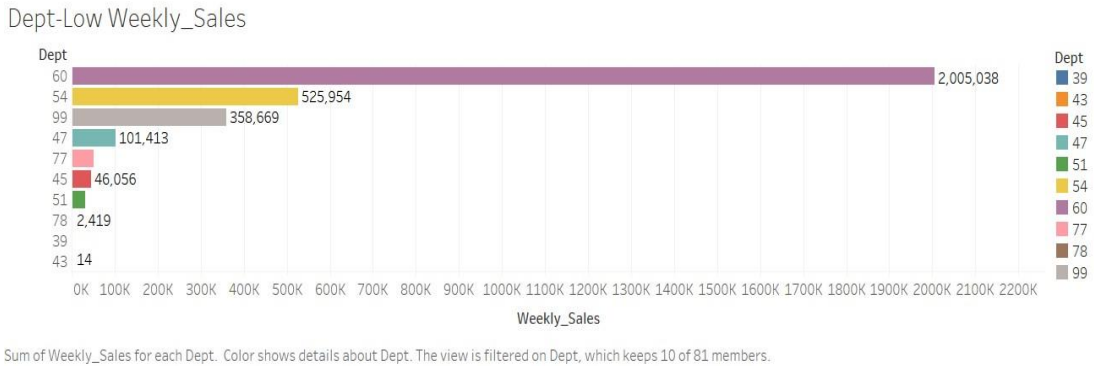


Figure 29 – Department Low Weekly Sales

Low Weekly Sales Department we want Focus on this Department to increased Sales, different Sales Strategies apply to Increase Sales and Revenue

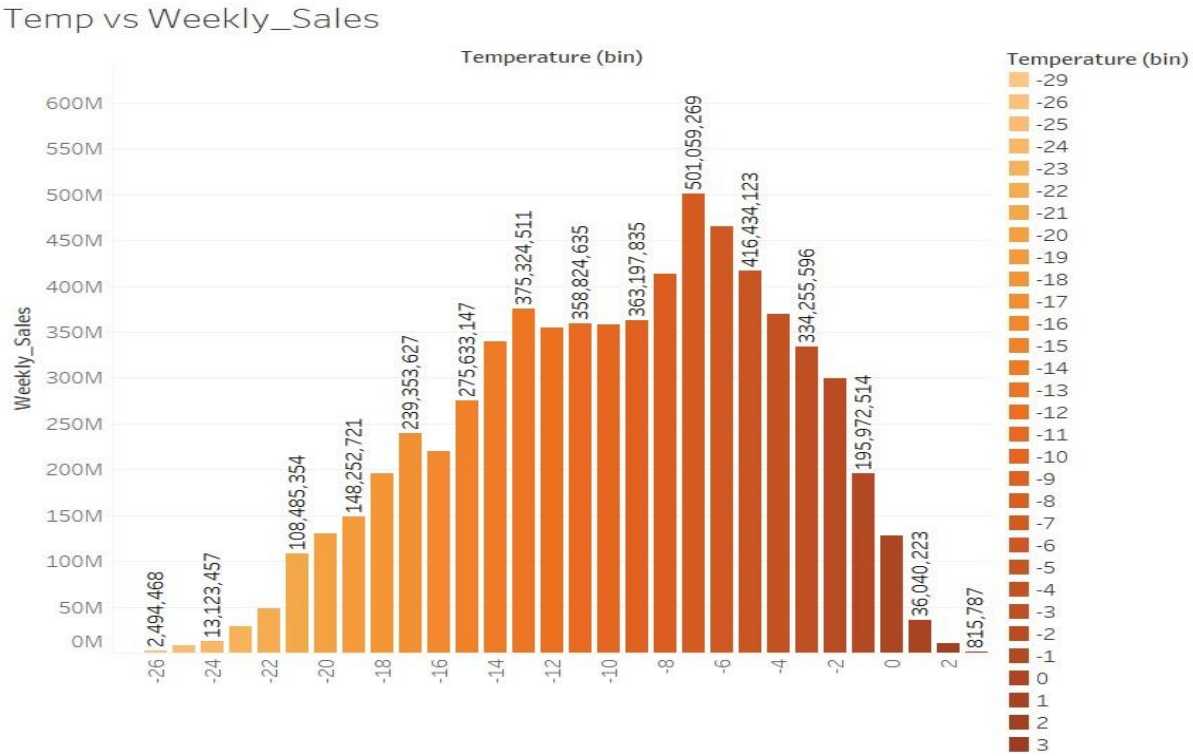


Figure 30 – Temperature v/s Weekly Sales

From the following graph, we can infer that the weekly sales are highest when temperature is -6 degree celsius. Weekly sales were high when the recorded temperatures were high.

Figure 31 – CPI v/s Weekly Sales

In the above figure, it is seen clearly that CPI has no relation with Weekly Sales.

