Regression Analysis on Future Sales Prediction

Kanike Lakshmi Narayana

Data science Trainee at

AlmaBetter

Abstract

Predicting sales performance is one of the key challenges every business faces. It is important for firms to predict customer demands to offer the right product at the right time and at the right place. In this project we are going to build a model that will predict the sales for future and help to estimate the stock to be made available in the stores to meet the demand.

Problem Statement

Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

You are provided with historical sales data for 1,115 Rossmann stores. The task is to forecast the "Sales" column for the test set. Note that some stores in the dataset were temporarily closed for refurbishment.

Introduction

The demand for a product or service keeps changing from time to time. No business can improve its financial performance without estimating customer demand and future sales of products/services accurately. Sales forecasting refers to the process of estimating demand for or sales of a particular product over a specific period of time.

Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance.

Machine Learning is concerned with computer programs that automatically improve their performance through experience. In machine learning we have supervised learning, unsupervised learning and reinforcement learning. Again the supervised learning is further divided into regression and classification. In this project we are going to look after the supervised learning regression model. This regression model is used when we have to predict something from the continuous valued output (prices).

In the given dataset we have the sales as our continuous valued output otherwise called as dependent variable.

Objective

The main objective of the project is to build a model to predict the daily sales of stores up to six weeks in advance.

Dataset Peeping

The given dataset consists of 2 separate datasets named as stores and sales. Coming to the size of the dataset the sales data is large and stores data consists of 1115 rows and 10 columns of data. The following steps are performed for the analysis purpose:

- We have NaN values in the store's dataset.
- Changed the format of the Date.
- Added some columns which are extracted from the Date column.

Data Description

Most of the fields are self-explanatory. The following are descriptions for those that aren't.

- Id - an Id that represents a (Store, Date) duple within the test set
- Store - a unique Id for each store
- Sales - the turnover for any given day
- Customers - the number of customers on a given day
- Open - an indicator for whether the store was open: 0 = closed, 1 = open
- StateHoliday - indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. a = public holiday, b = Easter holiday, c = Christmas, 0 = None
- SchoolHoliday - indicates if the (Store, Date) was affected by the closure of public schools
- StoreType - differentiates between 4 different store models: a, b, c, d
- Assortment - describes an assortment level: a = basic, b = extra, c = extended
- CompetitionDistance - distance in meters to the nearest competitor store
- CompetitionOpenSince[Month/Year] - gives the approximate year and month of the time the nearest competitor was opened
- Promo - indicates whether a store is running a promo on that day
- Promo2 - Promo2 is a continuing and consecutive promotion for some stores: 0 = store is not participating, 1 = store is participating

- Promo2Since[Year/Week] - describes the year and calendar week when the store started participating in Promo2
- PromoInterval - describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb, May, Aug, Nov" means each round starts in February, May, August, November of any given year for that store

Challenges Faced

The following are the challenges faced in the data analysis:

- The dataset contains two datasets of stores and sales.
- Added some of the columns.
- Conversion of Datetime features, categorical features.
- Merging of datasets

Approach

As the problem statement says the main object is to predict the daily sales for up to six weeks in advance and have the continuous data in the output feature, used the supervised learning regression analysis Linear Regression, Random Forest Regression for the purpose of training the dataset to predict future sales.

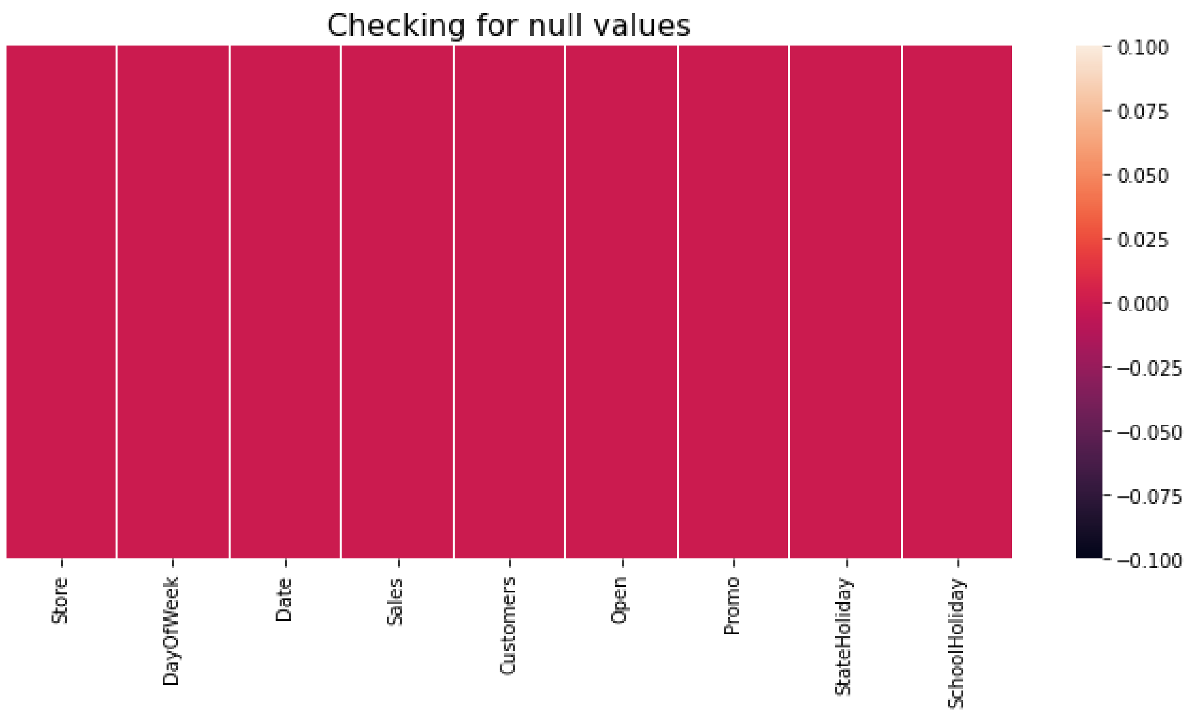
Tools Used

The whole project was done using python, in google colaboratory. Following libraries were used for analysing the data and visualizing it and to build the model to predict the sales:

- Pandas: Extensively used to load and wrangle with the dataset.
- Matplotlib: Used for visualization.
- Seaborn: Used for visualization.
- Datetime: Used for analysing the date variable.
- Warnings: For filtering and ignoring the warnings.
- Numpy: For some math operations in predictions.
- Sklearn: For the purpose of analysis and prediction.
- Datetime: For reading the date.
- Statsmodels: For outliers influence.

Exploratory Analysis on Sales dataset

Visualizing the presence of NaN values in Sales dataset



The above figure shows that there are no NaN (Not a Number) values in the given dataset.

Pandas DataFrame

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1
4	5	5	2015-07-31	4822	559	1	1	0	1

The table shows the dataset in the form of Pandas DataFrame.

The dataset has 1017209 rows and 9 columns wholly the shape of (1017209, 9). It contains the following columns:

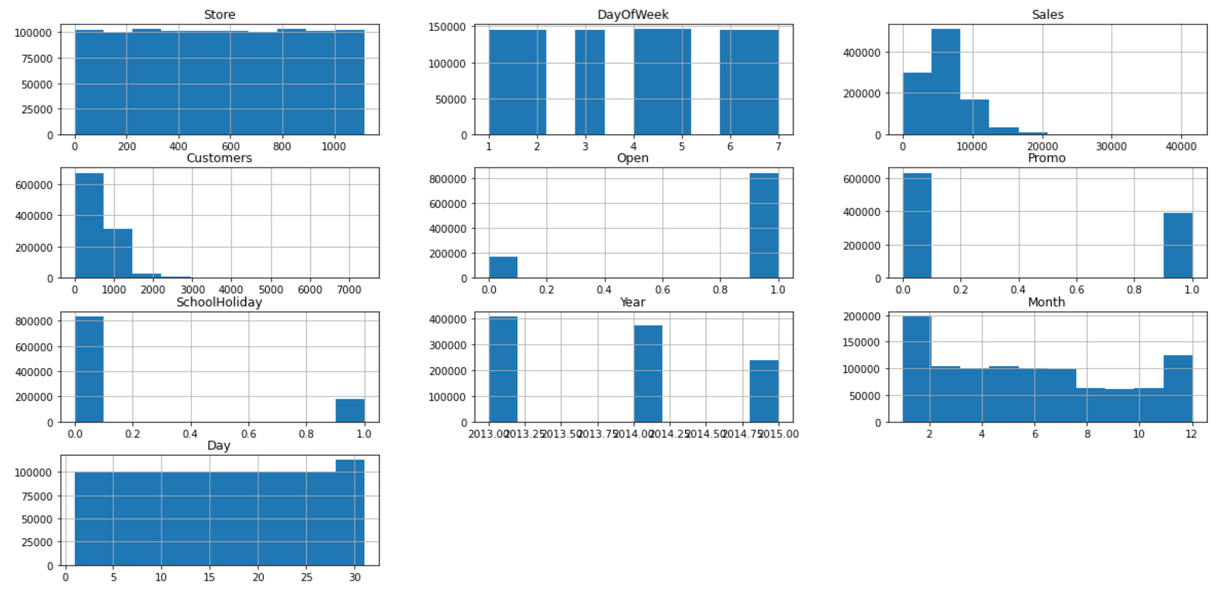
Store	Day of week
Date	Sales

Customers Open

Promo StateHoliday

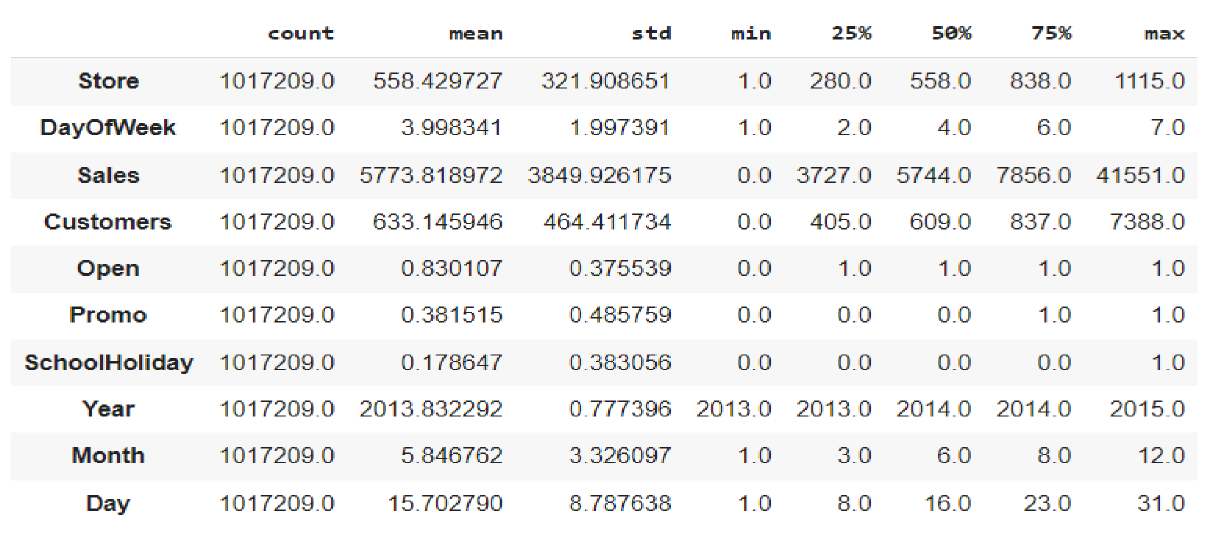
SchoolHoliday

Histogram representation of the data



The above figure depicts the distribution of all the columns of the dataset in a separate histogram and shows the density of such columns. It includes some extra columns that are added for analysis purposes.

