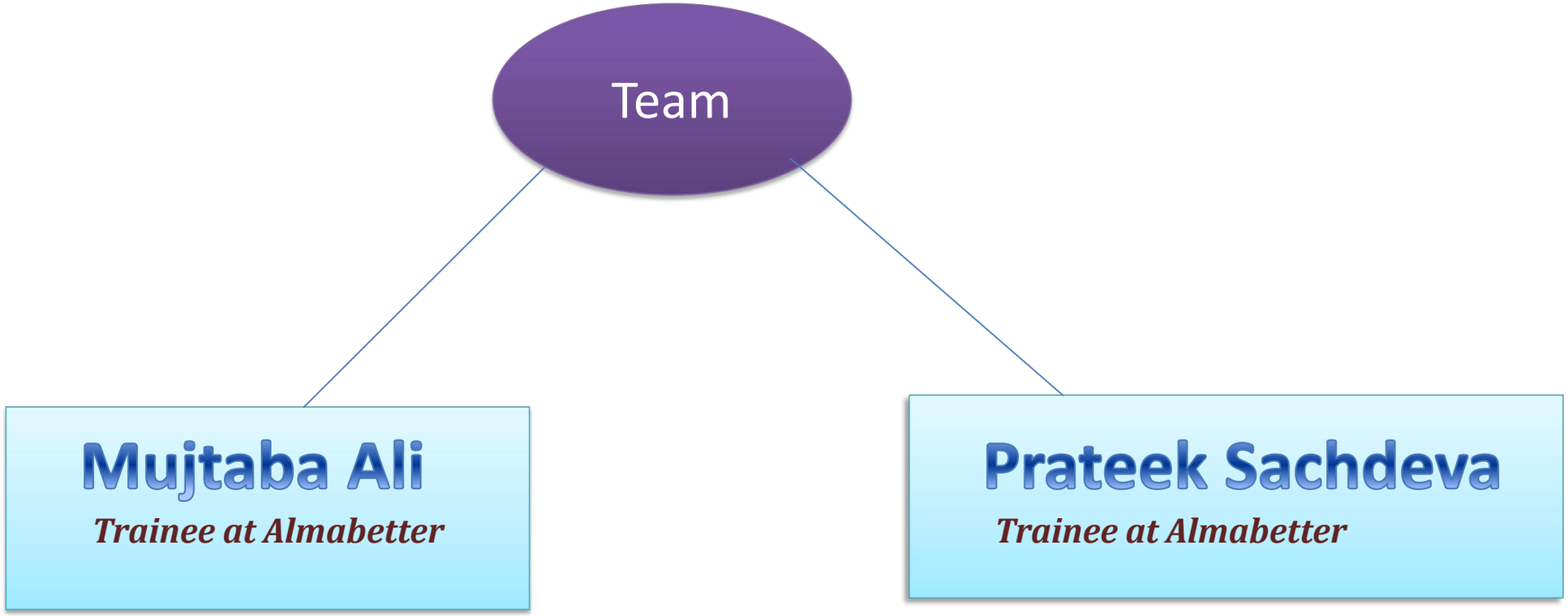
# Capstone Project - Regression

## Rossmann Store Sales Project


Team
Mujtaba Ali
Trainee at Almabetter
Prateek Sachdeva
Trainee at Almabetter

# ❑ Introduction

- 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.
- **Dirk Rossmann** GmbH, commonly referred to as Rossmann, is one of the largest drug store chains in Europe with around **56,200 employees** and more than 4000** stores**. In 2019 Rossmann had more than €10 billion turnover in Germany, Poland, Hungary, the Czech Republic, Turkey, Albania, Kosovo and Spain. The company was founded in **1972** by Dirk Rossmann with its headquarters in Burgwedel near Hanover in Germany. The Rossmann family owns 60% of the company. The Hong Kong-based A.S. Watson Group owns 40%, which was taken over from the Dutch Kruidvat in 2004.
- The product range includes up to **21,700 items** and can vary depending on the size of the shop and the location. In addition to drugstore goods with a focus on skin, hair, body, baby and health, Rossmann also offers promotional items ("World of Ideas"), pet food, a photo service and a wide range of natural foods and wines. There is also a **perfume range with around 200 commercial brands**. Rossmann has **29 private brands with 4600 products (as of 2019)**. In 1997, the first own brands Babydream, Facelle, Sunozon and Winston were introduced. The best-known Rossmann brands are Isana (skin, hair and body care), Alterra (natural cosmetics), domol (cleaning and laundry detergents) alouette (paper tissues etc).

# ❑ Data Summary

- In this session, we will have the overview of the basic understanding of our dataset variables. What does particular features means and how it's distributed, what type of data is it. There are two datasets in Rossmann Sales project, then we merged both of them to make a final dataset. Now Rossmann dataset is having 18 columns in total. We can get this by basic inspection of our dataset.

- We initially had two datasets. After combining both of them, the total records are 1017209.

| | Store | DayOfWeek | Date | Sales | Customers | Open | Promo | StateHoliday | SchoolHoliday | StoreType | Assortment | CompetitionDistance | CompetitionOpenSinceMonth | CompetitionOpenSinceYear |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 5 | 2015-07-31 | 5263 | 555 | 1 | 1 | 0 | 1 | c | a | 1270.0 | 9.0 | 2008.0 |
| **1** | 1 | 4 | 2015-07-30 | 5020 | 546 | 1 | 1 | 0 | 1 | c | a | 1270.0 | 9.0 | 2008.0 |
| **2** | 1 | 3 | 2015-07-29 | 4782 | 523 | 1 | 1 | 0 | 1 | c | a | 1270.0 | 9.0 | 2008.0 |
| **3** | 1 | 2 | 2015-07-28 | 5011 | 560 | 1 | 1 | 0 | 1 | c | a | 1270.0 | 9.0 | 2008.0 |
| **4** | 1 | 1 | 2015-07-27 | 6102 | 612 | 1 | 1 | 0 | 1 | c | a | 1270.0 | 9.0 | 2008.0 |

# ❑ Understand the variables

1. ***ID:***
    - An Id that represents a (Store, Date) duple within the test set.

***2.Store:***
    - A unique Id for each store.

3. ***Customers:***
    - The number of customers on a given day.

4. ***Open***:
    - An indicator for whether the store was open: 0 = closed, 1 = open.

5. ***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.

***6. SchoolHoliday :***

- If the (Store, Date) was affected by the closure of public schools.

***7. StoreType:***

- Differentiates between 4 different store models: a, b, c, d.

***8. Assortment:***

- Describes an assortment level: a = basic, b = extra, c = extended.

***9. CompetitionDistance:***

- Distance in meters to the nearest competitor store.

***10. CompetitionOpenSince[Month/Year]***

- Gives the approximate year and month of the time the nearest competitor was opened.

***11. Promo***

- Indicates whether a store is running a promo on that day.

***12. Promo2***

- Promo2 is a continuing and consecutive promotion for some stores: 0 = store is not participating, 1 = store is participating.

***13. Promo2Since[Year/Week]***

- describes the year and calendar week when the store started participating in Promo2.

***14. 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.

***15. Sales:***

- It is a target variable(dependent feature).