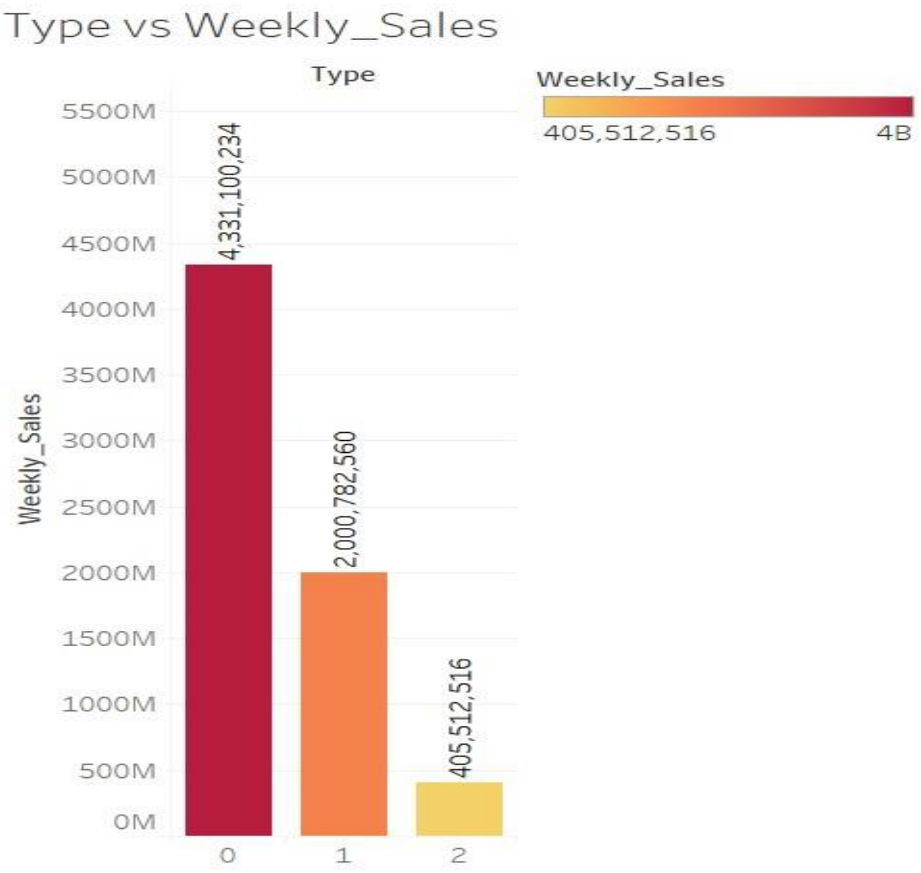


Figure 32 – Type v/s Weekly Sales

From the above figure it is observed that Type A stores had the maximum sales amounting to more than $4 Billion.

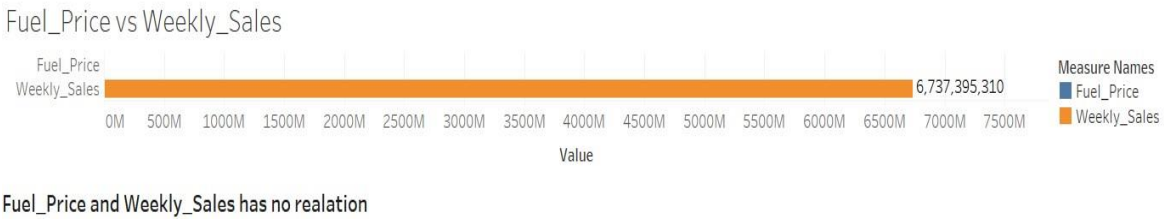


Figure 33 – Fuel Price v/s Weekly Sales Fuel

Price has no relation with Weekly Sales in the above figure.

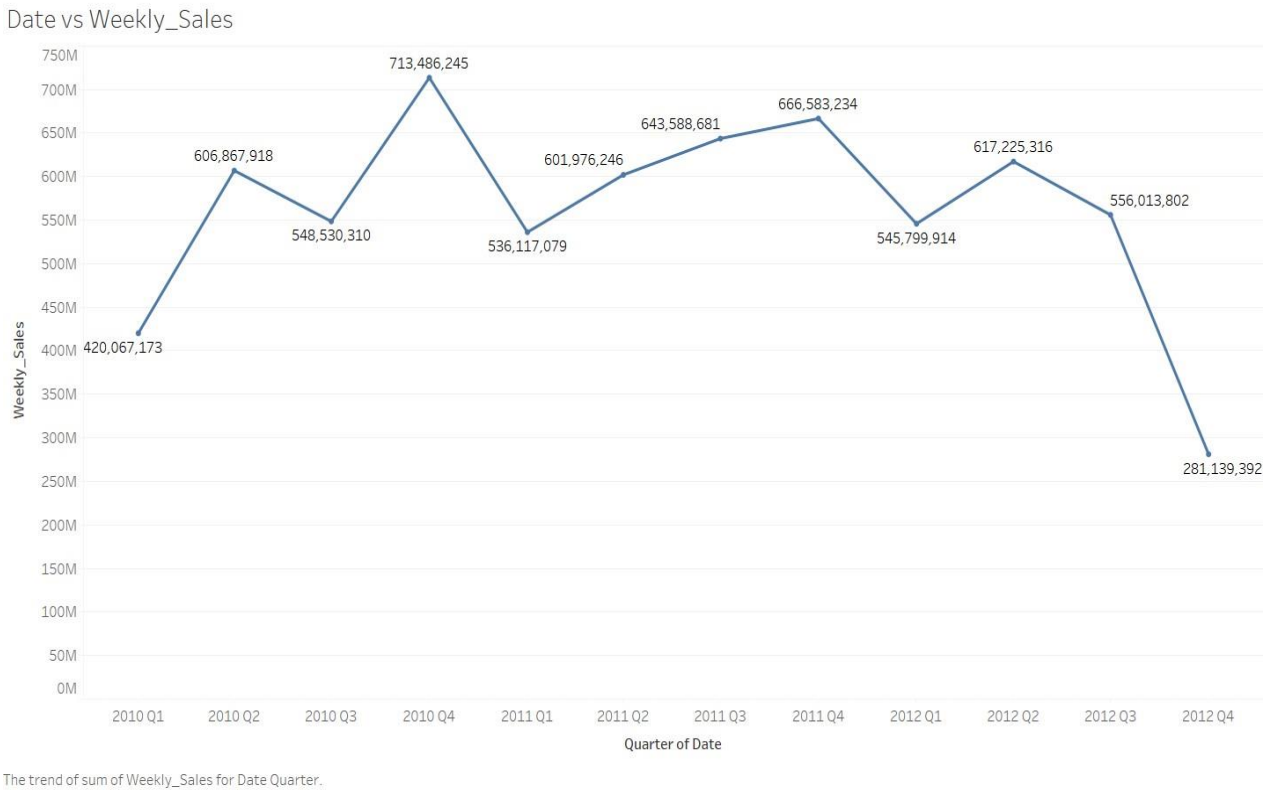


Figure 34 – Date v/s Weekly Sales

From above Figure it is observed that in the 4th quarter of the year 2011 and 2010 had highest sales compared to all the other quarters of those respective years. But sales during 4th quarter of 2012 were low, so we study what affected the sales during this period.