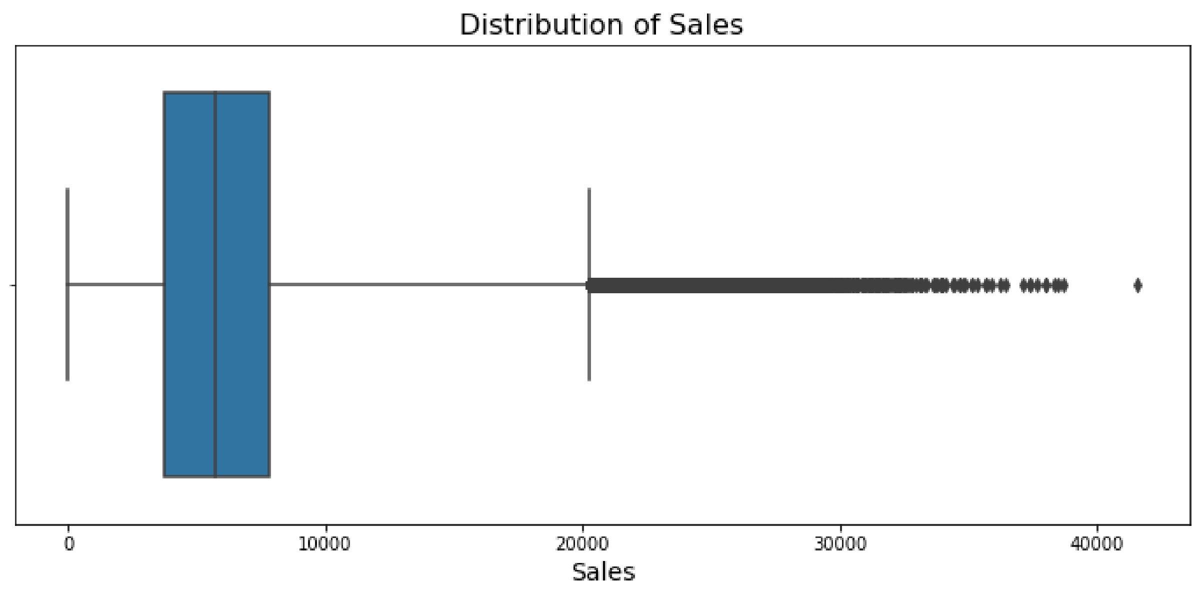
Description



	count	mean	std	min	25%	50%	75%	max
Store	1017209.0	558.429727	321.908651	1.0	280.0	558.0	838.0	1115.0
DayOfWeek	1017209.0	3.998341	1.997391	1.0	2.0	4.0	6.0	7.0
Sales	1017209.0	5773.818972	3849.926175	0.0	3727.0	5744.0	7856.0	41551.0
Customers	1017209.0	633.145946	464.411734	0.0	405.0	609.0	837.0	7388.0
Open	1017209.0	0.830107	0.375539	0.0	1.0	1.0	1.0	1.0
Promo	1017209.0	0.381515	0.485759	0.0	0.0	0.0	1.0	1.0
SchoolHoliday	1017209.0	0.178647	0.383056	0.0	0.0	0.0	0.0	1.0
Year	1017209.0	2013.832292	0.777396	2013.0	2013.0	2014.0	2014.0	2015.0
Month	1017209.0	5.846762	3.326097	1.0	3.0	6.0	8.0	12.0
Day	1017209.0	15.702790	8.787638	1.0	8.0	16.0	23.0	31.0

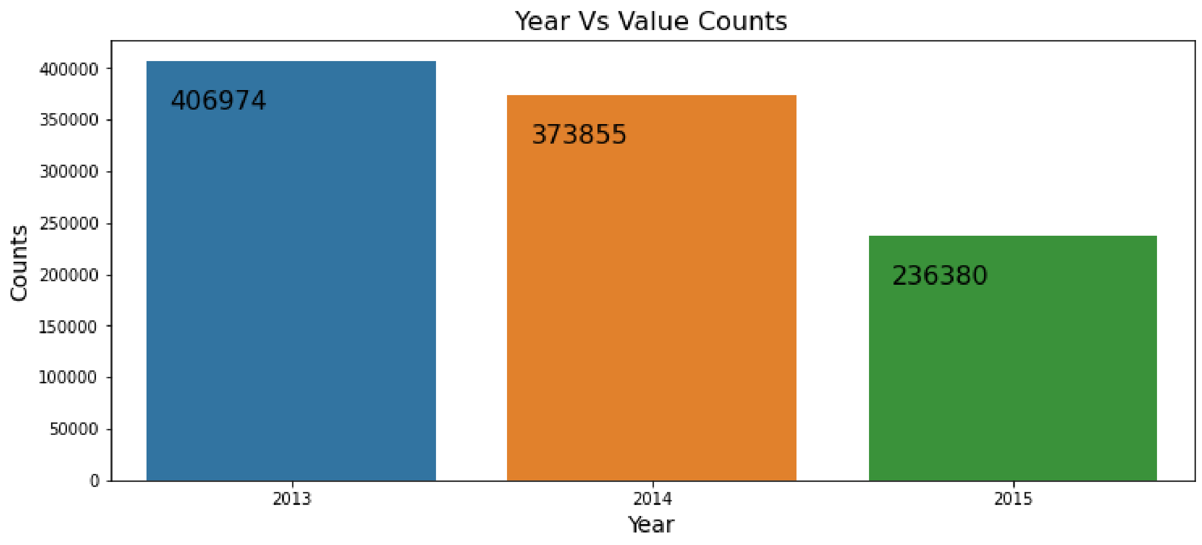
The above table shows the mathematical calculations such as count, mean, standard deviation, minimum and percentiles of all features of the dataset.

Distribution of Sales



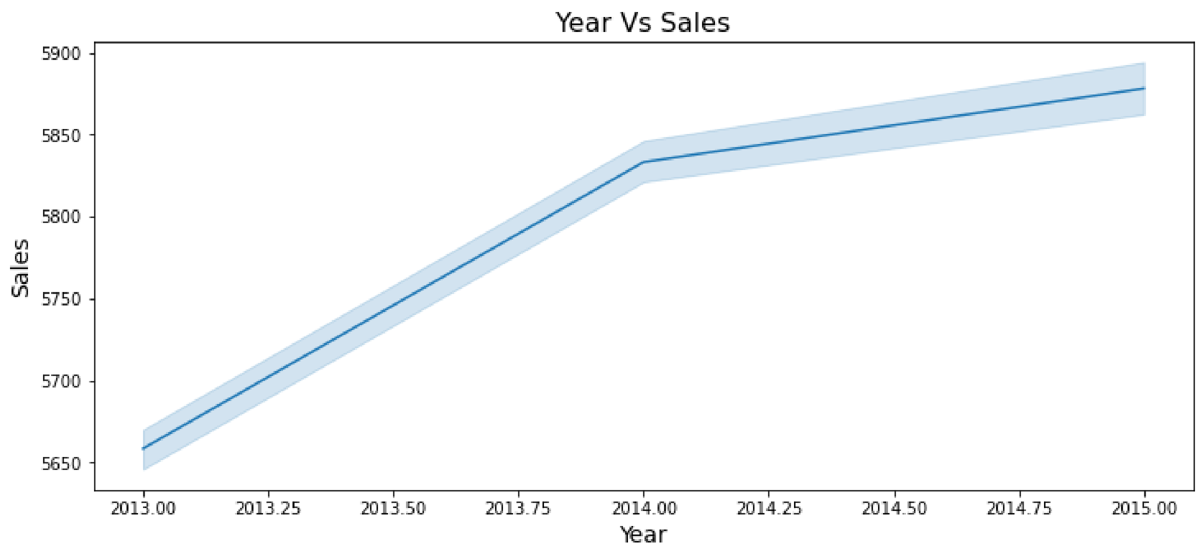
The above boxplot shows the distribution of the sales variable in the dataset, it extends from 0 to 42000, the data we have above 20000 is considered as within the given data and not dropped.

Value counts of Years



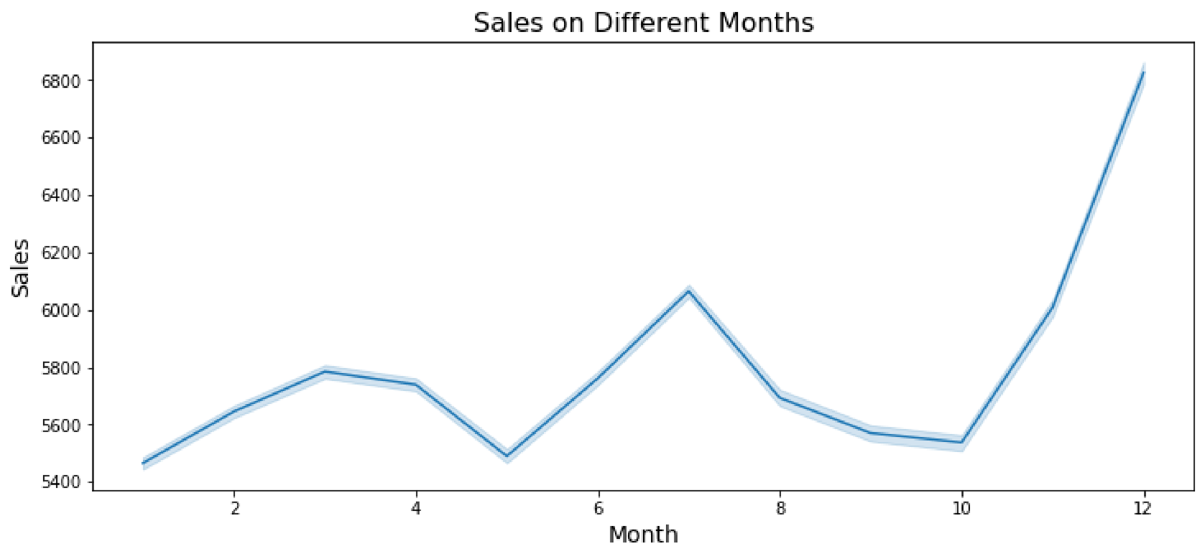
The count plot of value counts of the year shows the dataset has more records which are related to 2013 year and less for 2015 year.

Year vs. Sales



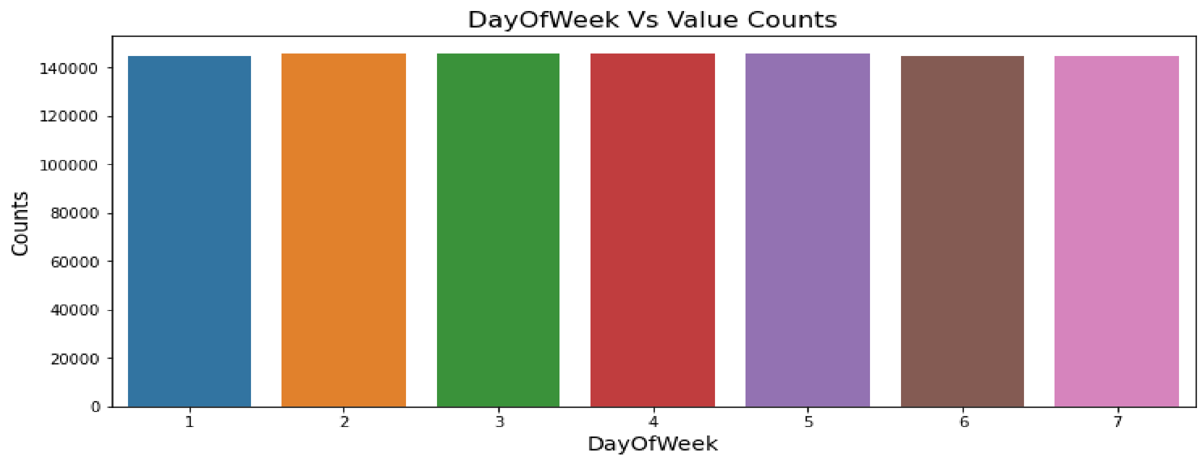
Count plot showed that 2013 had more data but the sales are increasing year by year, even though we have less counts for 2015 year the sales are at high in value.

Month vs. Sales



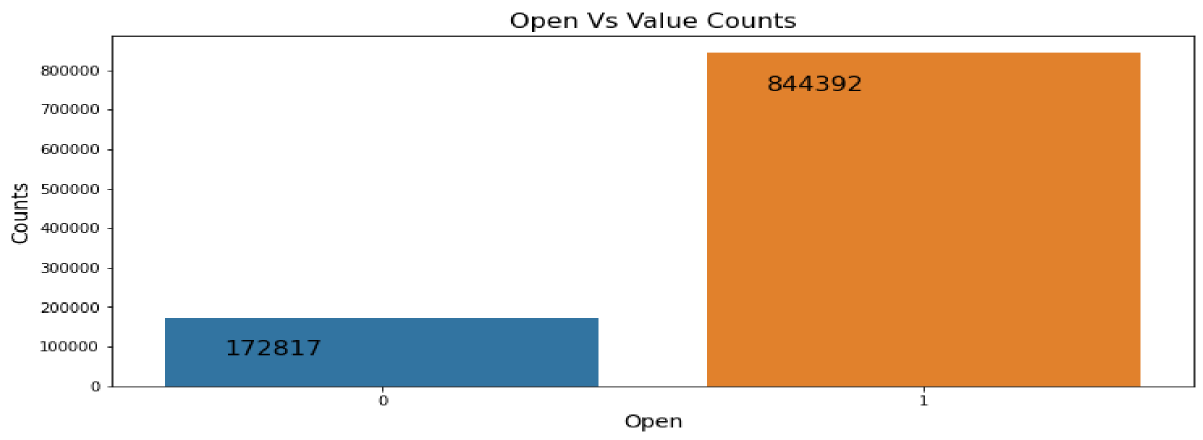
The line plot of sales vs. months shows that for every 4 months of the period we have increased and decreased in sales. In the months of March, July and December we have high sales in those 4 month periods. In all over months we have maximum sales in the month of December.

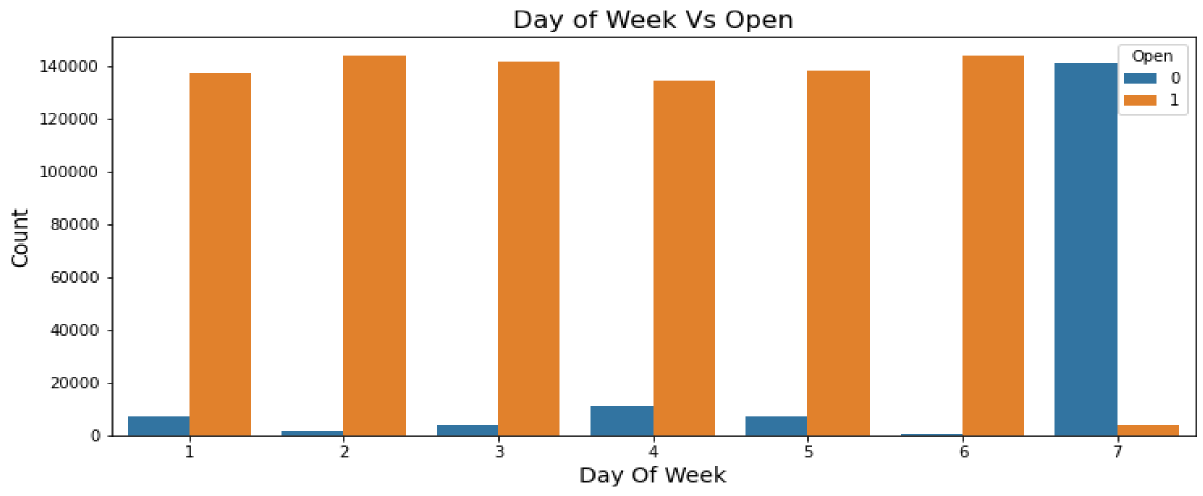
Value counts of Day of week



From the above figure we can see that the data is evenly distributed among all the days in the week.