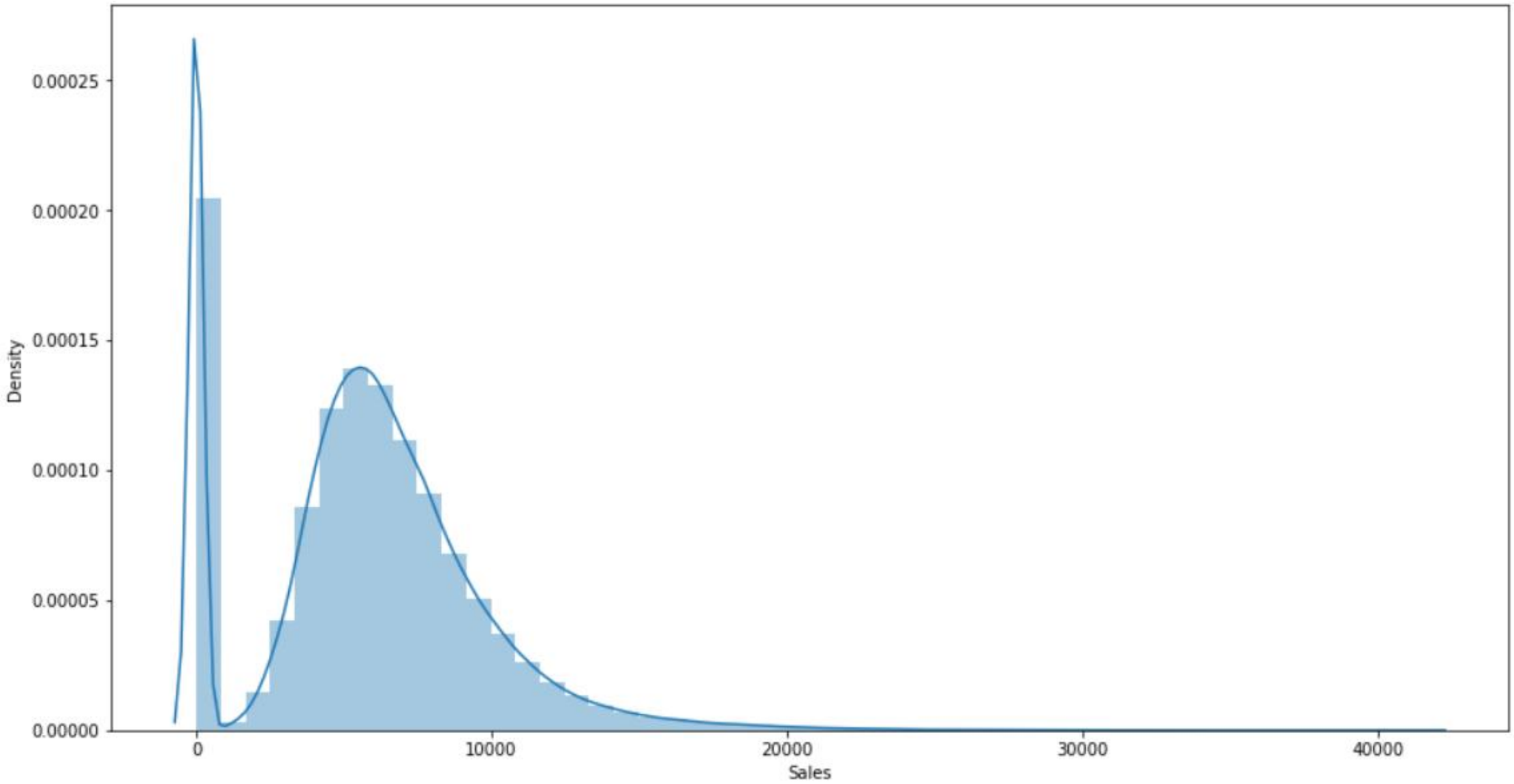
# ❑ Project Architecture:

## 1. EDA:

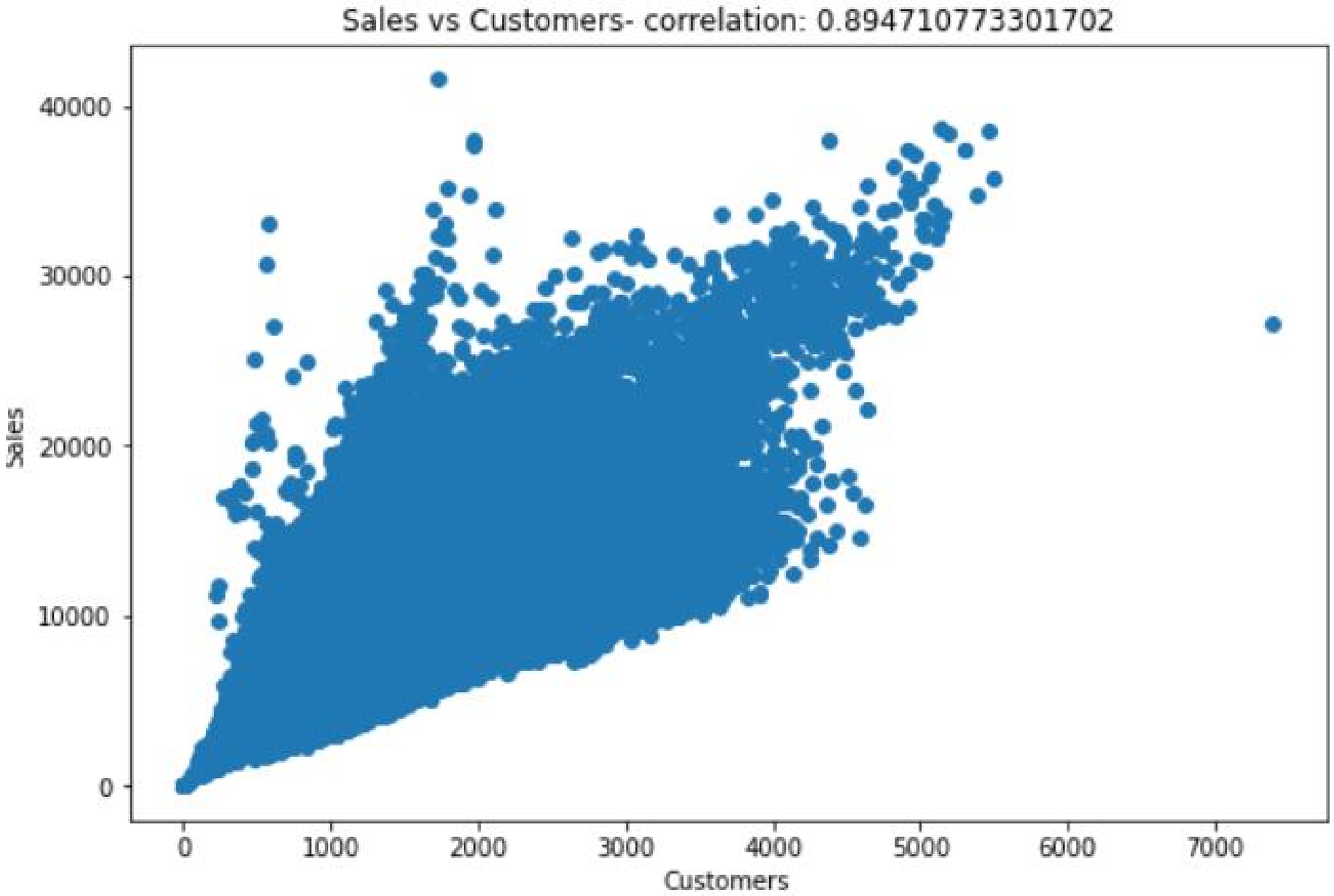
✓ ***Understanding business problem:*** This section, we try to understand the problem.

✓ ***Visualization and Analyzing relationships:*** In this section, we try to understand the distribution between dependent and independent features. We plotted the relationships using various plots like bar plot, histogram, lineplot, and scatterplot etc. We found some insights and relationships like the sales of day 7(Sunday) are the lowest among them (because of the store being closed on Sundays). The sales of store is greatly impacted by promo, promo running increases the sales. Also, the number of customers and sales are directly correlated, the correlation coefficient being 89%.