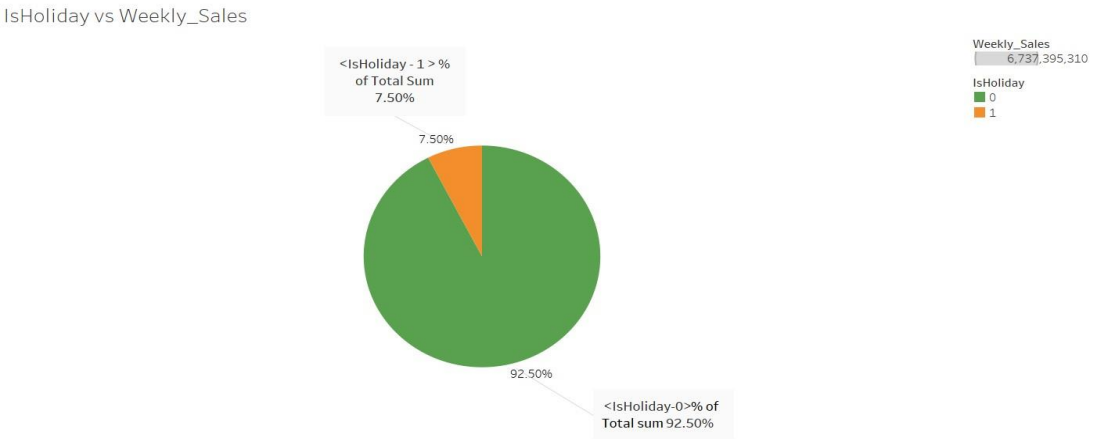


Figure 35 – IsHoliday v/s Weekly Sales

From the above pie chart, it is observed that 92.5% of the total sales happened on non-special holiday week category.

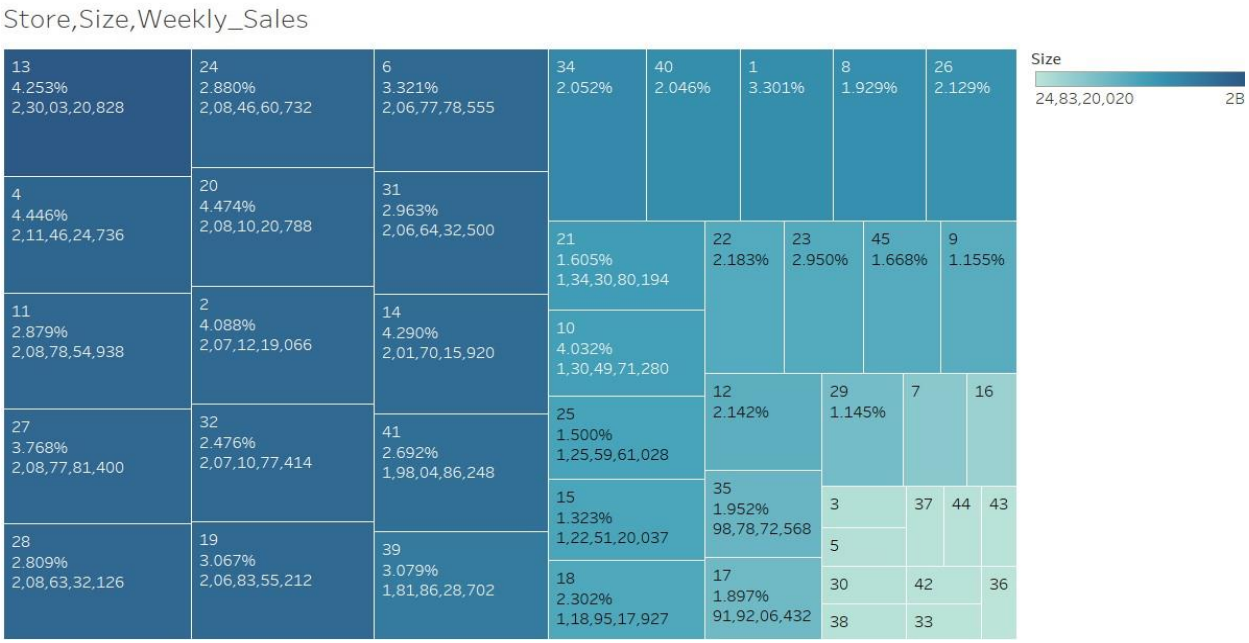
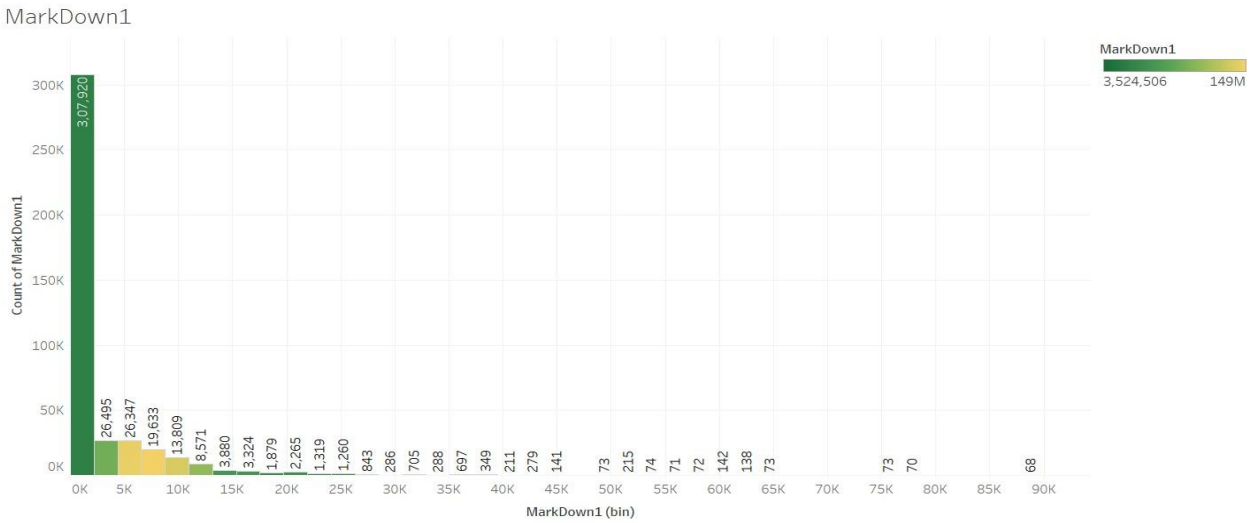


Figure 36 – Store, Size, Weekly Sales.

From the above figure, it is clear that store 13 has the highest size among all the other stores.

Size of the store isn‘t correlated with the weekly sales observed in the data.