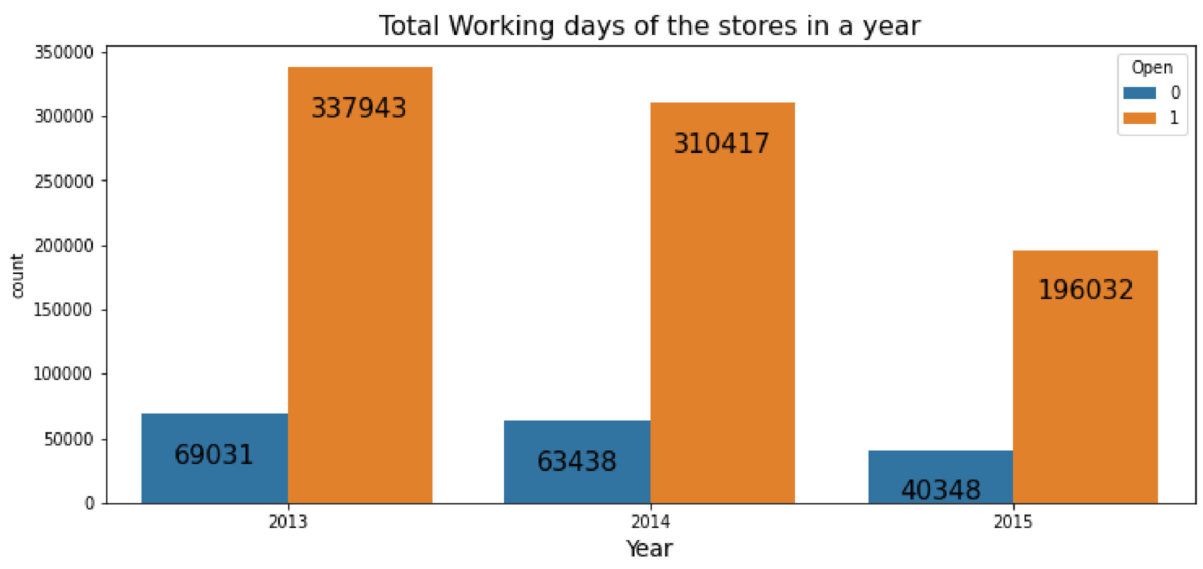
Open and close days of stores





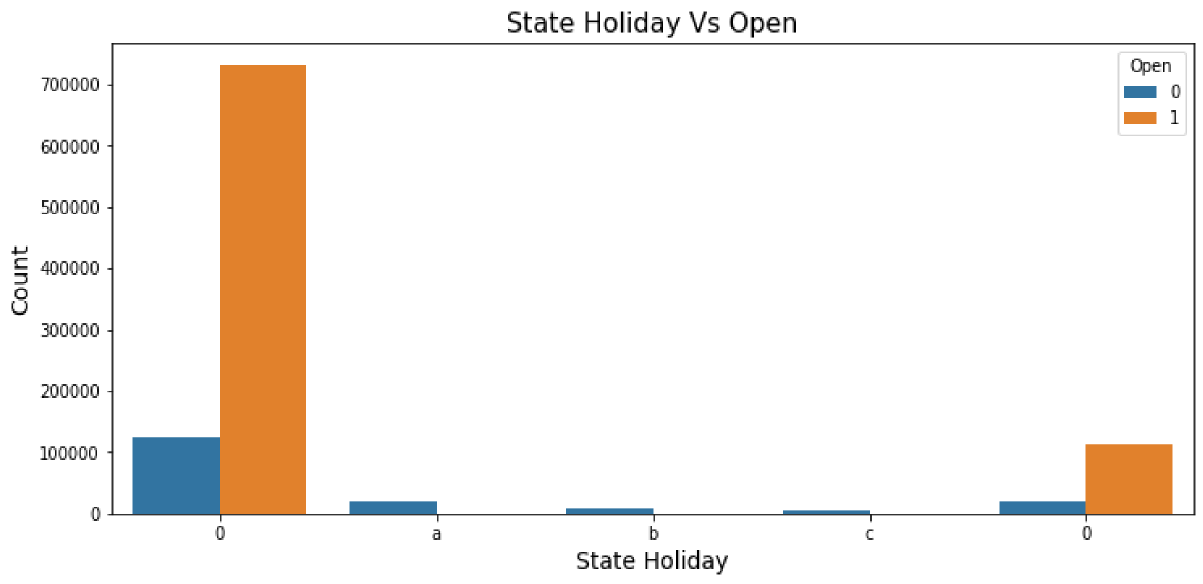
84% of the data belongs to stores that are open. It is assumed that 7 in the day of week is Sunday. The day of week vs. open shows that stores are mostly closed during Sundays.

Year vs. open or close of stores



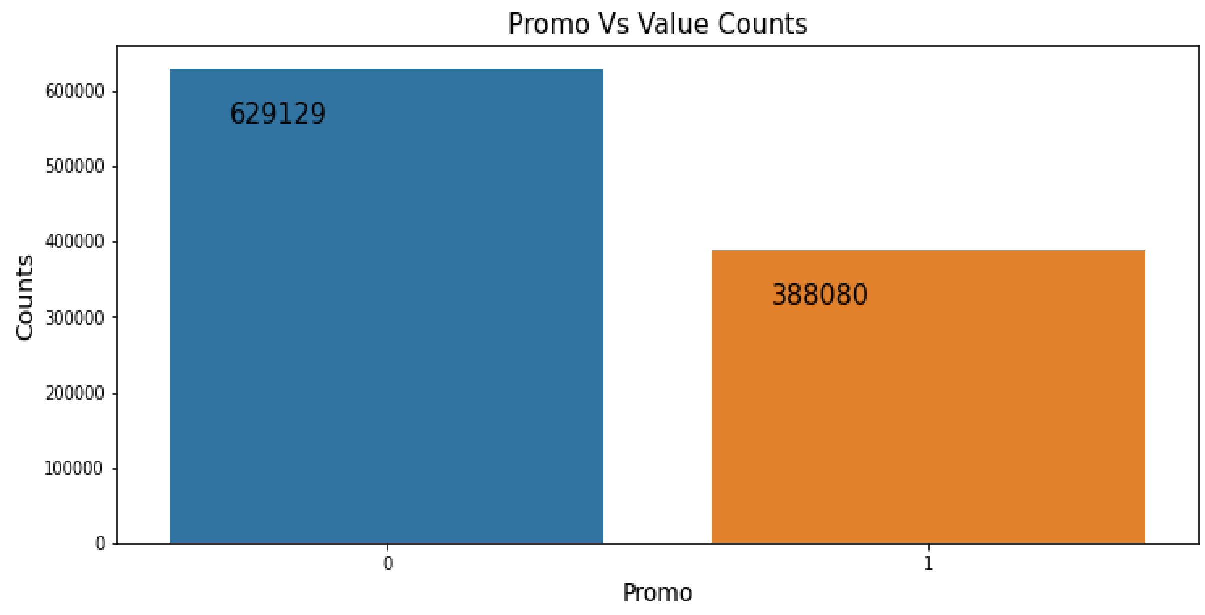
The working days of the stores are less in 2015. It is assumed that we have less records from the year 2015.

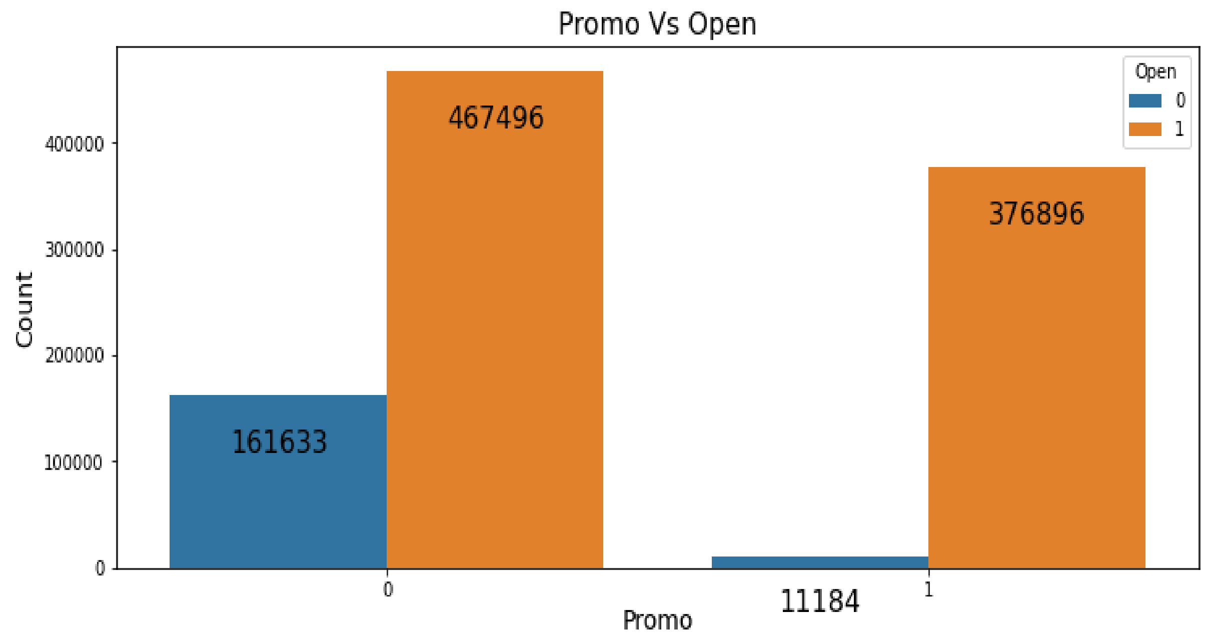
State holidays vs. open



The above figure states a: public holiday, b: Easter holiday and c: Christmas and 0 for none. Normally all stores with few exceptions are closed on state holidays.

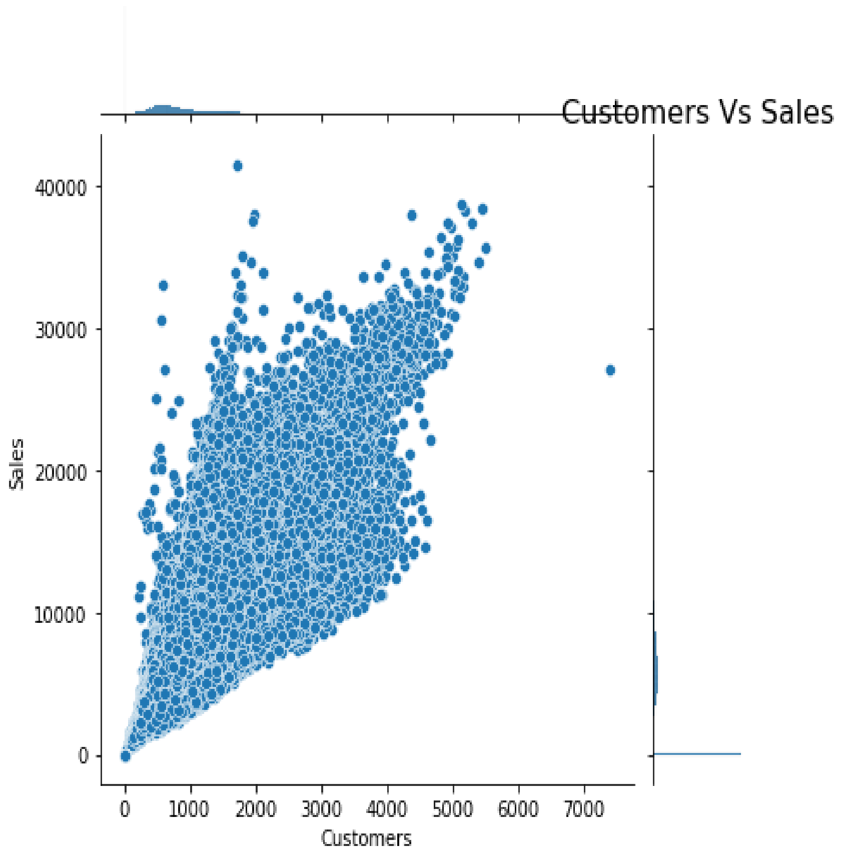
Promo Value counts





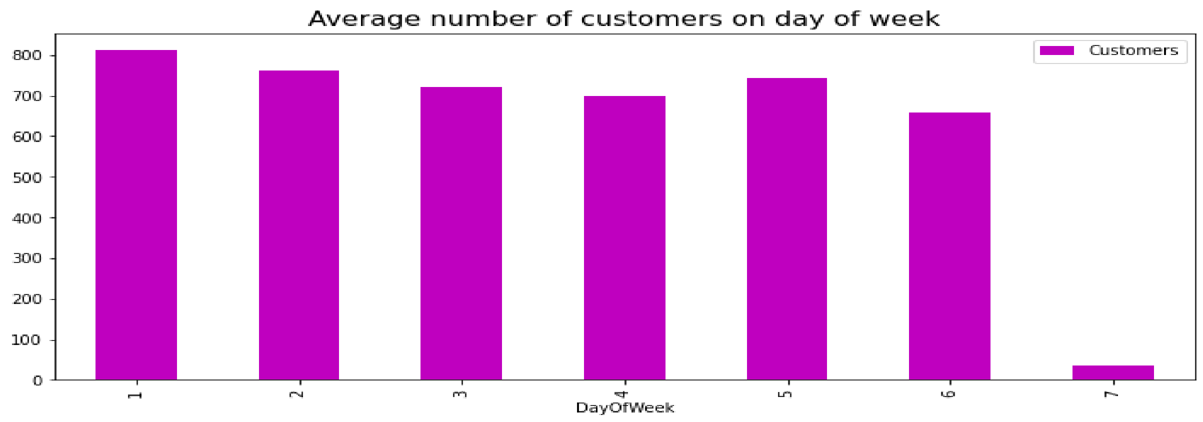
The above figure shows the stores opened or closed during the promo days. It is assumed that holidays are also included in these as the store stands closed during state holidays.