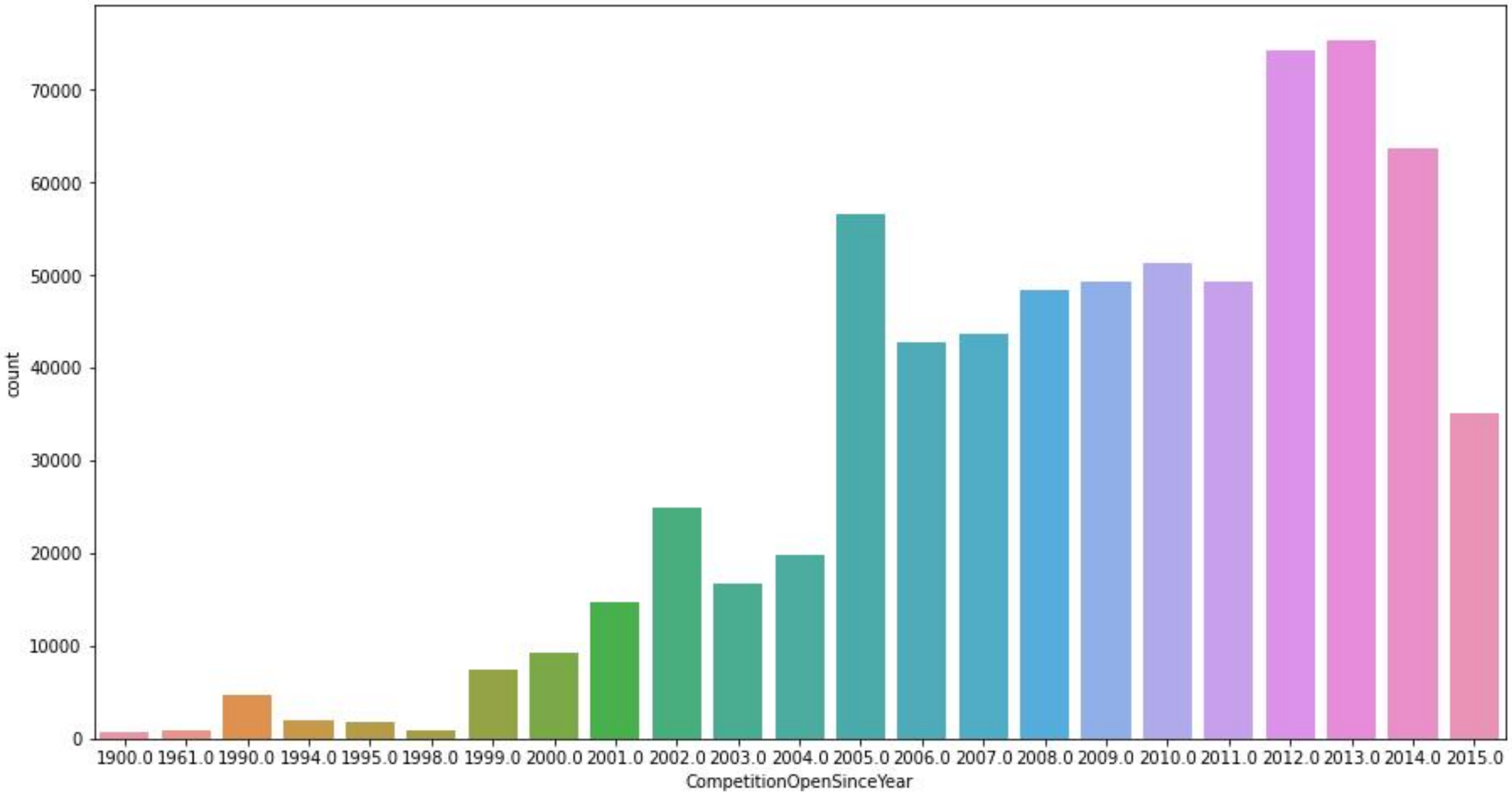
*The distribution plot of Sales(target variable) which is right skewed.*



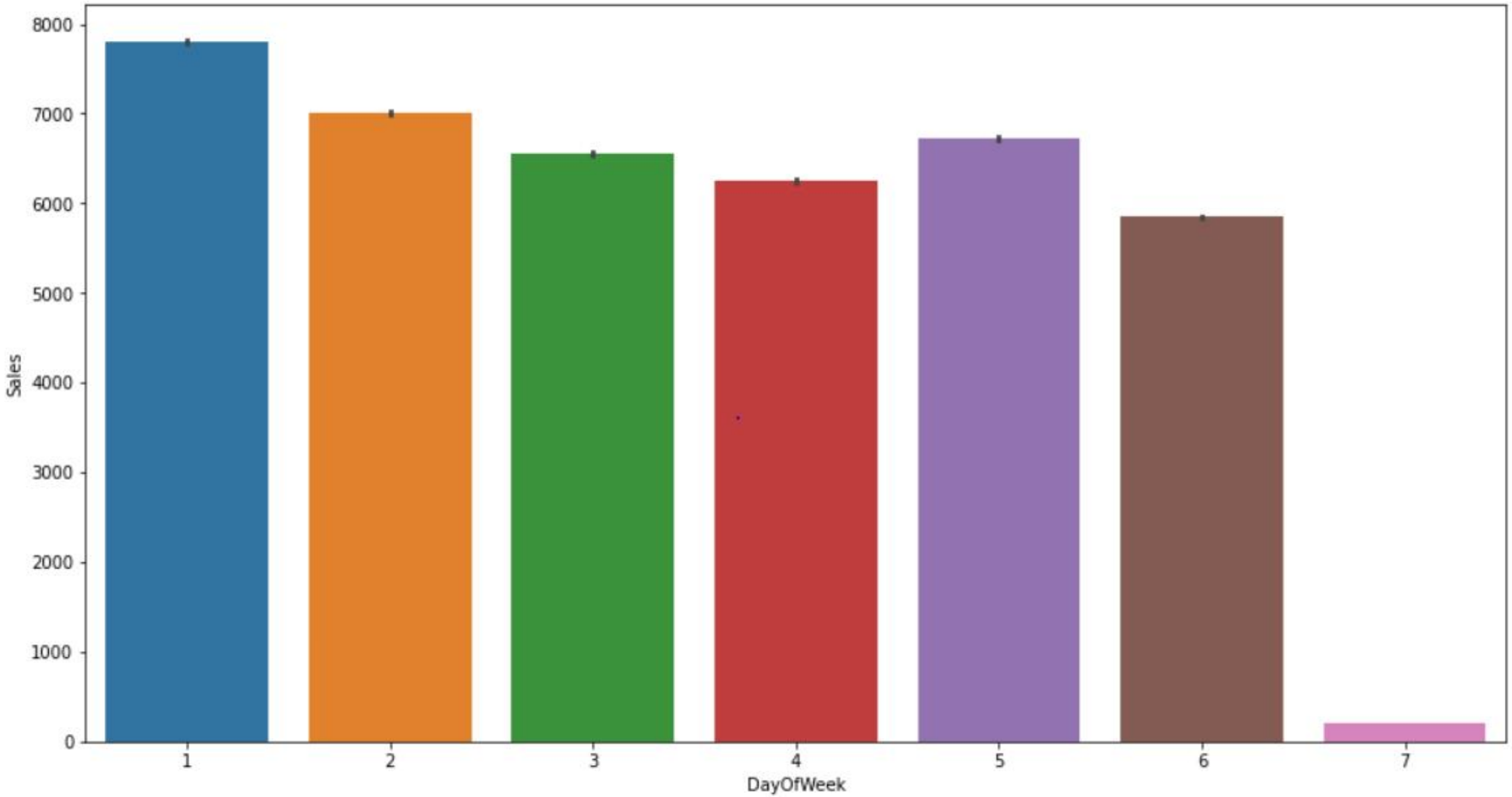
*The plot of Sales Vs  number of  customers having correlation coefficient of 89%*



*The count plot of competition distance since year over the time period of 1900-2015*



✓ ***Forming assumptions and obtaining insights:*** Here we obtained insights through visualization, like Most of the store are closed on Sunday. therefore Sales on Sunday are very- very low, sales is high when Promo is running. On Sunday, Mostly Stores are closed, So the sales is very less.



# 2. Clean-Up:

- ***Missing Values:*** We found the missing values in some columns. We imputed the categorical variables with the mode and the numerical ones with the median of the respective column.