When MarkDown1 is zero then Sales is high

Figure 37 – MarkDown1 v/s Weekly Sales

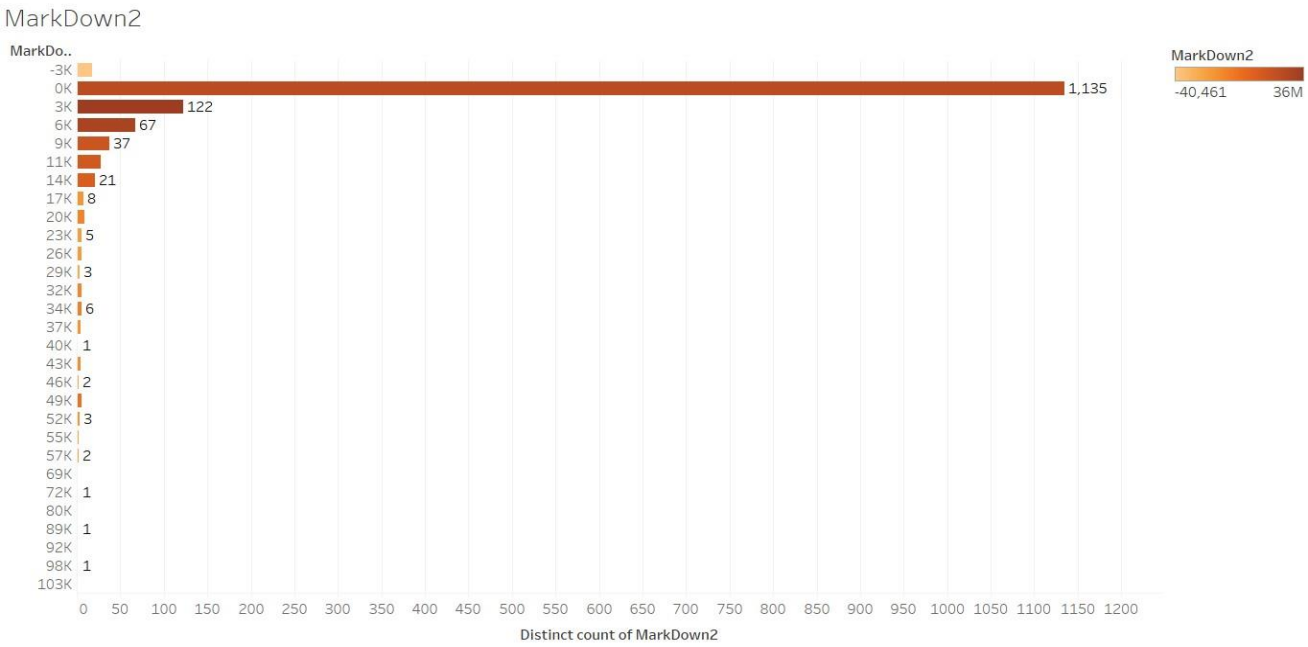


Figure 38 – MarkDown2 v/s Weekly Sales

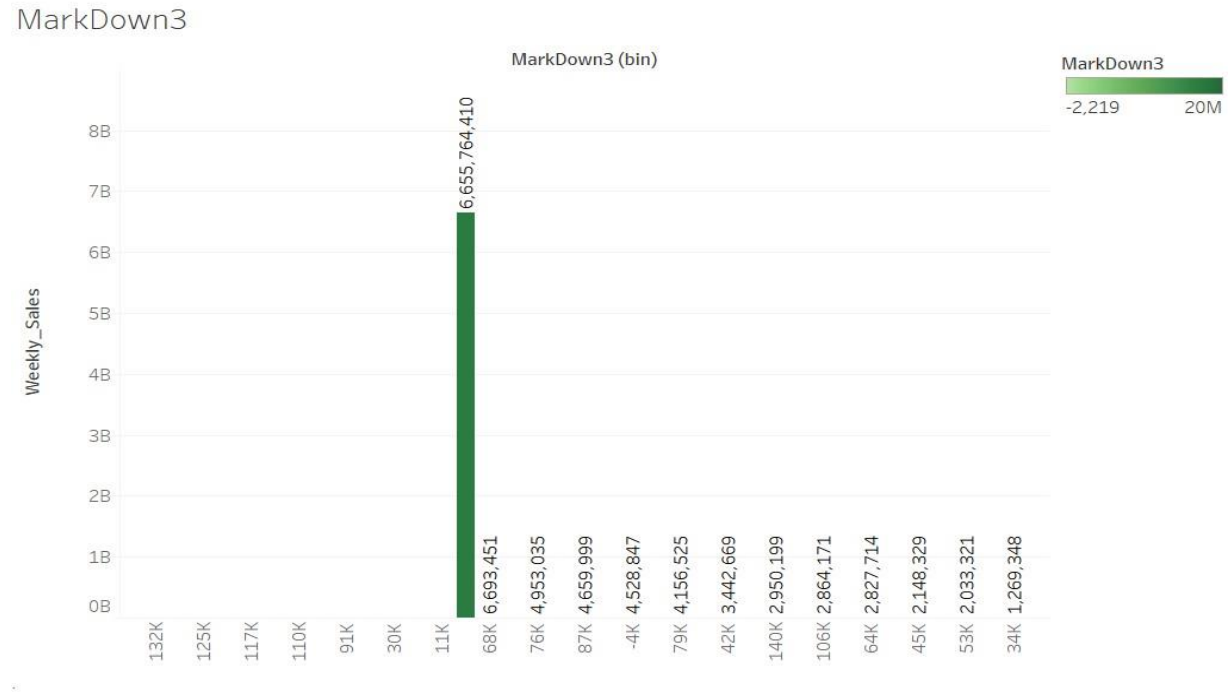


Figure 39 – MarkDown3 v/s Weekly Sales

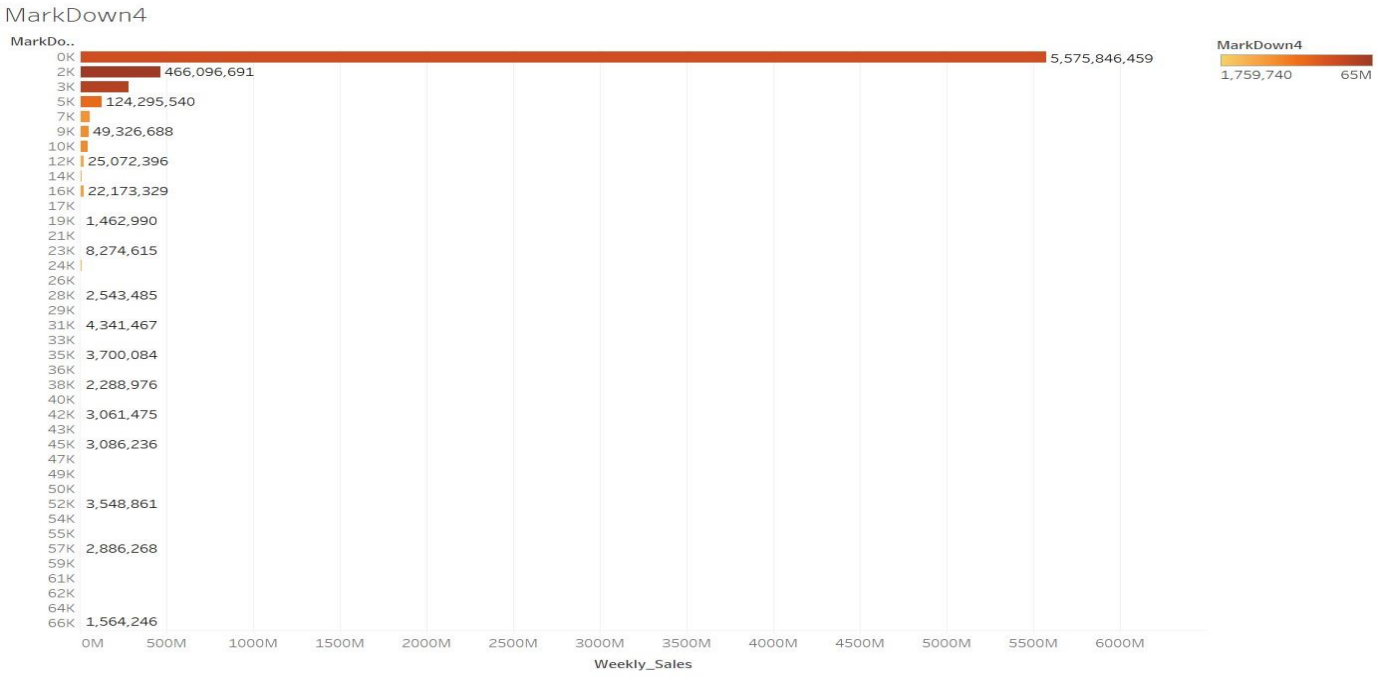


Figure 40 – MarkDown5 v/s Weekly Sales

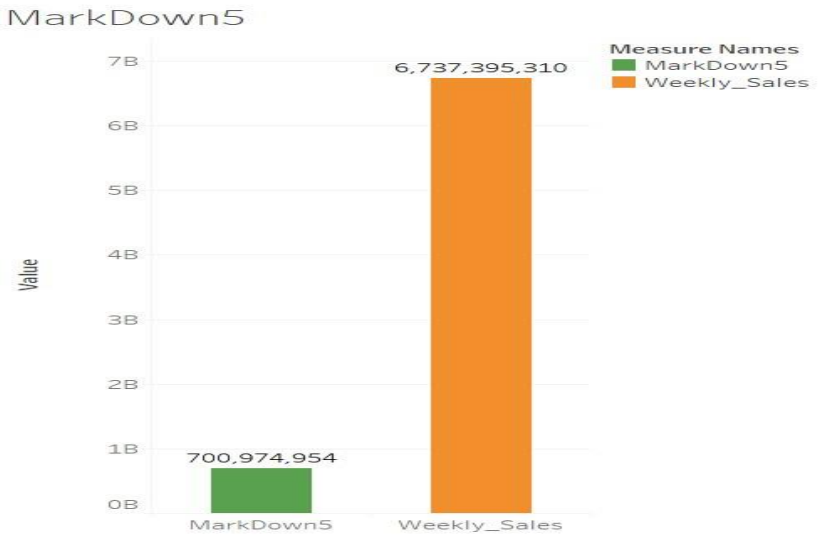


Figure 41 – MarkDown5 v/s Weekly Sales