Customer vs. Sales



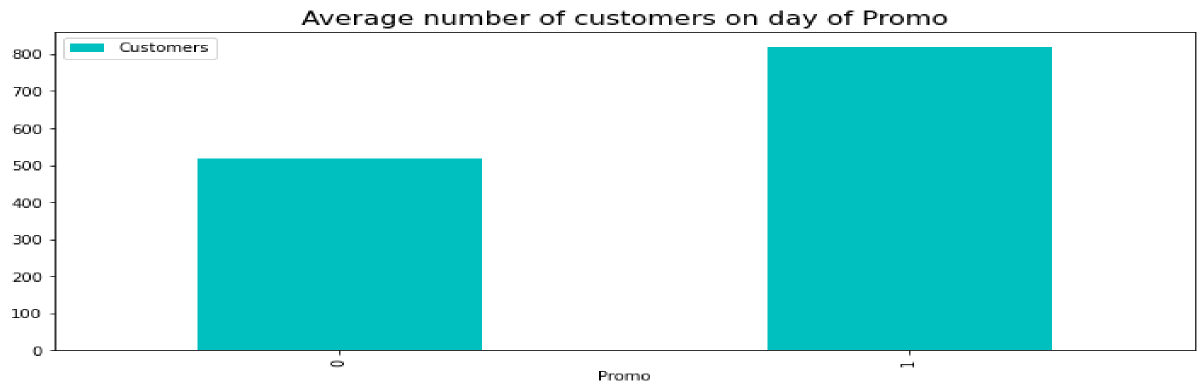
The above joint plot shows the relationship between the customers and sales. It shows that we have more customers in the range of 0 to 2000 with the sales range of 0 to 10000.

Customer vs. Day of week



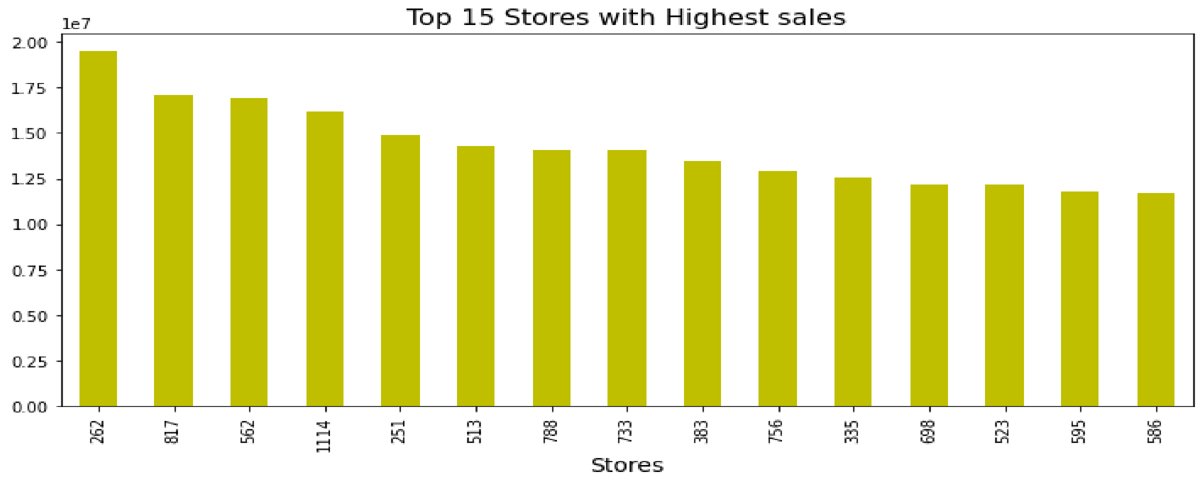
The above bar plot shows the average customer visits on each day of week. As we see before mostly the stores are closed on Sundays we have less customer visits on those days.

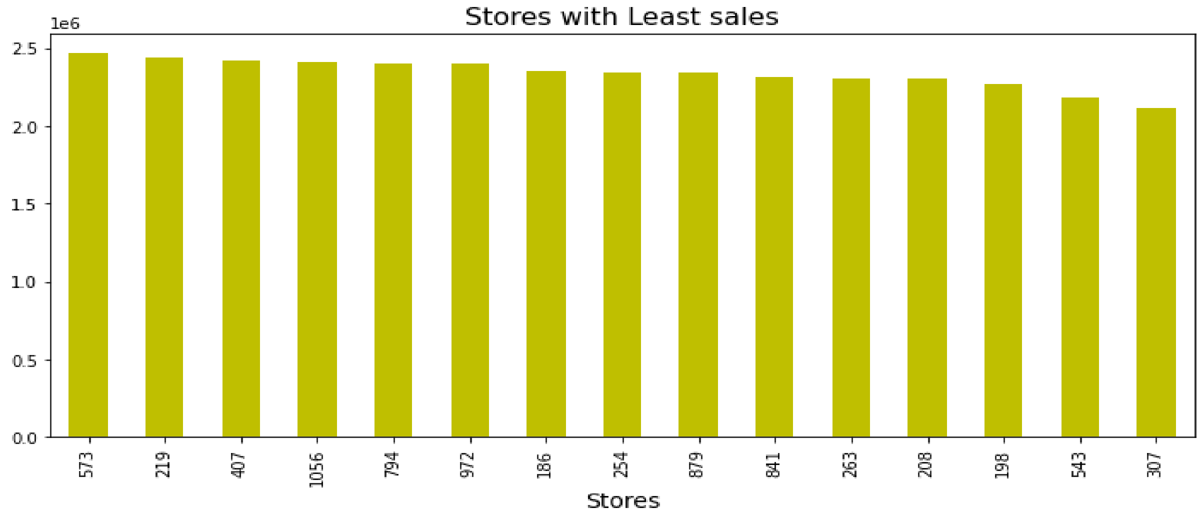
Customer vs. Promo



We have more customers visiting during the promo days than non- promo days.

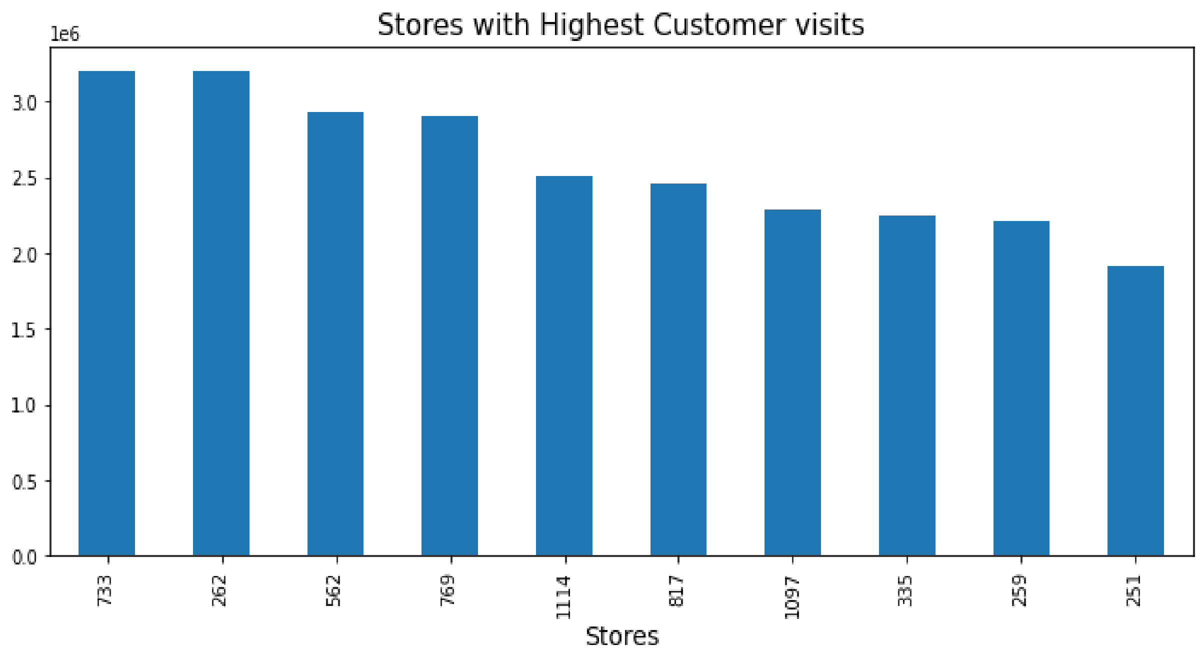
Stores vs. sales

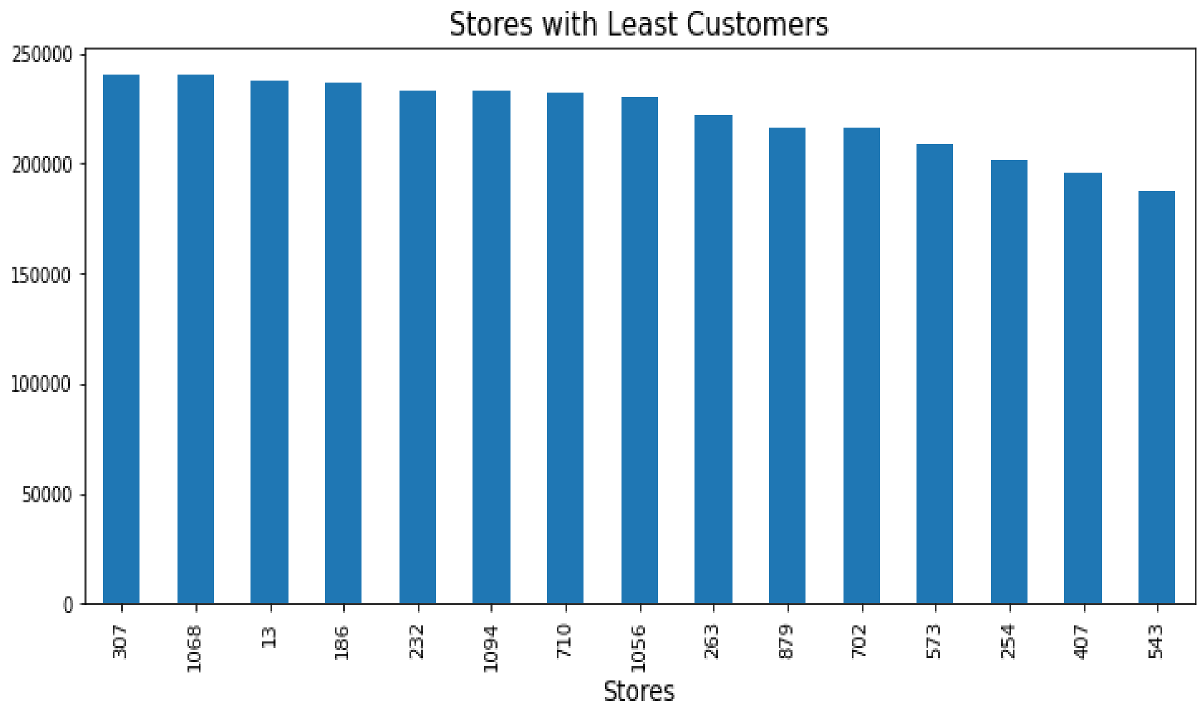




The above bar plots show the stores with highest and least sales.

Stores vs. Customers

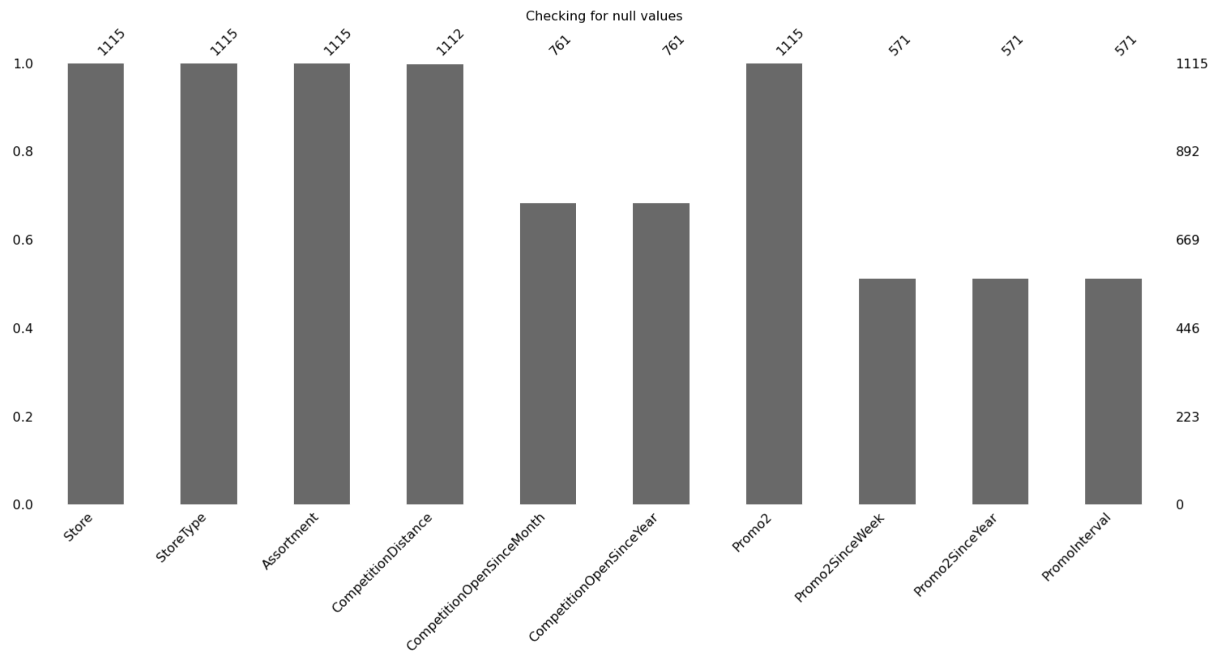




The above 2 bar plots show the stores with highest and least customer visits.