*Finding the missing value percentage in each column*

| | Missing_Values | Percentage | DataType |
|---|---|---|---|
| **PromoInterval** | 508031 | 49.943620 | object |
| **Promo2SinceYear** | 508031 | 49.943620 | float64 |
| **Promo2SinceWeek** | 508031 | 49.943620 | float64 |
| **CompetitionOpenSinceYear** | 323348 | 31.787764 | float64 |
| **CompetitionOpenSinceMonth** | 323348 | 31.787764 | float64 |
| **CompetitionDistance** | 2642 | 0.259730 | float64 |
| **DayOfWeek** | 0 | 0.000000 | int64 |
| **Promo2** | 0 | 0.000000 | int64 |

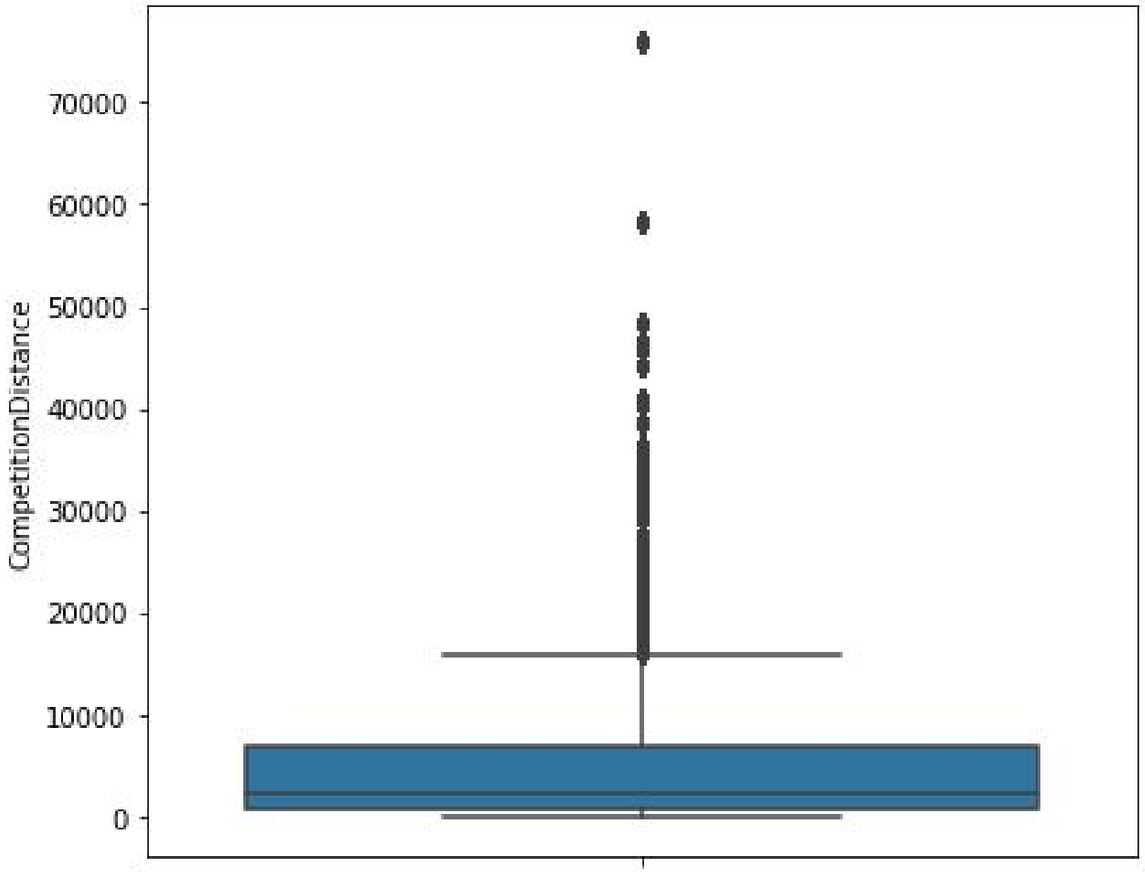
*After imputing the missing values, we can see that there is no null value left in any column.*

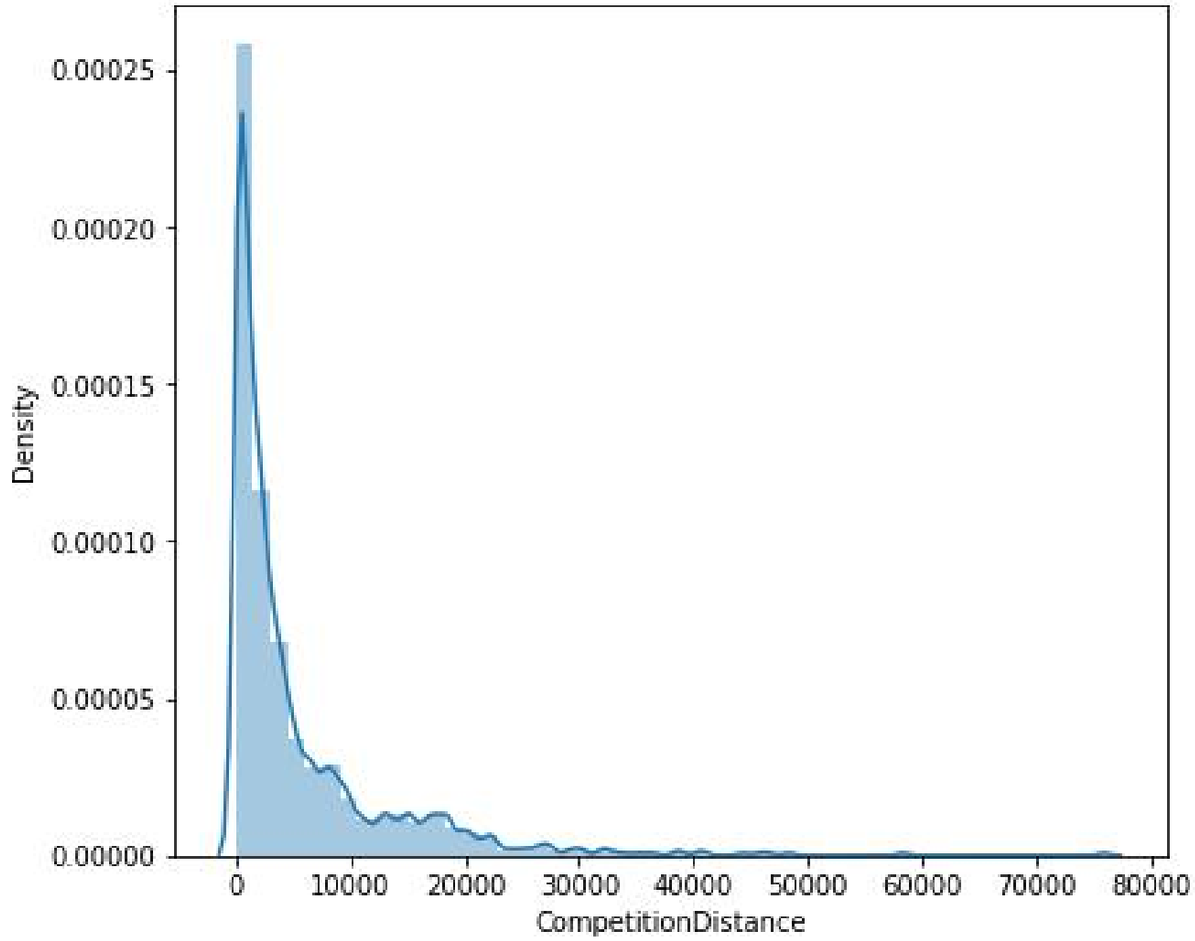
| | Missing_Values | Percentage | DataType |
|---|---|---|---|
| **Store** | 0 | 0.0 | int64 |
| **DayOfWeek** | 0 | 0.0 | int64 |
| **Promo2SinceYear** | 0 | 0.0 | float64 |
| **Promo2SinceWeek** | 0 | 0.0 | float64 |
| **Promo2** | 0 | 0.0 | int64 |
| **CompetitionOpenSinceYear** | 0 | 0.0 | float64 |
| **CompetitionOpenSinceMonth** | 0 | 0.0 | float64 |
| **CompetitionDistance** | 0 | 0.0 | float64 |
| **Assortment** | 0 | 0.0 | object |
| **StoreType** | 0 | 0.0 | object |
| **SchoolHoliday** | 0 | 0.0 | int64 |
| **StateHoliday** | 0 | 0.0 | object |
| **Promo** | 0 | 0.0 | int64 |

✓ ***Outliers:*** After imputing the missing values. We detected the outliers in our dataset using boxplot. Later on we handled those in feature scaling section, where we used Robust scaler.