Figures that belong in Figure 18 show that MarkDown1-5 has no effect on Weekly Sales. Surprisingly, to our notice it is seen that Weekly Sales are better, on non-special holiday weeks.

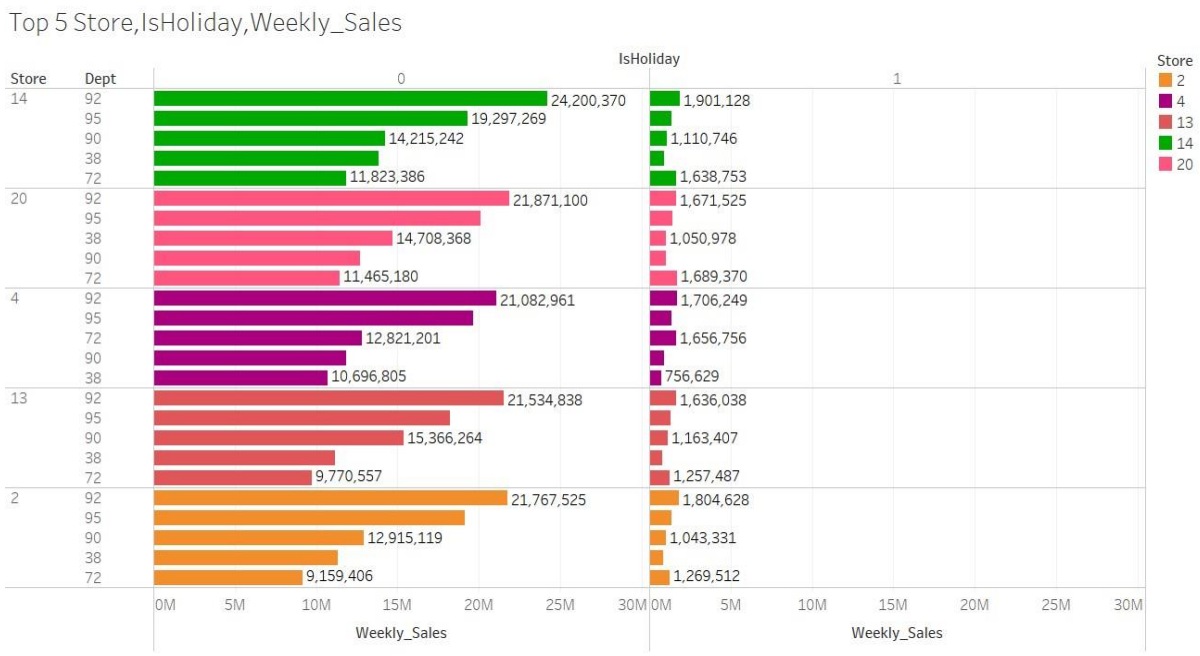


Figure 42 – Top Store, Dept vs Weekly Sales

The above figure represents Top 5 Weekly Sales Store and Department wise, using this we can find out what strategies were applied here, to obtain the sales this high.