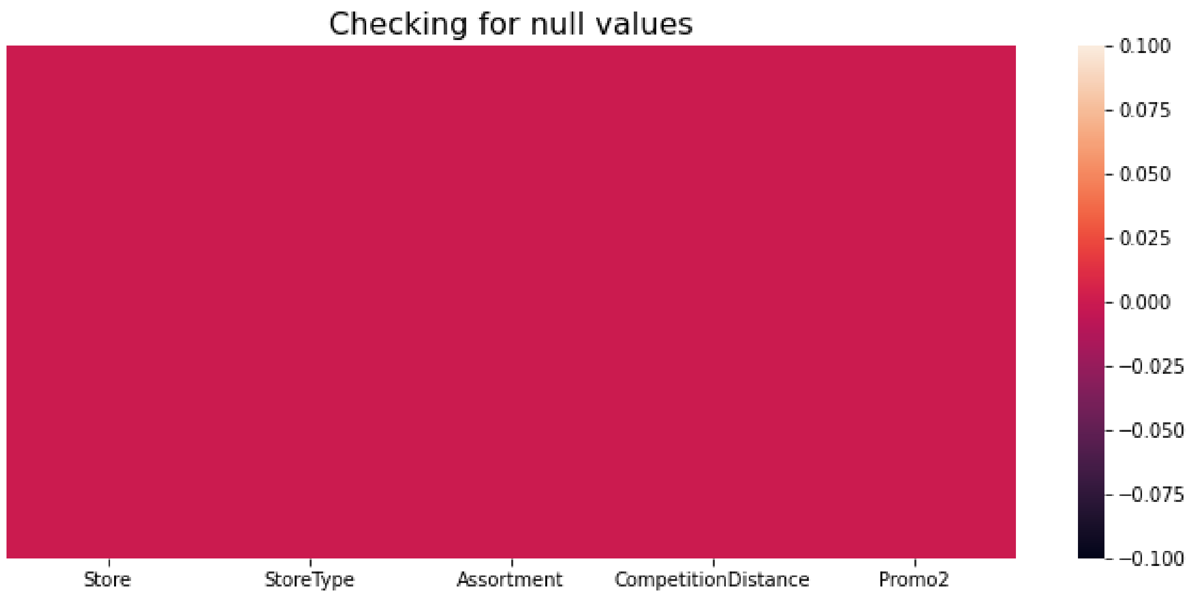
Exploratory Analysis on Stores dataset

Visualizing the presence of NaN values in Stores dataset



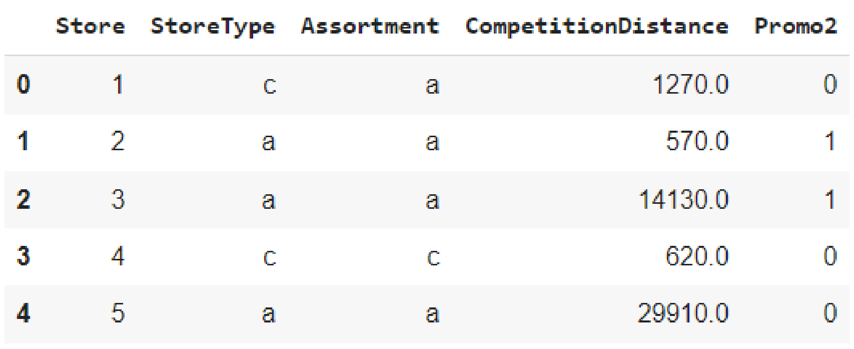
The above bar plot shows the presence of NaN values in Competition Open since Month, Competition Open since Year, Promo2SinceWeek, Promo2SinceYear and PromoInterval.

The above mentioned features are dropped as these contain more null values and to make the analysis easy. Competition distance is having one NaN value and it is replaced with 0 assuming that it does not affect the analysis.



The above figure shows that there are no NaN (Not a Number) values in the given dataset.

Pandas DataFrame

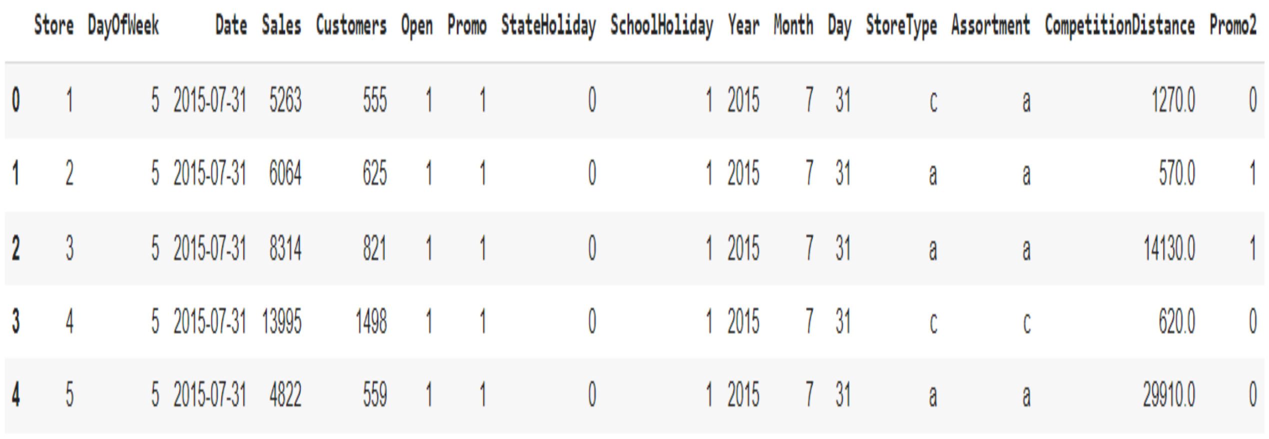


	Store	StoreType	Assortment	CompetitionDistance	Promo2
0	1	c	a	1270.0	0
1	2	a	a	570.0	1
2	3	a	a	14130.0	1
3	4	c	c	620.0	0
4	5	a	a	29910.0	0

The above is the data frame of the stored dataset after the data cleaning. The dataset has 1115 rows and 5 columns total the shape is (1115, 5).

The above two dataset are merged to analyse the datasets and build models to predict the future sales.

Pandas DataFrame

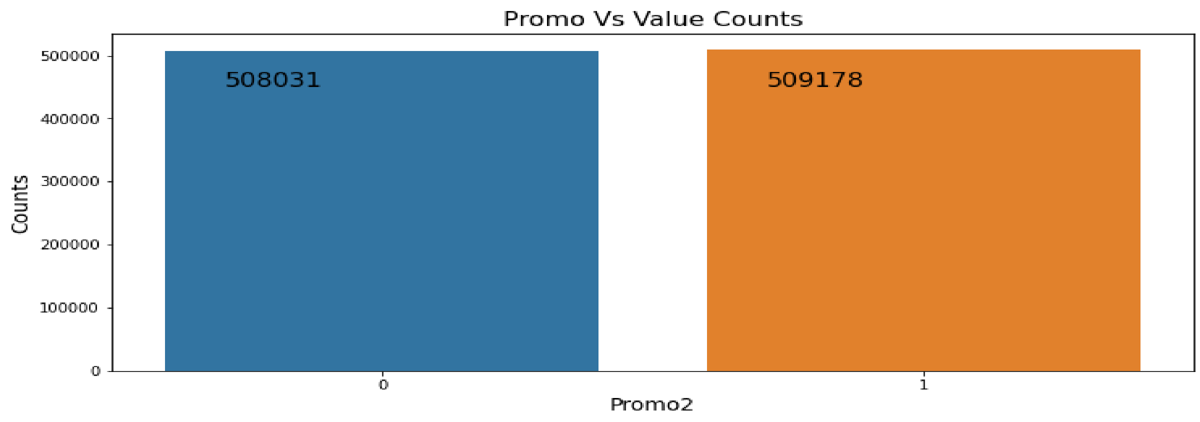


	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday	Year	Month	Day	StoreType	Assortment	CompetitionDistance	Promo2
0	1	5	2015-07-31	5263	555	1	1	0	1	2015	7	31	c	a	1270.0	0
1	2	5	2015-07-31	6064	625	1	1	0	1	2015	7	31	a	a	570.0	1
2	3	5	2015-07-31	8314	821	1	1	0	1	2015	7	31	a	a	14130.0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1	2015	7	31	c	c	620.0	0
4	5	5	2015-07-31	4822	559	1	1	0	1	2015	7	31	a	a	29910.0	0

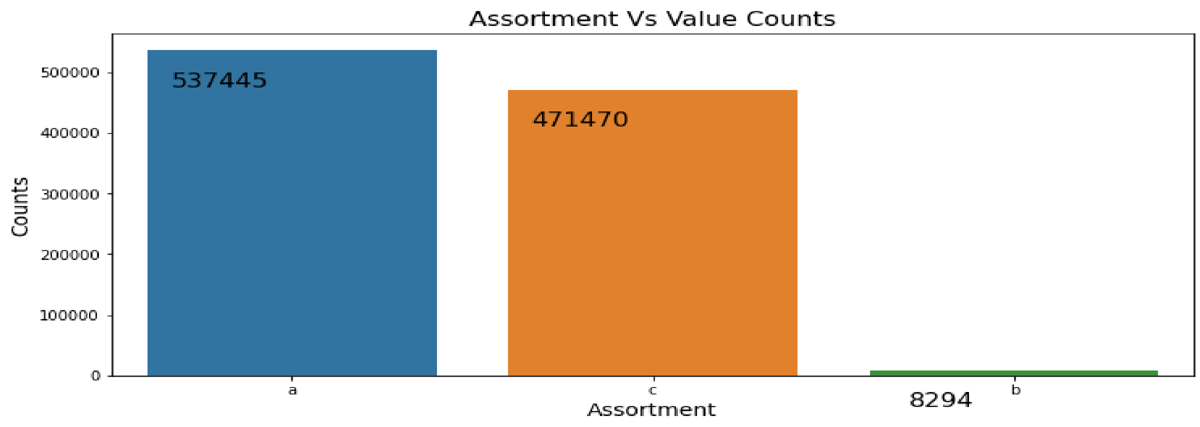
The table shows the stores dataset in the form of Pandas DataFrame.

The dataset has 1017209 rows and 16 columns wholly the shape of (1017209, 16). The above figure shows the features in it.

Value counts of Promo2

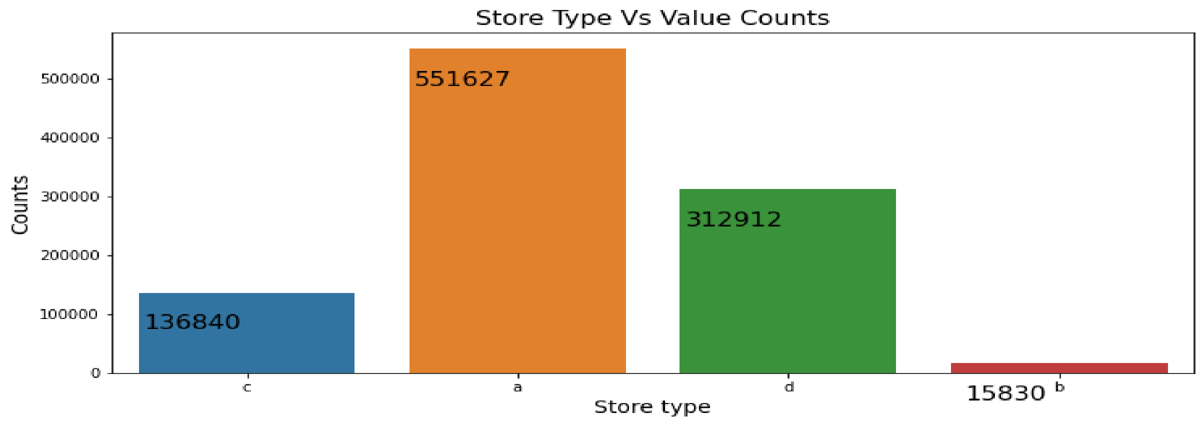


Assortment vs. value counts



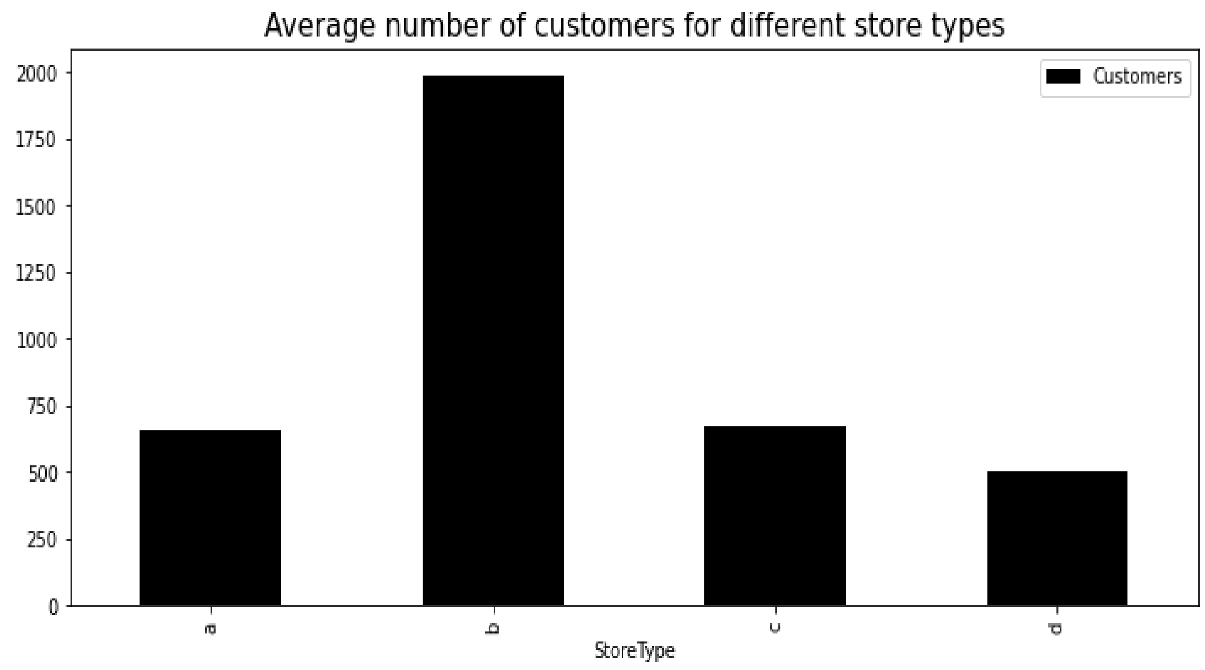
Describes the assortment level a: basic, b: extra, c: extended.