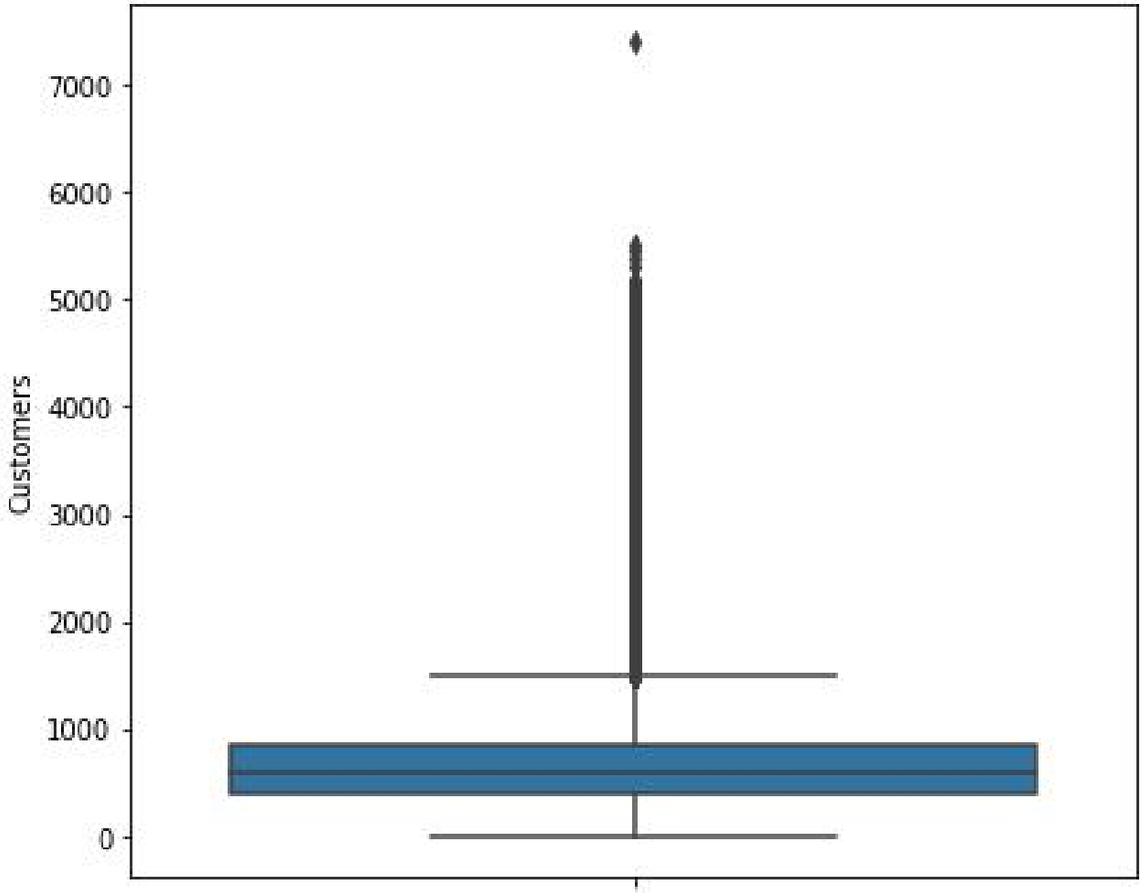
✓ After Imputing the missing values and detecting the outliers we are done with cleaning up part.

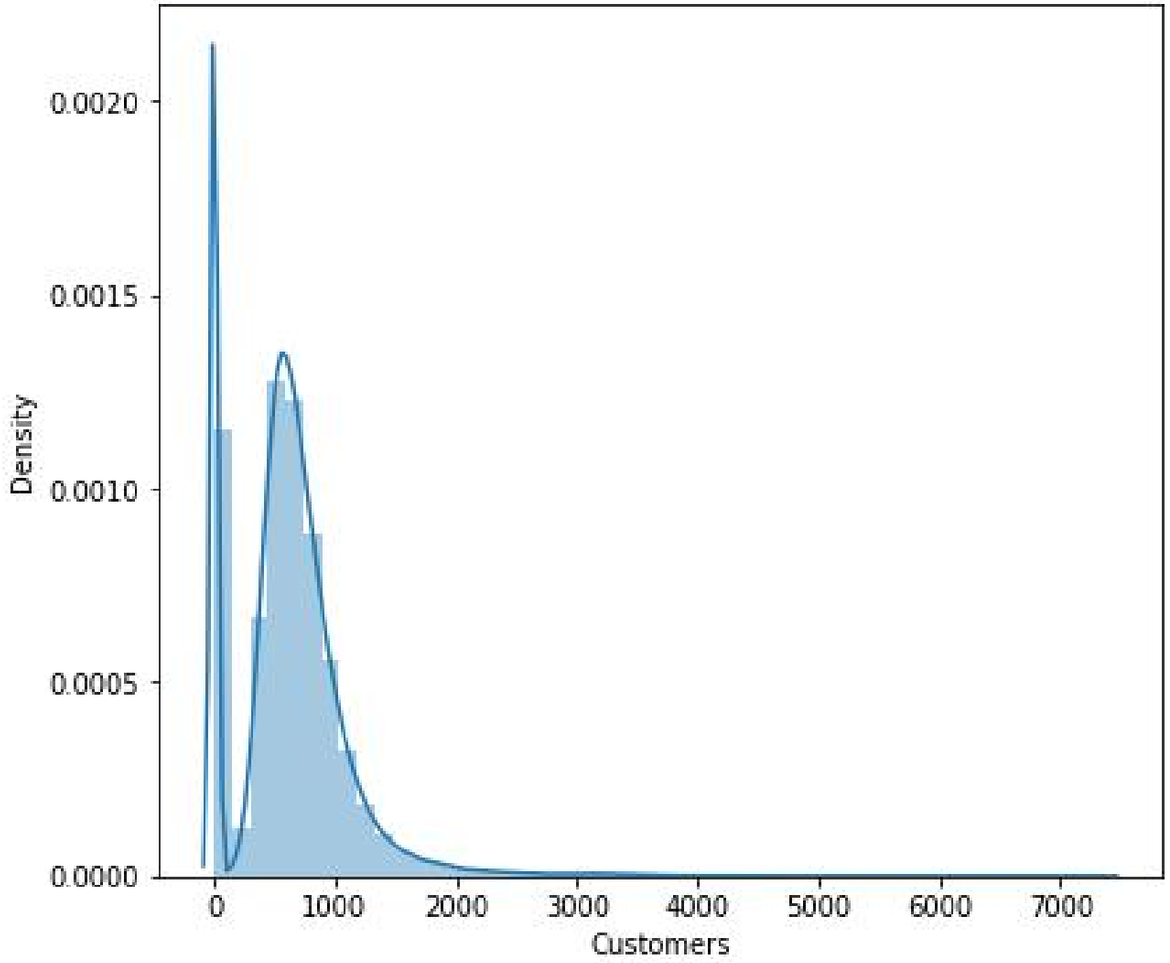
*The box-plot and distribution plot of Competition distance shown.*



*The box-plot and distribution plot of Customers shown.*



# 3. Feature Engineering:

✓ ***Feature Encoding:*** In feature engineering part, we encoded categorical feature. We applied ordinal encoding on ordinal categorical features and one-hot encoding on nominal categorical features.

✓ ***Features Construction:*** In feature construction, we extracted day, month, and year from the existing date column and created new features accordingly.