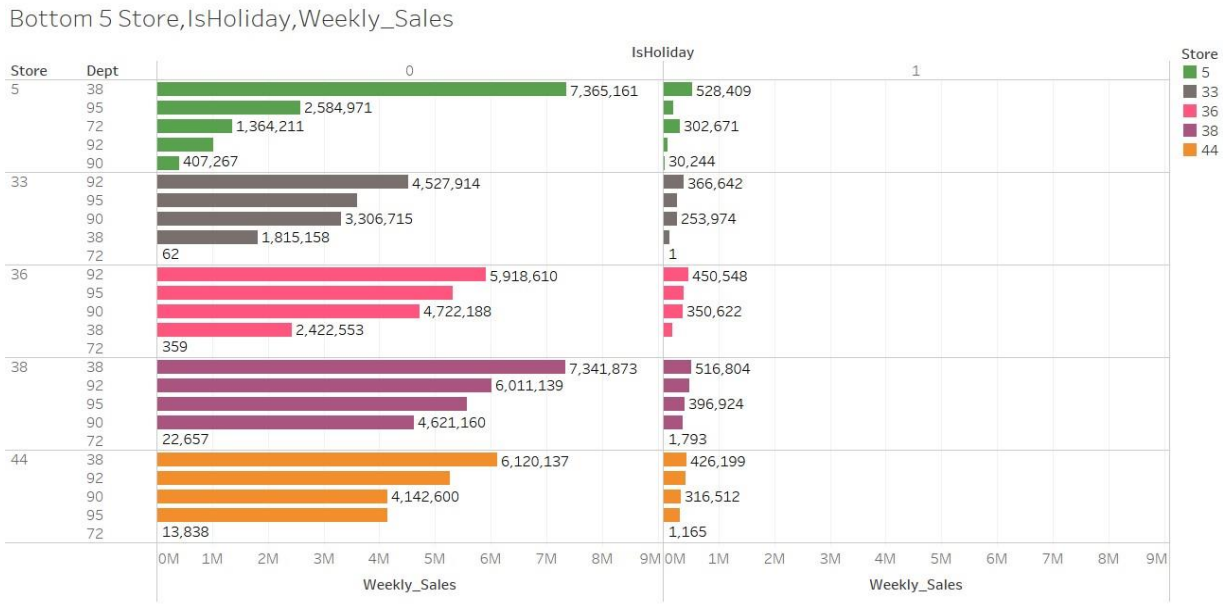


Figure 43 – Bottom Store, Dept vs Weekly Sales

The above figure shows Bottom 5 Weekly Sales Store and Department wise.

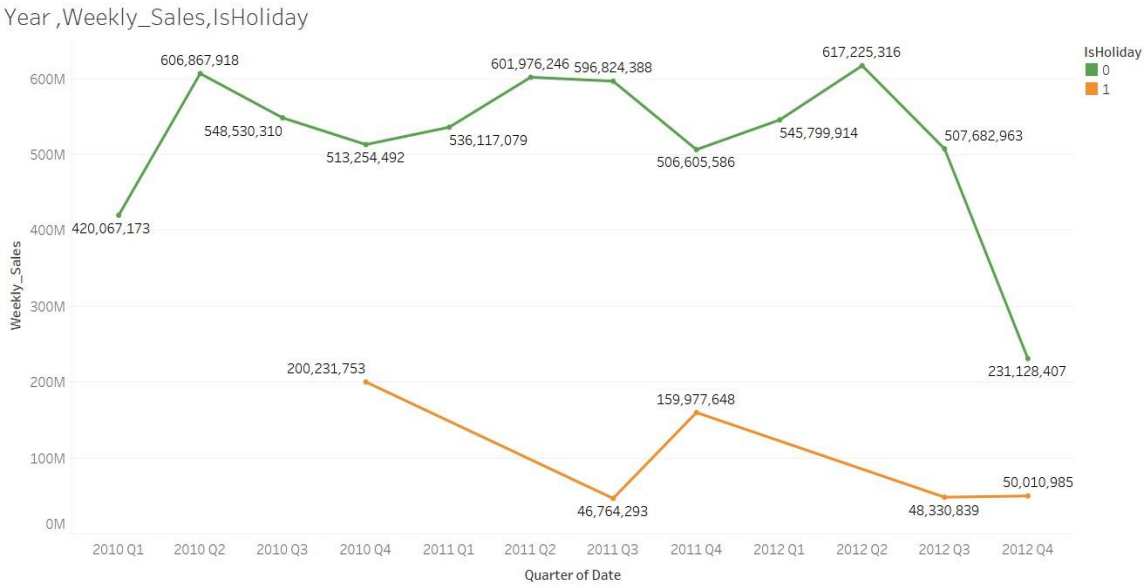


Figure 44 – Year v/s Weekly Sales hue IsHoliday

Above graph show quarter wise Weekly Sales v/s IsHoliday. Here we find a seasonality of sales present in the data is present in data. Clearly, non-special holiday weeks have better Weekly Sales.