Store type vs. value counts

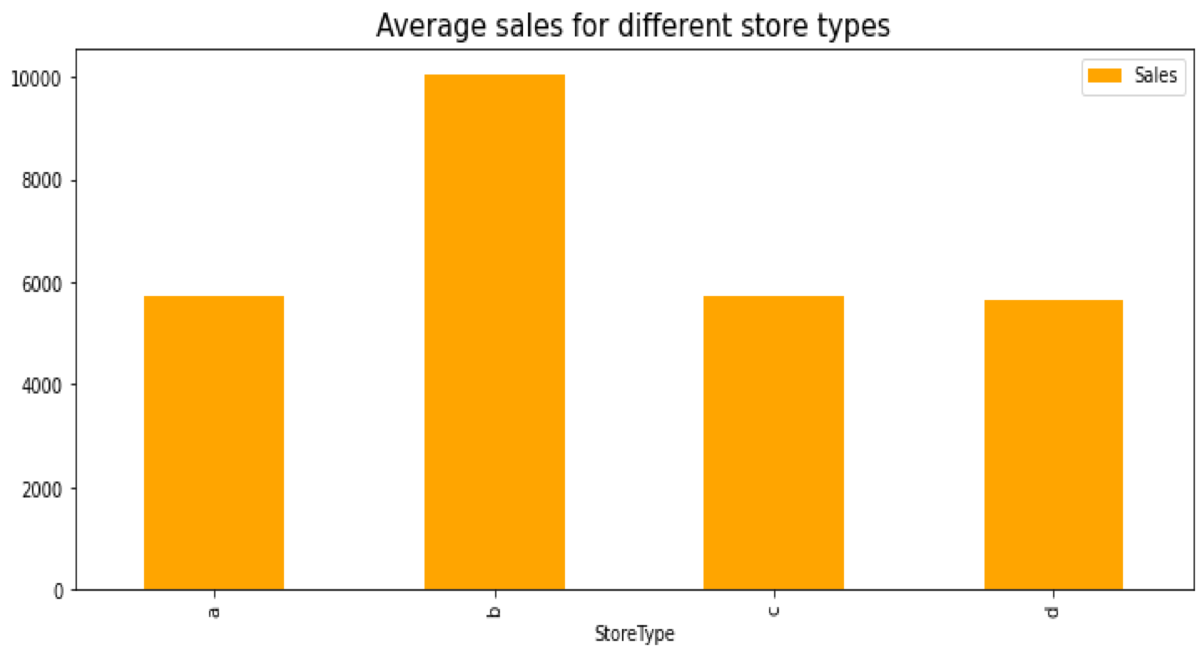


Differentiates between 4 different store models: a, b, c and d. Store type a has more records in the dataset.

Store type vs. Customers

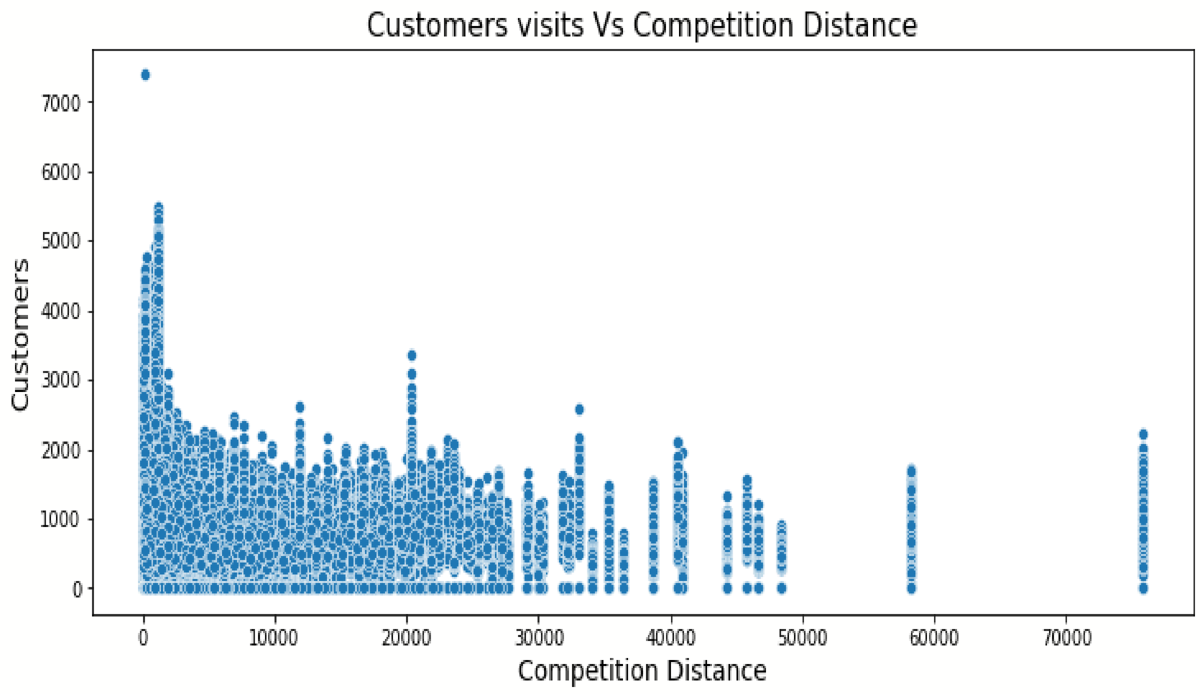


Store type vs. sales



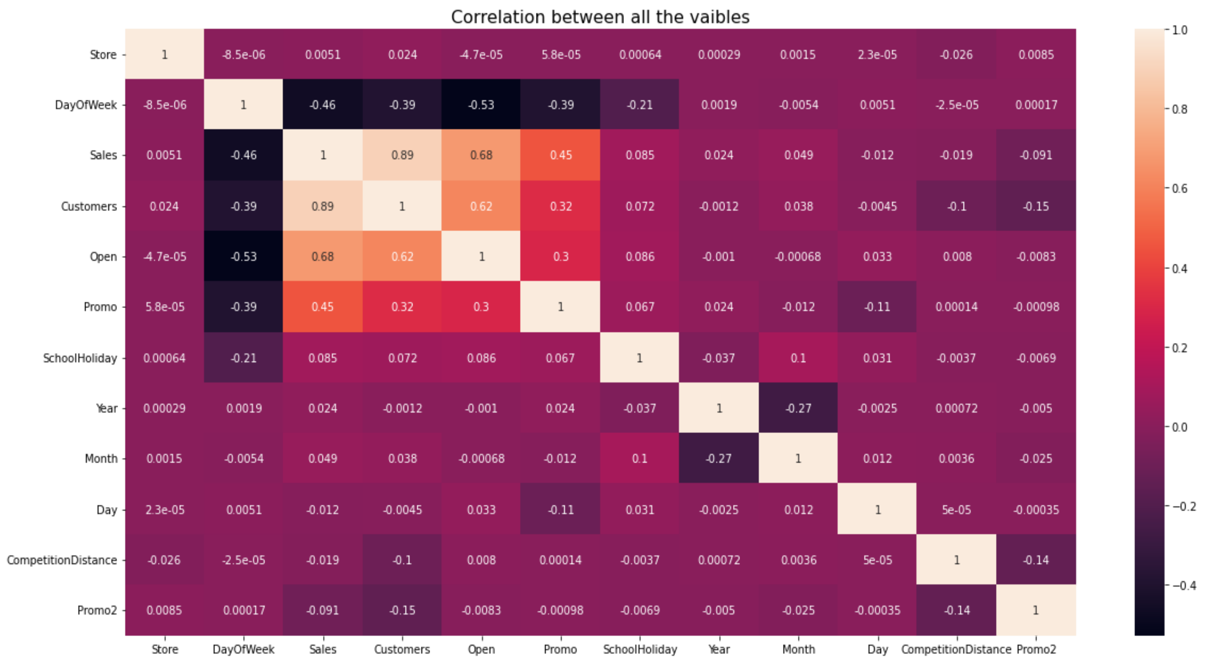
Even though store type ‘a’ is having more value counts in the dataset, store type b is having the more number of customer visits and sales.

Customers visits vs. Competition Distance



The competition distance shows the distance in meters to the nearest competitor store. We have the competitor stores mostly in the range of 0 to 30000 meters of distance and there is diversion of customers from our stores.

Correlation Heatmap



	Store	DayOfWeek	Sales	Customers	Open	Promo	SchoolHoliday	Year	Month	Day	CompetitionDistance	Promo2
Store	1	-8.5e-06	0.0051	0.024	-4.7e-05	5.8e-05	0.00064	0.00029	0.0015	2.3e-05	-0.026	0.0085
DayOfWeek	-8.5e-06	1	-0.46	-0.39	-0.53	-0.39	-0.21	0.0019	-0.0054	0.0051	-2.5e-05	0.00017
Sales	0.0051	-0.46	1	0.89	0.68	0.45	0.085	0.024	0.049	-0.012	-0.019	-0.091
Customers	0.024	-0.39	0.89	1	0.62	0.32	0.072	-0.0012	0.038	-0.0045	-0.1	-0.15
Open	-4.7e-05	-0.53	0.68	0.62	1	0.3	0.086	-0.001	-0.00068	0.033	0.008	-0.0083
Promo	5.8e-05	-0.39	0.45	0.32	0.3	1	0.067	0.024	-0.012	-0.11	0.00014	-0.00098
SchoolHoliday	0.00064	-0.21	0.085	0.072	0.086	0.067	1	-0.037	0.1	0.031	-0.0037	-0.0069
Year	0.00029	0.0019	0.024	-0.0012	-0.001	0.024	-0.037	1	-0.27	-0.0025	0.00072	-0.005
Month	0.0015	-0.0054	0.049	0.038	-0.00068	-0.012	0.1	-0.27	1	0.012	0.0036	-0.025
Day	2.3e-05	0.0051	-0.012	-0.0045	0.033	-0.11	0.031	-0.0025	0.012	1	5e-05	-0.00035
CompetitionDistance	-0.026	-2.5e-05	-0.019	-0.1	0.008	0.00014	-0.0037	0.00072	0.0036	5e-05	1	-0.14
Promo2	0.0085	0.00017	-0.091	-0.15	-0.0083	-0.00098	-0.0069	-0.005	-0.025	-0.00035	-0.14	1

The above heatmap shows the correlation of all variables in the merged dataset.

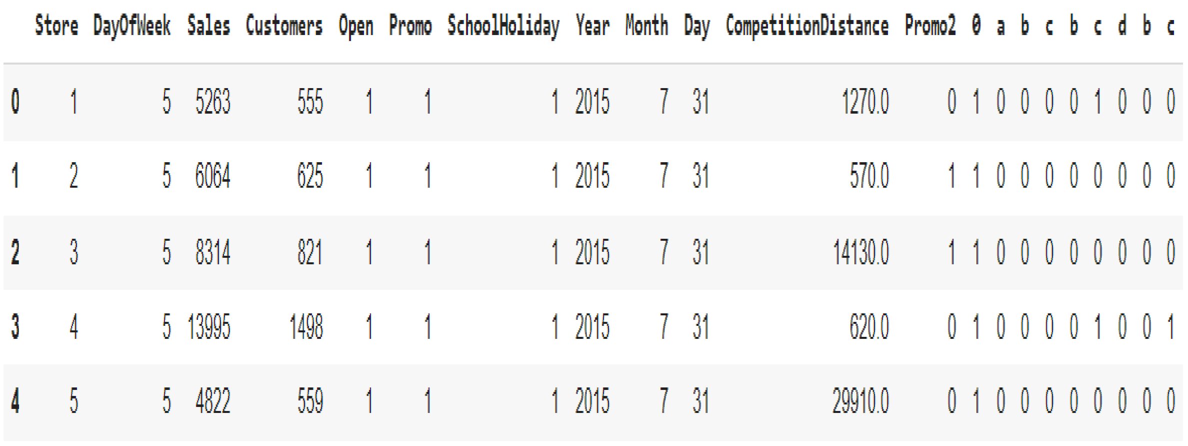
Variance Inflation Factor

The Variance Inflation Factor is used when we have the multi-collinearity between the features. The factor helps in reducing the inflation between the features by dropping some of the features which are having the high correlation among them.