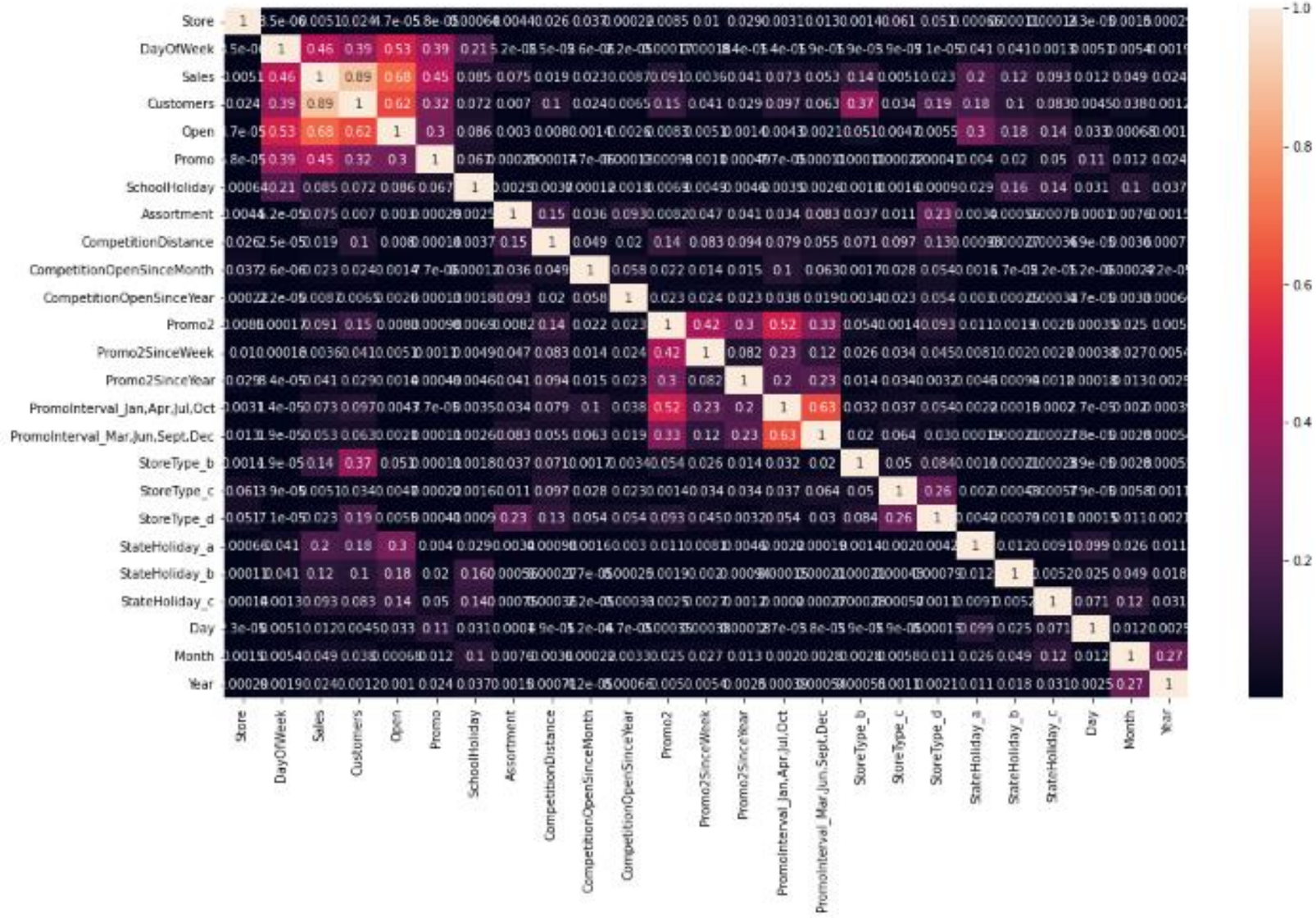
```
# Creating New feature from Existing One.

df['Day'] = df['Date'].dt.day # Extracting Day from Date column using day attribute.
df['Month'] = df['Date'].dt.month # Extracting Month from Date column using month attribute.
df['Year'] = df['Date'].dt.year # Extracting Year from Date column using year attribute.
```

```
df.drop(columns='Date',inplace = True) # Dropping Date coulum beacause we made new features from Date column.
```

+ Code + Text

✓ ***Removing Multicollinearity:*** *We checked the multicollinearity among the independent variables using variance Inflation factor and removed the features having high factor of VIF.*

*The heatmap of variables shown, having correlation values among themselves.*



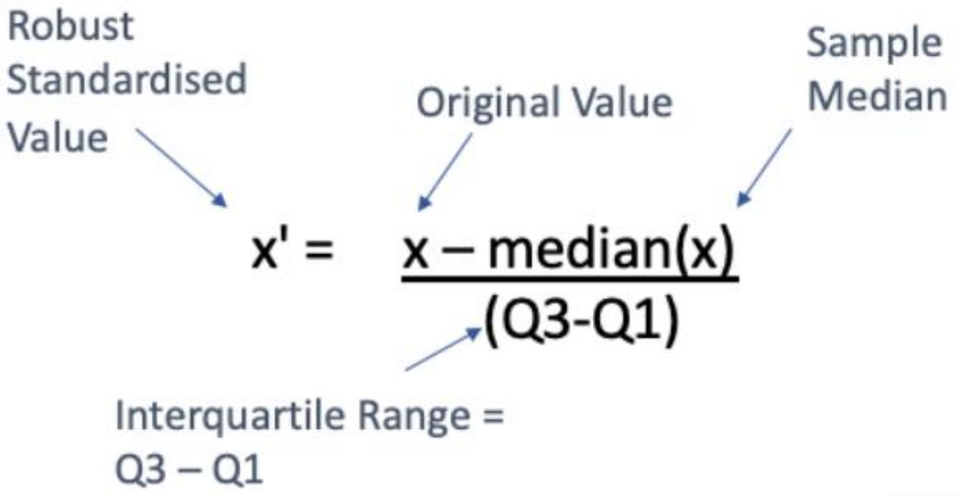
| | variables | VIF |
|---|---|---|
| 0 | DayOfWeek | 8.622225e+00 |
| 1 | Sales | 7.498183e+00 |
| 2 | Open | 1.466119e+01 |
| 3 | Promo | 2.228408e+00 |
| 4 | SchoolHoliday | 1.347532e+00 |
| 5 | Assortment | 5.260831e+00 |
| 6 | CompetitionOpenSinceMonth | 9.028725e+00 |
| 7 | CompetitionOpenSinceYear | 1.416801e+05 |
| 8 | Promo2 | 3.468423e+00 |
| 9 | Promo2SinceWeek | 4.953029e+00 |
| 10 | Promo2SinceYear | 2.216282e+06 |
| 11 | PromoInterval_Jan,Apr,Jul,Oct | 9.619577e+00 |
| 12 | PromoInterval_Mar,Jun,Sept,Dec | 1.863055e+00 |
| 13 | StoreType_b | 1.084716e+00 |
| 14 | StoreType_c | 1.266941e+00 |
| 15 | StoreType_d | 1.683757e+00 |
| 16 | StateHoliday_a | 1.239169e+00 |
| 17 | StateHoliday_b | 1.121933e+00 |
| 18 | StateHoliday_c | 1.080689e+00 |
| 19 | Day | 4.322120e+00 |
| 20 | Month | 4.322035e+00 |
| 21 | Year | 2.295002e+06 |

*This is the VIF value table showing the VIF value of each variable. Typically we drop the variables having high value of VIF. Here the highest value is of Promo2SinceYear, so we will drop it first.*

# 4. Pre-Processing:

- ✓ ***Feature Scaling:*** After removing multicollinearity, we scaled the features using standard-scaler, and robust-scaler(for continuous features). After Standardization, we shift the mean of all the independent variable to 0 and their Standard deviation to 1, thereby making the distribution standard normal. Using robust scaler, we are handling all the outliers present in our data.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$



# 5. Train_Test_Split:

✓ ***Train Test Split:*** After Pre-processing data, we split them into training and testing dataset in the ratio of 80:20 respectively using train_test_split method.

```
[ ] # Applying train_test_plit to split dataframe in train and test.
    # Testing size is 20% of the whole dataframe.

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

```
# Checking the shape.

print("Size of X_train: ",X_train.shape)
print("Size of X_test : ",X_test.shape)
print("Size of y_train : ",y_train.shape)
print("Size of y_test : ",y_test.shape)
```

```
Size of X_train:  (813767, 21)
Size of X_test :  (203442, 21)
Size of y_train :  (813767,)
Size of y_test :  (203442,)
```

# 6. Model Implementation:

1. ***Linear Regression:*** Linear regression analysis is used to predict the value of the target variable based on independent variables(features) by using the best fit line having optimum value of intercept and coefficients(slope/weights).