Hyperparameter Visualization

Decision Tree Hyperparameters

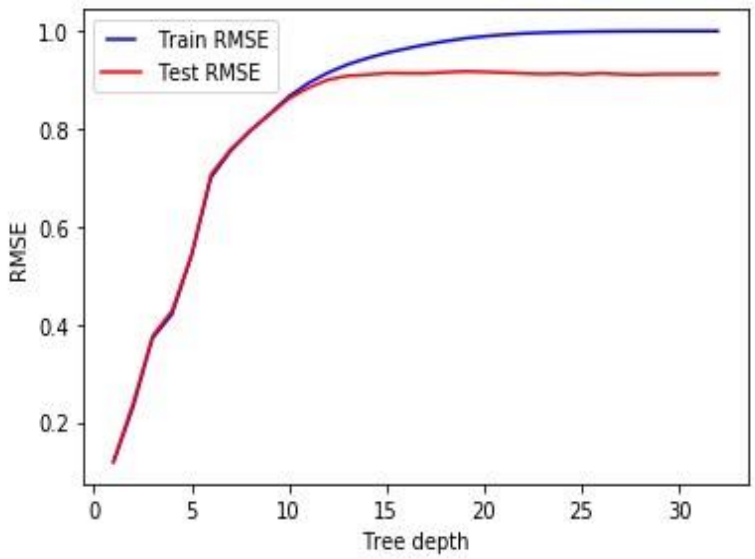


Figure 45 – Tree Depth v/s RMSE

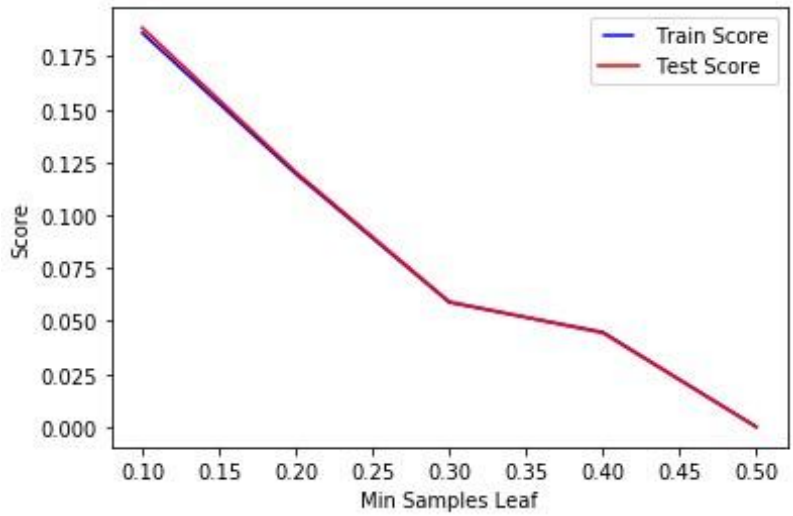


Figure 46 – Minimum Samples Leaf v/s Score

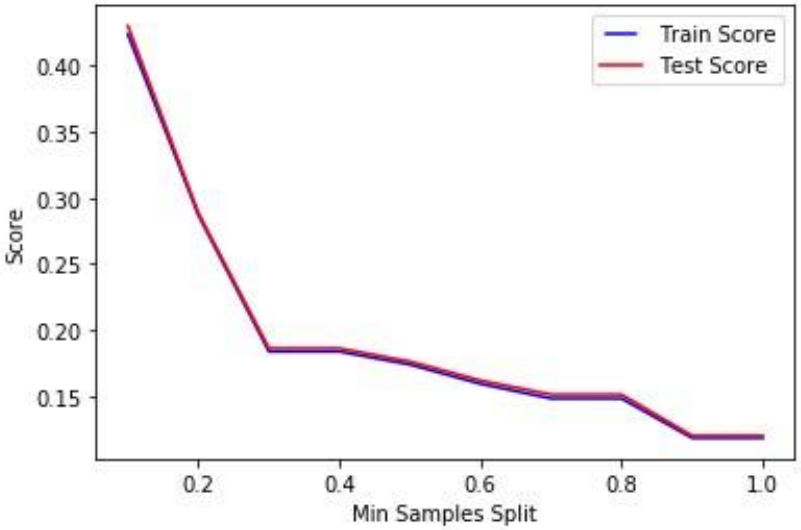


Figure 47 – Minimum Samples Split v/s Score

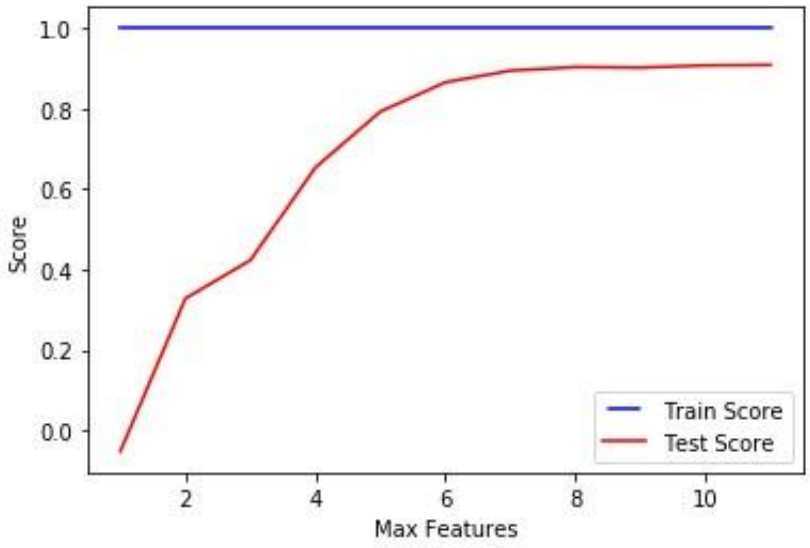


Figure 48 – Maximum Features v/s Score

Random Forest Hyperparamters

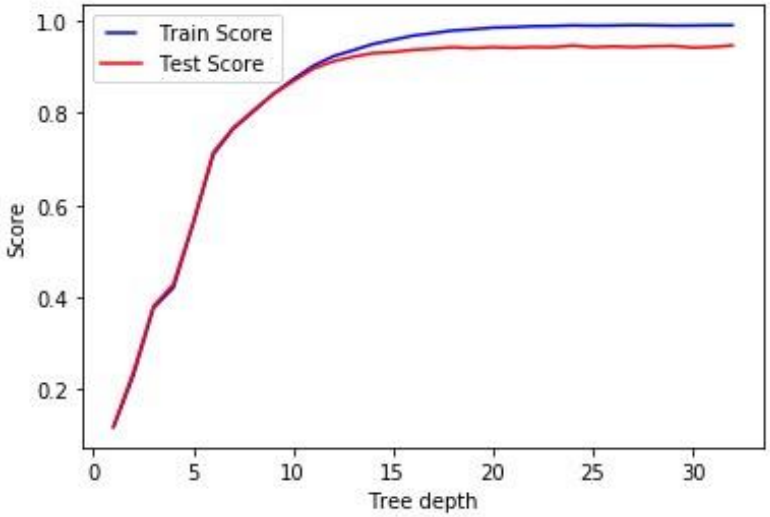


Figure 49 – Tree Depth v/s Score

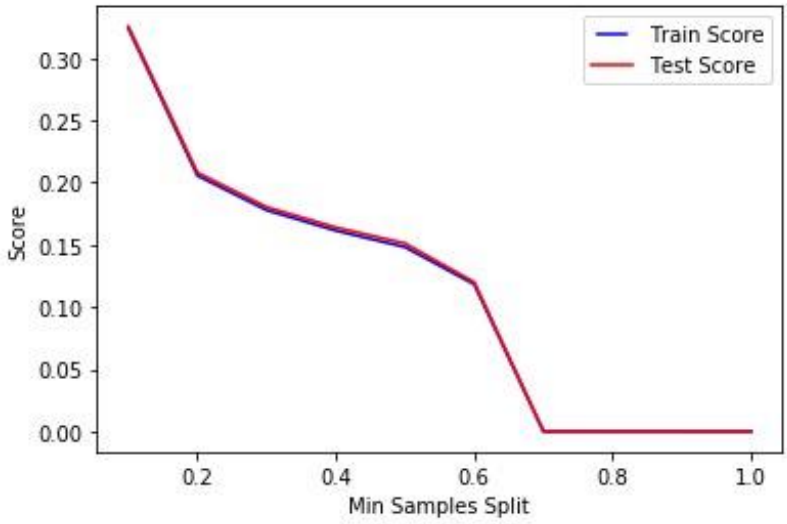


Figure 50 – Minimum Samples Split v/s Score

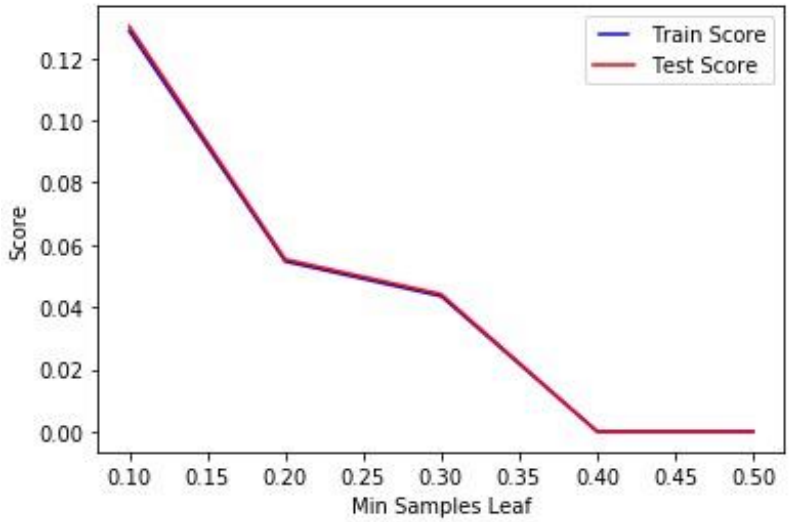


Figure 51 – Minimum Samples Leaf v/s Score

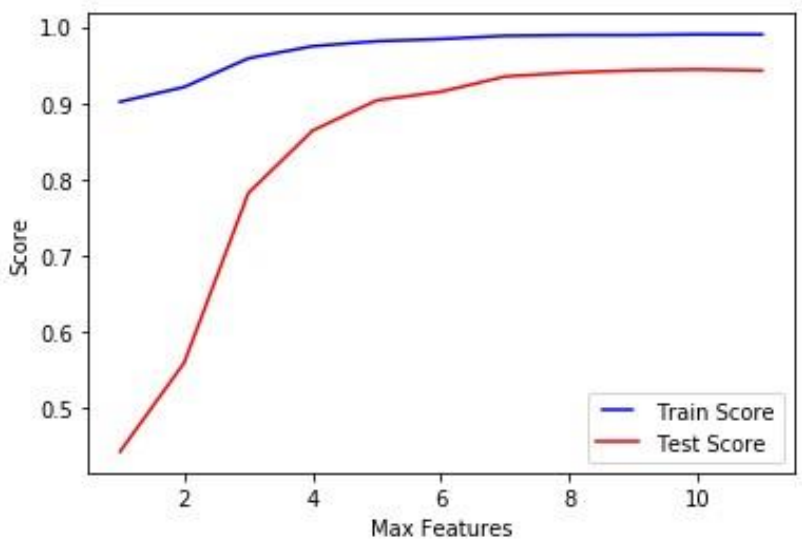


Figure 52 – Maximum Features v/s Score

Recommendations and Limitations

- To focus on low weekly sales for departments lying on bottom-line.
- The weekly sales show up and down trend, which implies that there is seasonality seen in the weekly sales, so according to season-wise the products are sold. But this doesn‘t give us an insight as to what products are sold region-wise
- Departments that had low sales have to be monitored in terms of what products are sold season-wise.
- By applying four P‘s like Price, Product, Promotion and Place on stores with low weekly sales, there is no improvement shown on the sales, hence those stores can be shut.
- Since there are missing values for a year in promotional MarkDowns, locations of sales are not given, department names are not disclosed, we cannot strongly recommend this model in the future.