In the given dataset we don't have much correlation between the features and the features which are dropped in the data column to reduce the linearity among them and also the data in it was splitted.

One hot encoding

State holiday, store type and assortment are the columns having categorical features. These are converted into binary format. After creating the dummy variables to these features the main features containing categorical format are dropped. The below is the data frame after the data wrangling.



	Store	DayOfWeek	Sales	Customers	Open	Promo	SchoolHoliday	Year	Month	Day	CompetitionDistance	Promo2	0	a	b	c	b	c	d	b	c
0	1	5	5263	555	1	1	1	2015	7	31	1270.0	0	1	0	0	0	0	1	0	0	0
1	2	5	6064	625	1	1	1	2015	7	31	570.0	1	1	0	0	0	0	0	0	0	0
2	3	5	8314	821	1	1	1	2015	7	31	14130.0	1	1	0	0	0	0	0	0	0	0
3	4	5	13995	1498	1	1	1	2015	7	31	620.0	0	1	0	0	0	0	1	0	0	1
4	5	5	4822	559	1	1	1	2015	7	31	29910.0	0	1	0	0	0	0	0	0	0	0

Data Modelling

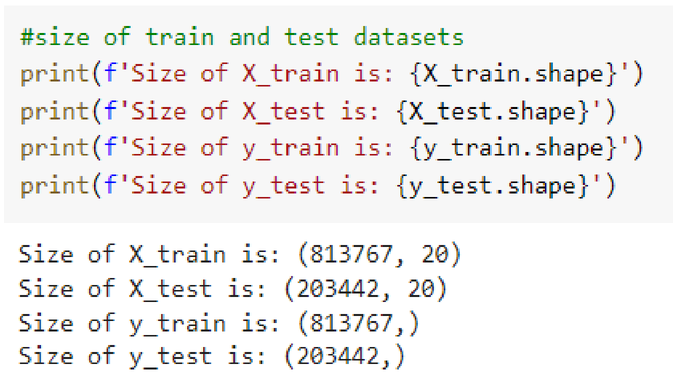
After the data preparation was completed it is ready for the purpose of building the model to predict the sales. Only numerical valued features are taken into consideration. The data was combined and labelled as X and y as independent and dependent variables respectively. The sales feature is taken as dependent variable (y) and remaining all are considered as independent variables (X).

Splitting the data

The train_test_split was imported from the sklearn.model_selection. The data is now divided into 80% and 20% as train and test splits respectively. 80% of the data is taken for training the model and 20% is for test and the random state was taken as 42.

Scaling the data

To normalise the data standardscaler was used from sklearn.preprocessing. It scales the data in the form of standard deviation of the feature divided from the difference of variable and its mean of the feature. At first the training data was made fit into the scaling function and test data is transformed now. The output we get are X_train, X_test, y_train, y_test. The below figure shows the shape of each dataset.



```python
#size of train and test datasets
print(f'Size of X_train is: {X_train.shape}')
print(f'Size of X_test is: {X_test.shape}')
print(f'Size of y_train is: {y_train.shape}')
print(f'Size of y_test is: {y_test.shape}')
```

```
Size of X_train is: (813767, 20)
Size of X_test is: (203442, 20)
Size of y_train is: (813767,)
Size of y_test is: (203442,)
```

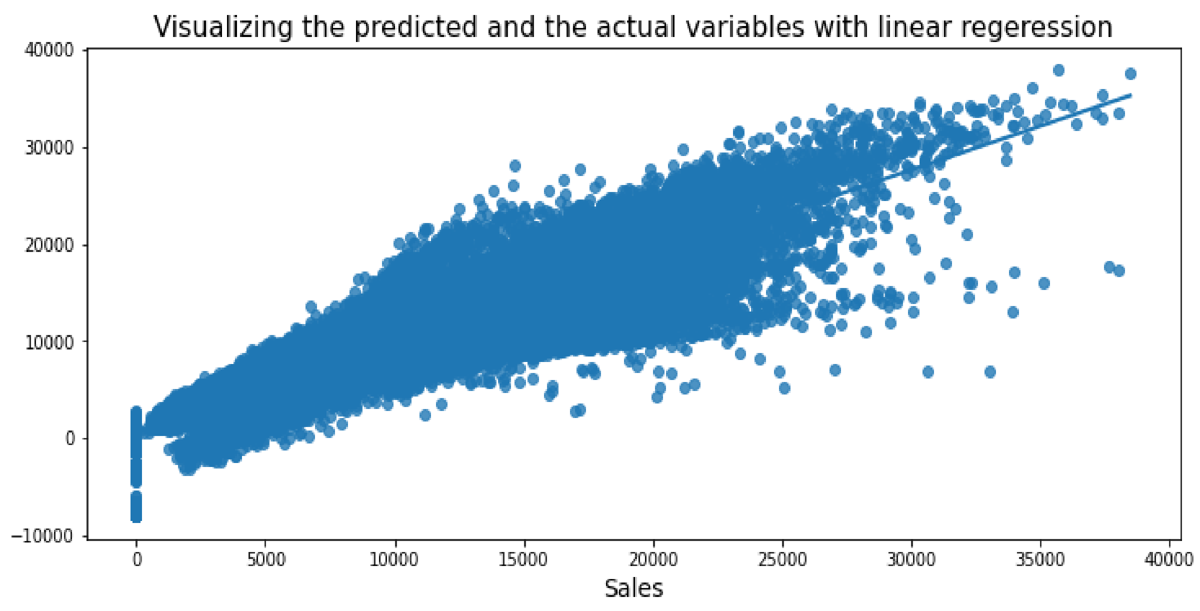
Linear Regression

The next step is implementing and training the model. A linear regression is a type of statistical procedure that involves finding a relationship between a linearly-related variable and a prediction. Mathematically it solves problem in the form of:

$$\min_{w} ||Xw - y||_2^2$$

The first thing we do is to fit the training set and train the model. The score we obtained in training the model is 90.07%. To get the predicted data we fit the independent variables in the regression model and define the predicted variable. The available variables in hand at last are the actual y values and the predicted values. With these we can make the visualizations as follows.

Training data

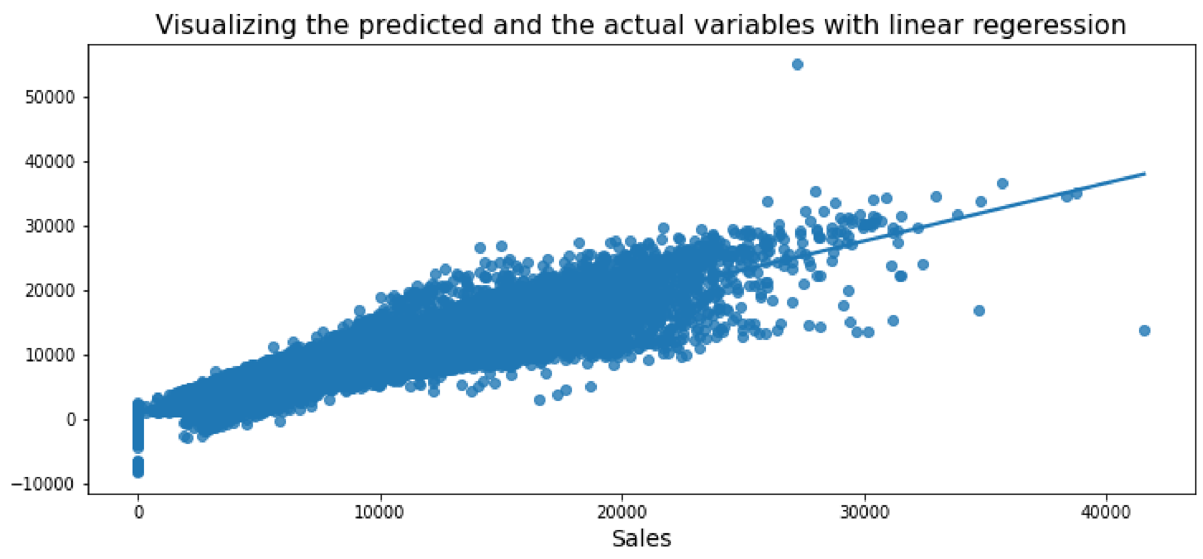


```
#Evaluating the training dataset
#Mean Squared Error
MSE_train = mean_squared_error(y_train, pred_train)
print(f'MSE= {MSE_train}')

#Root Mean Squared Error
RMSE_train = np.sqrt(MSE_train)
print(f'RMSE= {RMSE_train}')

#R2_Score
R2_Score_train = r2_score(y_train, pred_train)
print(f'R2_Score= {R2_Score_train}')
```

```
MSE= 1473080.6552463644
RMSE= 1213.7053411954503
R2_Score= 0.9006701662380696
```

The above figure shows how the model was trained and fit with the data. It consists of both predicted and actual training dataset in it. With the accuracy of 90.07% of the model can predict correctly what the sales are in the near future.

Test dataset



```
#Evaluating the test dataset
#Mean Squared Error
MSE_test = mean_squared_error(y_test, pred_test)
print(f'MSE= {MSE_test}')

#Root Mean Squared Error
RMSE_test = np.sqrt(MSE_test)
print(f'RMSE= {RMSE_test}')

#R2_Score
R2_Score_test = r2_score(y_test, pred_test)
print(f'R2_Score= {R2_Score_test}')
```

```
MSE= 1488637.608160531
RMSE= 1220.0973765075191
R2_Score= 0.8993401767895278
```

The above figure shows how the model was trained and fit with the data. It consists of both predicted and actual data from the test dataset in it. With the accuracy of 89.93% of the model can predict correctly what the sales are in the near future.

Metrics

The metrics are tools used for evaluating the performance of a regression model. The following are the metrics that have been used in the analysis of the data.

Mean Squared Error

The mean squared error (MSE) is a commonly used metric for estimating errors in regression models. It provides a positive value as the error gets closer to zero. It is simply the average of the squared difference between the target value and the value predicted by the regression model.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

MSE score on training and test sets is 1473080.66 and 1488637.61 respectively.

Root Mean Squared Error

The root-mean-Square error or RMSE is a frequently used measure to evaluate the differences between the values that a model has predicted and the values that were observed. It is computed by taking the second sample moment and dividing it by the quadratic mean of the differences.

RMSD is a non-zero measure that shows a perfect fit to the data, and it is generally better than a higher one. It is not used to evaluate the relationships among different types of data.

RMSD is the sum of the average of all errors. It is sensitive to outliers.

$$\mathrm{RMSD}(\hat{\theta}) = \sqrt{\mathrm{MSE}(\hat{\theta})} = \sqrt{\mathrm{E}((\hat{\theta} - \theta)^2)}.$$

RMSE score on training and test sets is 1213.71 and 1220.61 respectively.

R2 score

The goodness-of-fit evaluation should not be performed on the R2 linear regression because it quantifies the degree of linear correlation between the two values. Instead, the linear correlation should only be taken into account when evaluating the Ypred-Yobs relationship.

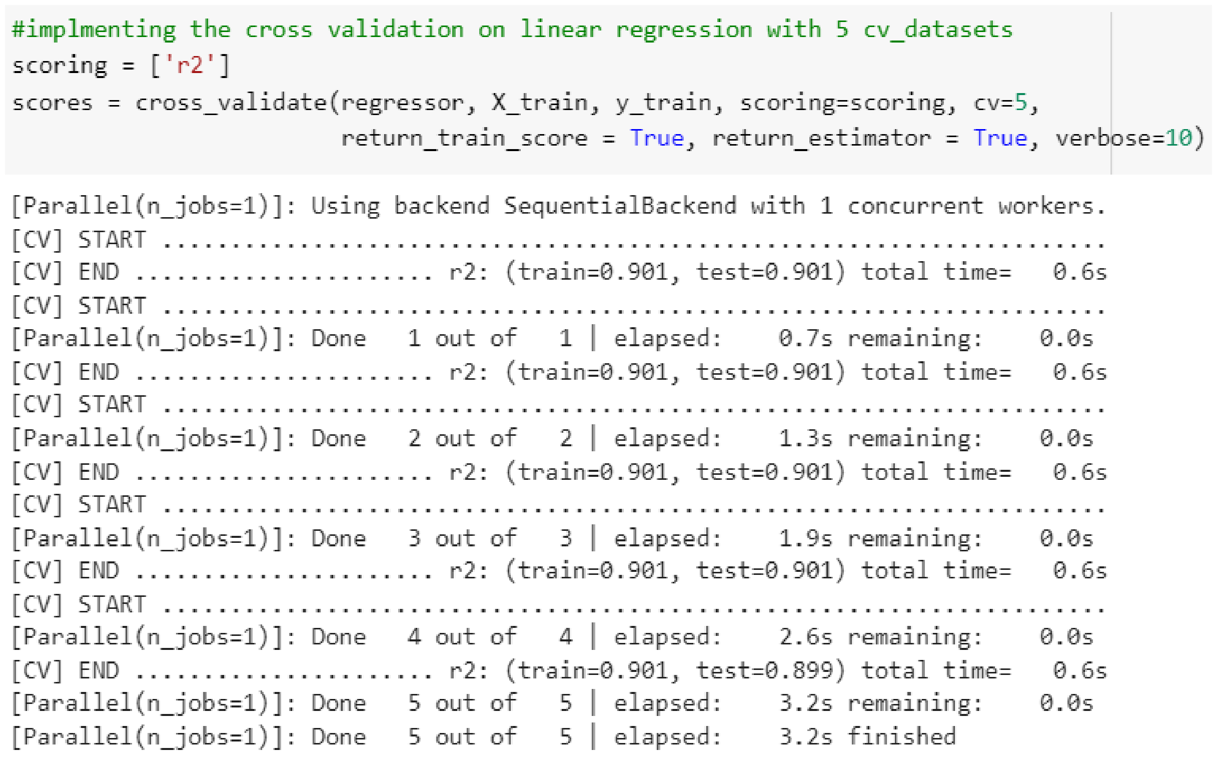
The metric helps us to compare our current model with a constant baseline and tells us how much our model is better. The constant baseline is chosen by taking the mean of the data and drawing a line at the mean. R^2 is a scale-free score that implies it doesn't matter whether the values are too large or too small, the R^2 will always be less than or equal to 1.