```
# Finding Intercept of linear model using intercept_ attribute

lr.intercept_
```

```
5536.9062933057785
```

```
# Checking the coefficient of linear model.

lr.coef_
```

```
array([ 125.54826281, 3123.00116679,  -36.50456958,  -98.96517365,
        324.20797805,  547.44529393,   11.93003971,  128.31904845,
        -30.25103432,   99.66579365,   82.50604155,   29.17801222,
        -42.11665015, -654.21475928,  -32.01942977,  427.29015142,
        -79.49570752,  -59.81179785,  -17.34008533,   10.08250472,
         71.28855858])
```

*After building the linear regression, we got the intercept value is 5536.90 and the coefficients are given in graph which we can see in left.*

## ✓ ***Score of linear regression:***

AI

```
# Accuray Score before Validation

print("Accuray Score at training data : ",lr.score(X_train_Scaled, y_train))
print("Accuracy score at testing data : ",lr.score(X_test_Scaled, y_test))
print("R2 Score : ",r2_score(y_test, y_pred_lr))
mse = mean_squared_error(y_test, y_pred_lr)
print("Mean Squared Error : ",mse)
rmse = np.sqrt(mse)
print("Root Mean Squared Error is : ",rmse)
```

```
Accuray Score at training data :  0.8955253095474183
Accuracy score at testing data :  0.8951596497664747
R2 Score :  0.8951596497664747
Mean Squared Error :  1545491.6503600355
Root Mean Squared Error is :  1243.1780445133495
```

In linear regression, we got 89% accuracy. And also we found the MSE(mean square error), and RMSE(root mean square error).

# ❑Regularization:

We applied two techniques for regularization, first one is Ridge and second one is Lasso.

## *i. Lasso:*

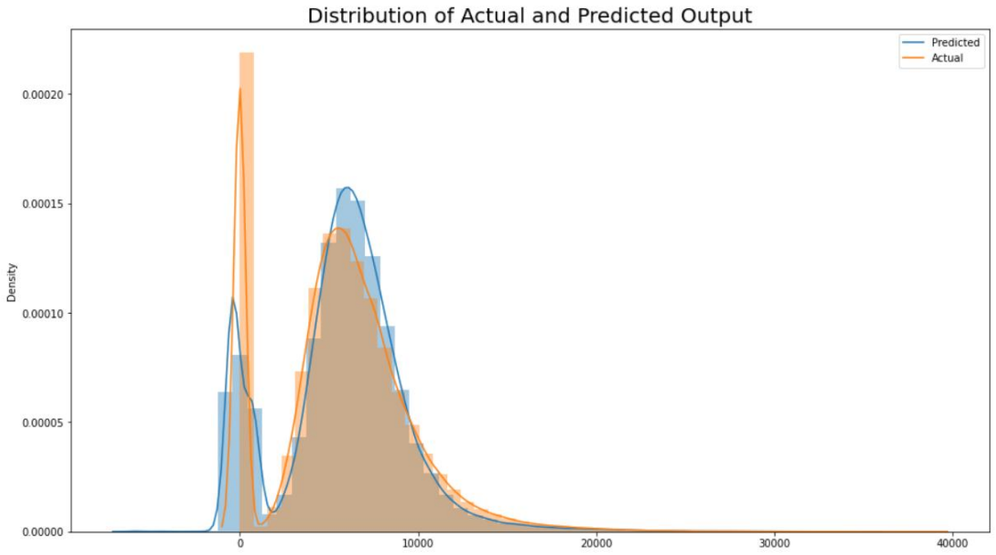
```
# Score of Lasso.
print("Accuray Score at training data : ",lasso.score(X_train_Scaled, y_train))
print("Accuracy score at testing data : ",lasso.score(X_test_Scaled, y_test))
print("R2 Score : ",r2_score(y_test, y_pred_lasso))
print("Adjusted R2 : ",1-(1-r2_score(y_test, y_pred_lasso))*((X_test_Scaled.shape[0]-1)/(X_test_Scaled.shape[0]-X_test_Scaled.shape[1]-1)))
mse = mean_squared_error(y_test, y_pred_lasso)
print("Mean Squared Error : ",mse)
rmse = np.sqrt(mse)
print("Root Mean Squared Error is : ",rmse)
```

```
Accuray Score at training data :  0.8955240041865958
Accuracy score at testing data :  0.895158656541069
R2 Score :  0.895158656541069
Adjusted R2 :  0.8951478332778076
Mean Squared Error :  1545506.291875138
Root Mean Squared Error is :  1243.1839332436443
```

# ii. ***Ridge Regularization:***

```
# accuracy_score of Ridge Regression before validation.

print("Accuray Score at training data : ",rig.score(X_train_Scaled, y_train))
print("Accuracy score at testing data : ",rig.score(X_test_Scaled, y_test))
print("R2 Score : ",r2_score(y_test, y_pred_rig))
print("Adjusted R2 : ",1-(1-r2_score(y_test, y_pred_rig))*((X_test_Scaled.shape[0]-1)/(X_test_Scaled.shape[0]-X_test_Scaled.shape[1]-1)))
mse = mean_squared_error(y_test, y_pred_rig)
print("Mean Squared Error : ",mse)
rmse = np.sqrt(mse)
print("Root Mean Squared Error is : ",rmse)
```

```
Accuray Score at training data :  0.8955253095452929
Accuracy score at testing data :  0.8951596491169851
R2 Score :  0.8951596491169851
Adjusted R2 :  0.8951488259561919
Mean Squared Error :  1545491.6599344078
Root Mean Squared Error is :  1243.1780483641141
```

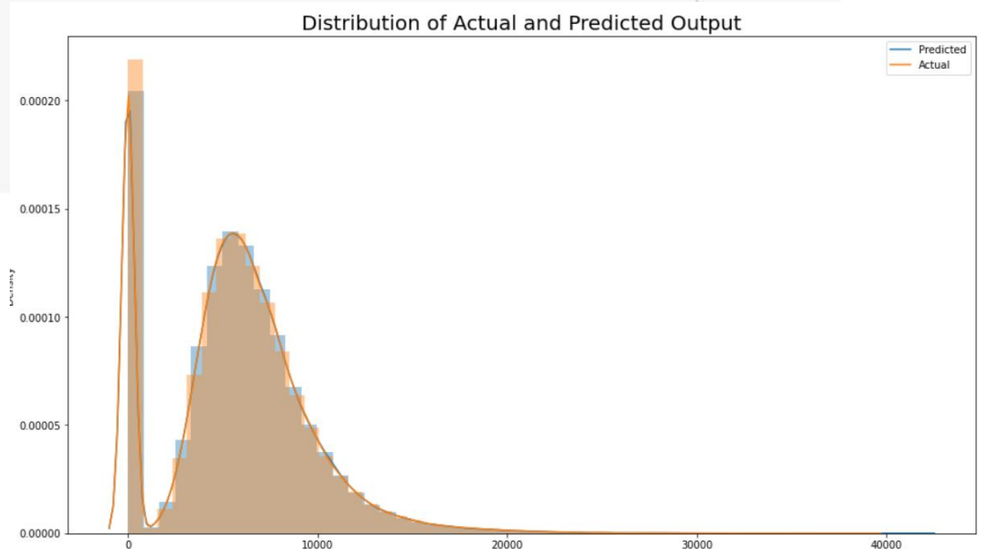



- We can see that in Ridge, the mean squared error is little bit decreased only. But if we talk about r squared, adjusted 2 squared as same.
- As we can see the distribution plots of actual and predicted values.

# 2. ***Decision Tree Regressor:***

```
# finding score of decision tree regressor.

print("Accuray Score at training data : ",dtree.score(X_train_Scaled, y_train))
print("Accuracy score at testing data : ",dtree.score(X_test_Scaled, y_test))
print("R2 Score : ",r2_score(y_test, y_pred_dtree))
mse = mean_squared_error(y_test, y_pred_dtree)
print("Mean Squared Error : ",mse)
rmse = np.sqrt(mse)
print("Root Mean Squared Error is : ",rmse)
```