Closing Reflections

- Data was not complete, there was huge amount of missing data, data was not sufficient to make better predictions.
- Time series analysis can be performed on this data, since there was a Date feature.
- By applying four P‘s like Price, Product, Promotion and Place on stores with low weekly sales, there is no improvement shown on the sales, hence those stores can be shut.
- In general we see that, when discounts are given more often, then the sales increase automatically. Since there is no information on MarkDown for one of the years, we couldn‘t gain a better understanding on the effect of these discounts on weekly sales, if this data was complete, we could understand the effect of discounts on the sales.

References

1. Mining big data: current status, and forecast to the future. ACM SIGKDD Explorations Newsletter, Fan W, Bifet A (2013).

2. How ―big data ―is different. MIT Sloan Management Review, 54(1), pp. 22 Davenport T, Barth P, Bean R (2012).

3. Big data: the management revolution, Harvard Business Review, 90(10), pp. 1, McAfee A, Brynjolfsson, E. (2012).

4. Retail Analytics: Driving Success in Retail Industry with Business Analytics Sudeep B. Chandramana (2017)

5.http://www.emerce.nl/nieuws/personalisatieconversiekrachtdata?utm_source=rss&utm_medim=rss&utm_campaign=personalisatie-conversiekracht-data

6. https://www.mordorintelligence.com/industry-reports/retail-industry