R2 score on training and test sets is 0.9007 and 0.8993 respectively.

Cross Validation on Linear Regression

Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

Cross-validation gives a more accurate measure of model quality, which is especially important if you are making a lot of modeling decisions.



```
#implmenting the cross validation on linear regression with 5 cv_datasets
scoring = ['r2']
scores = cross_validate(regressor, X_train, y_train, scoring=scoring, cv=5,
                        return_train_score = True, return_estimator = True, verbose=10)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[CV] START .....................................................................
[CV] END ...................... r2: (train=0.901, test=0.901) total time=   0.6s
[CV] START .....................................................................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.7s remaining:    0.0s
[CV] END ...................... r2: (train=0.901, test=0.901) total time=   0.6s
[CV] START .....................................................................
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    1.3s remaining:    0.0s
[CV] END ...................... r2: (train=0.901, test=0.901) total time=   0.6s
[CV] START .....................................................................
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    1.9s remaining:    0.0s
[CV] END ...................... r2: (train=0.901, test=0.901) total time=   0.6s
[CV] START .....................................................................
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    2.6s remaining:    0.0s
[CV] END ...................... r2: (train=0.901, test=0.899) total time=   0.6s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    3.2s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    3.2s finished
```

Made a cross validation on the model with 5 CV and r2_score as the accuracy score of the model implemented. By using the cross validation we have seen that there is an increase in the accuracy score of the train and test datasets.

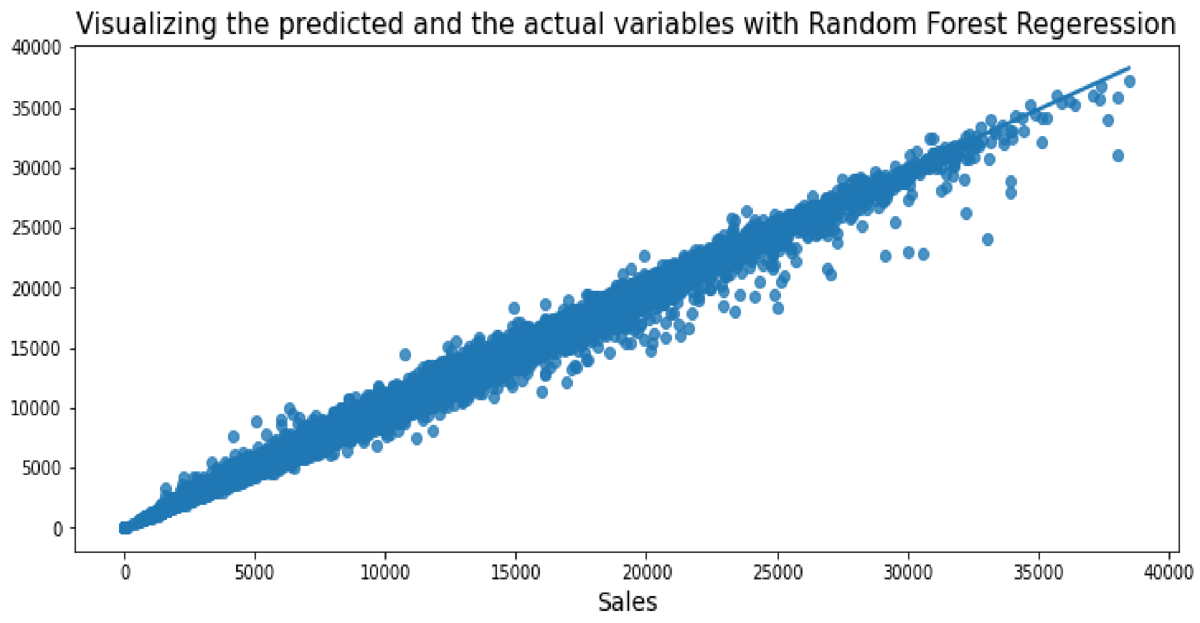
The accuracy score we have for the 5 sets on the training dataset is ([0.90059602, 0.90057281, 0.90061433, 0.90057982, and 0.90099255]). And the test data set is ([0.90059602, 0.90057281, 0.90061433, 0.90057982, and 0.90099255]).

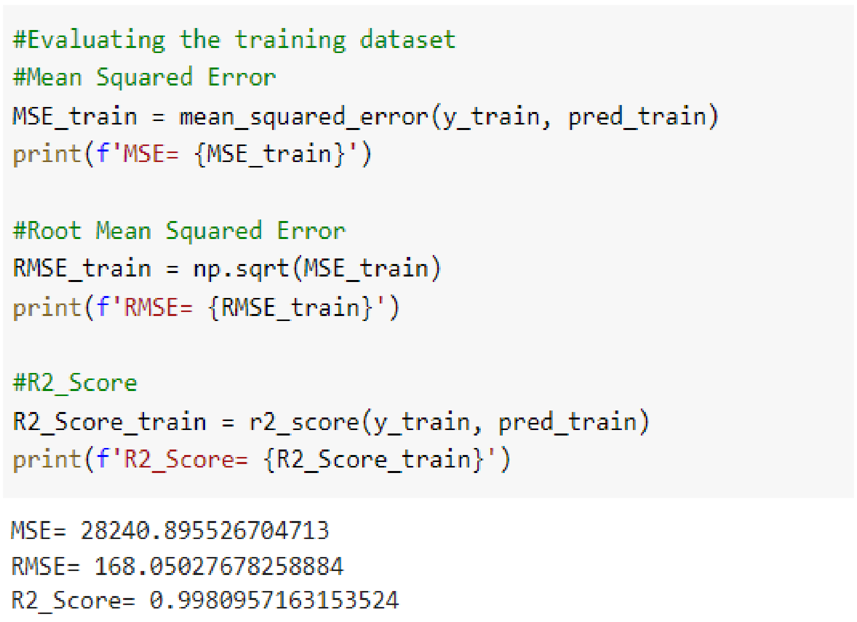
Random Forest Regression

Random Forest is a supervised learning algorithm which uses ensemble learning methods for statistical regression. Ensemble learning method is a technique that combines the predictions from multiple machine learning algorithms to make a more accurate prediction. A random forest operates by constructing several decision trees during training time and outputs the mean of the classes at the prediction of all the trees.

The model was imported and trained with the data available in the training dataset. Defined the predicted variable and checked the score of the model.

Training data



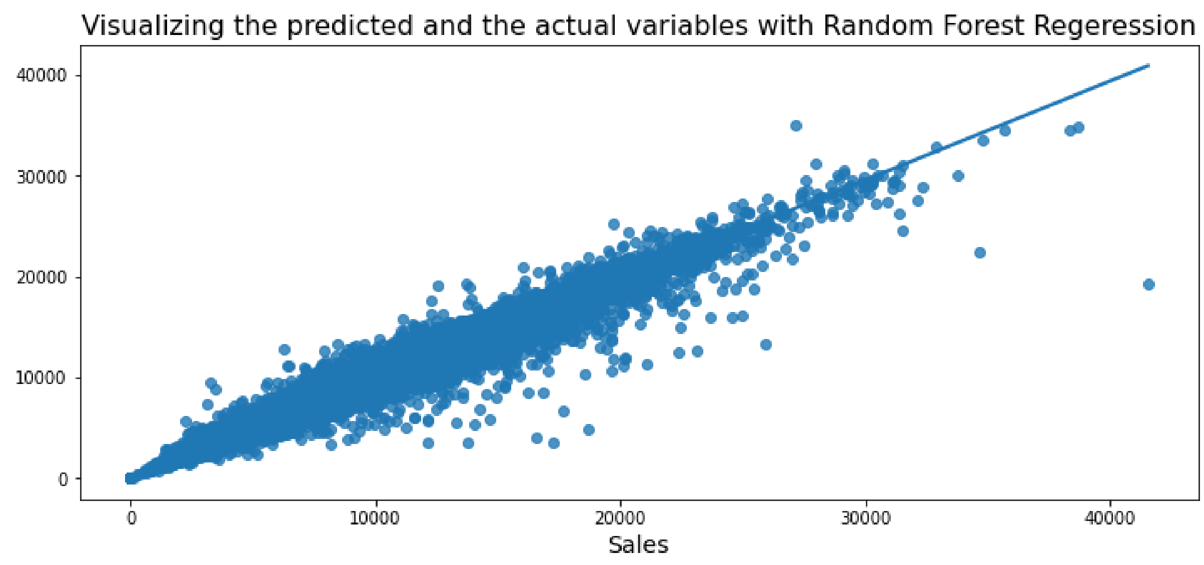


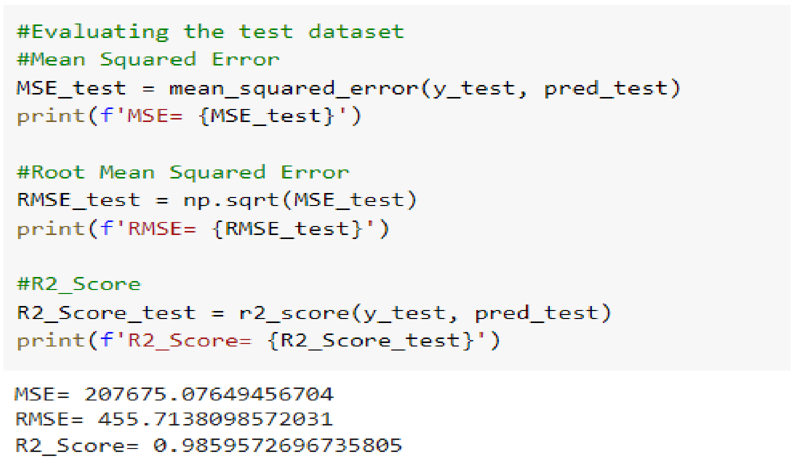
```
#Evaluating the training dataset
#Mean Squared Error
MSE_train = mean_squared_error(y_train, pred_train)
print(f'MSE= {MSE_train}')

#Root Mean Squared Error
RMSE_train = np.sqrt(MSE_train)
print(f'RMSE= {RMSE_train}')

#R2_Score
R2_Score_train = r2_score(y_train, pred_train)
print(f'R2_Score= {R2_Score_train}')
```

```
MSE= 28240.895526704713
RMSE= 168.05027678258884
R2_Score= 0.9980957163153524
```

From the above figure we can see that the best fit line passes through the points and shows how good the model is trained and fits with the data. The model with the random forest regression has an accuracy of 99.81% on the training dataset. The below figure shows the result of the evaluation of the model.

Test dataset





```
#Evaluating the test dataset
#Mean Squared Error
MSE_test = mean_squared_error(y_test, pred_test)
print(f'MSE= {MSE_test}')

#Root Mean Squared Error
RMSE_test = np.sqrt(MSE_test)
print(f'RMSE= {RMSE_test}')

#R2_Score
R2_Score_test = r2_score(y_test, pred_test)
print(f'R2_Score= {R2_Score_test}')
```

```
MSE= 207675.07649456704
RMSE= 455.7138098572031
R2_Score= 0.9859572696735805
```

The above figure shows how good the model has predicted and performed on the test data. It has an accuracy of 99.81% on the training data set and 98.60% on the test dataset.

Conclusion

- Even though the dataset contains more records of the year 2013, 2015 is having more sales among the years.
- In the months of March, July and December we have high sales in those 4 month periods. In all over months we have maximum sales in the month of December.

- 84% of the data belongs to stores that are open. It is assumed that 7 in the day of week as the Sunday and stores are mostly closed during Sundays. And the stores which are opened on Sundays are also having the least number of customer visits during Sundays.
- The working days of the stores are less in 2015. It is assumed that we have less records from the year 2015.
- All schools are closed on public holidays and weekends.
- Sales are highly correlated with the customers.
- We have more customers in the range of 0 to 2000 with the sales range of 0 to 10000. This means their purchasing index lies between 0 and 10000.
- Store number 733 has the highest number of customer visits and store number 543 has the least.
- Even though the above-mentioned stores are having the most customer visits, store number 262 has the highest sales and store number 307 has the least sales.
- Even though store type ‘a’ is having more value counts in the dataset, store type b is having the more number of customer visits and sales.
- We have the competitor stores mostly in the range of 0 to 30000 meters of distance and there is diversion of customers from our stores.
- Linear Regression and Random Forest Regression are used to train the model.
- As per the evaluation it is better to implement the Random Forest Regression rather than going for Linear Regression.
- When it comes to the accuracy the Random Forest Regression is performing well on the test dataset with the accuracy of 98.60%. As it can predict the daily sales of stores for up to six weeks in advance with the same accuracy.