```
Accuray Score at training data :  1.0
Accuracy score at testing data :  0.9717114405156861
R2 Score :  0.9717114405156861
Mean Squared Error :  417012.4611977861
Root Mean Squared Error is :  645.7650201100909
```




✓By using decision tree regressor, we got 97% accuracy(over testing data).

✓ The distribution plot of predicted Vs actual output values.

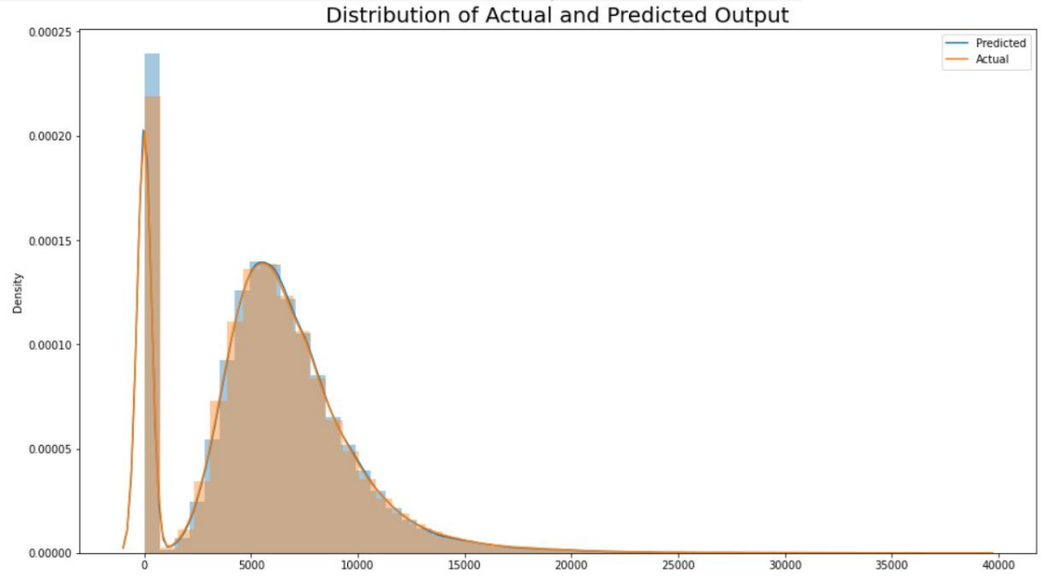
# ❑Emsembles of Decision Tree:

## 1. *Random Forest:*

```
# accuracy_score of random foreset regressor

print("Accuray Score at training data : ",randomfrst.score(X_train_Scaled, y_train))
print("Accuracy score at testing data : ",randomfrst.score(X_test_Scaled, y_test))
print("R2 Score : ",r2_score(y_test, y_pred_randomfrst))
mse = mean_squared_error(y_test, y_pred_randomfrst)
print("Mean Squared Error : ",mse)
rmse = np.sqrt(mse)
print("Root Mean Squared Error is : ",rmse)
```

```
Accuray Score at training data :  0.9980065641051742
Accuracy score at testing data :  0.9858123675035616
R2 Score :  0.9858123675035616
Mean Squared Error :  209145.31010990997
Root Mean Squared Error is :  457.3240755852571
```

✓We got 98% accuracy by using Random forest algorithm. And we found this is the best accuracy for predicting the target variable till now.




✓The distribution plot of predicted and actual output variable.

# 2.XGBoostRegressor:

```
# Accuracy of XGBRegressor

print("Accuray Score at training data : ",Xgboost.score(X_train_Scaled, y_train))
print("Accuracy score at testing data : ",Xgboost.score(X_test_Scaled, y_test))
print("R2 Score : ",r2_score(y_test, y_pred_xgboost))
mse = mean_squared_error(y_test, y_pred_xgboost)
print("Mean Squared Error : ",mse)
rmse = np.sqrt(mse)
print("Root Mean Squared Error is : ",rmse)
```

```
Accuray Score at training data :  0.9392046647415322
Accuracy score at testing data :  0.9381481716590879
R2 Score :  0.9381481716590879
Mean Squared Error :  911781.4281186373
Root Mean Squared Error is :  954.8724669392438
```

When we used XGBoost Regressor, we found the 93.81% accuracy.

# ❑Model Explainability

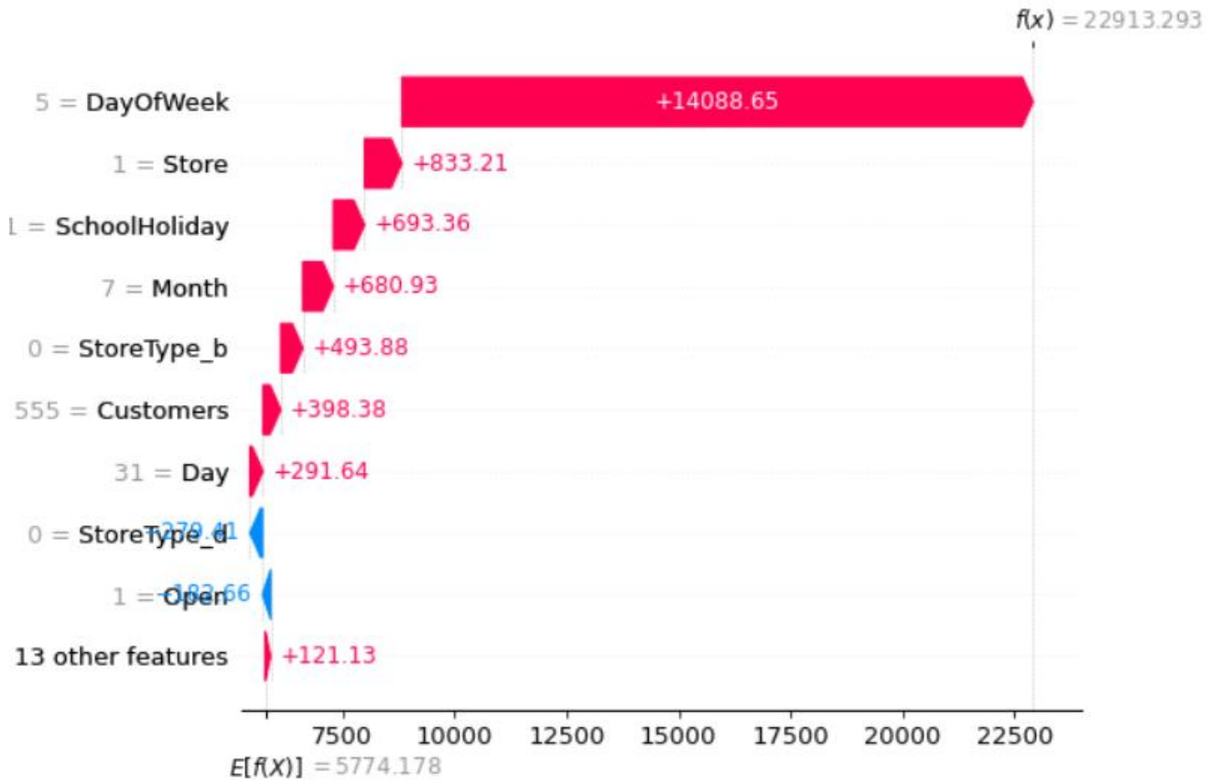
**Model Explainability:**

- Explainability in machine learning means that you can explain what happens in your model from input to output. It makes models transparent and solves the black box problem. Explainability is the degree to which a human can understand the cause of a decision or the degree to which a human can consistently predict ML model results. Explainability and interpretability are often used interchangeably. Although they have the same goal to understand the model.

## *Model Explainability Using SHAP (SHapley Additive exPlanations)*

SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the o utput of any machine learning model. It connects optimal credit allocation with loca l explanations using the classic Shapley values from game theory and their related e xtensions (see papers for details and citations).

# Model Explainability of Xgboost Regressor




✓As we can see that the DayOfWeek is the most important variable having the highest weightage. It means that it is contributing the most in predicting the target.

✓We have noticed that StoreType_d, and Open are one of the least contributer in predicting the target.

# Challenges:

- There were two datasets present initially having different sets of information. It was difficult to work on them separately. So we decided to merge them on the basis on their common attributes.
- There were three columns, in which almost half of the values were missing.
- There were a lot of outliers in numeric columns, which could have badly affected the prediction. However we dealt them with robust scaler(transformation technique).
- The data was highly right skewed, on which linear regresssion does not work well. So we standardized it(made the distribution standard normal) using standard-scaler.
- Due to too heavy dataset(more than a million of records), the computation was very slow each time we applied an machine learning algorithm.

# ❑ Conclusions:

- DayOfWeek was found to be the most important feature(using xgboost regressor), which is contributing the highest in predicting the target variable.
- We found 89% accuracy through linear regression.
- The Regularization techniques of linear regression(Lasso, and Ridge) did not help in improving the accuracy much.
- In decision tree regression, we found 97% accuracy.
- The ensembles of decision tree i.e. Random forest gave us the highest accuracy i.e. 98% in predicting the target variable.
- However through Xgboost Regressor, we got 93% accuracy.
- So we found Random forest to be the best performing algorithm.

AI

# THANK YOU