Capstone Project II
Retail Sales Prediction

Cohort Name-Zurich
Team Name- ml ninjas
Team Members

Aadarsh Soni
Akshat Pathak
Bhaskar Kumar
Santosh Mahanati

Abstract:

Sales forecasting is the process of taking into consideration historic sales, and features affecting those sales volume to predict what might be upcoming sales for the upcoming period. Sales prices are affected by a whole lot of factors like whether promotion is running, holidays, day of the week, promo intervals, and average number of sales per day, average customers per day. Retail sales prediction is important for businesses to be able to plan for the industry changes they would be facing in the future, and be prepared for what might be coming their way. Businesses need these predictions to sit down and think about what they can do in future and what they have been doing wrong till now.

Keywords: exploratory data analysis, data cleaning, visualization, wrangling, correlation

1. 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.

2. Introduction

The data we have is from 2013-2015 mid July, for about 2.5 years. Datasets like these are required by forecasters to build models that would predict prices for some time ahead in the future, so that businesses can plan and improve their sales models on the basis of these predictions.

For these types of problem statements we need to have a good dataset as well, because a different or irrelevant dynamic can actually impact patterns underlying in the data.

For our purposes we have tried understanding and exploring trends in data before doing feature engineering and building models upon them.

3. Understanding the Data

First step involved is understanding the data and getting answers to some basic questions like; What is the data about? How many rows or observations are there in it? How many features are there in it? What are the data types? Are there any missing values? And anything that could be relevant and useful to our investigation.

Let’s just understand the dataset first and the terms involved before proceeding further. Our dataset consists of two csv files, the first consists of historical data with 1017209 rows or observations and 9 columns with no null values.

The second dataset was supplementary information about the stores with 1115 rows and 10 columns and a lot of missing values in a few columns. The data types were of integer, float and object in nature. Let’s define the features involved:

● Id - an Id that represents a (Store, Date) duple within the set

● Store - a unique Id for each store

● Sales - the turnover for any given day (Dependent Variable)

● 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. An assortment strategy in retailing involves the number and type of products that stores display for purchase by consumers.

● 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

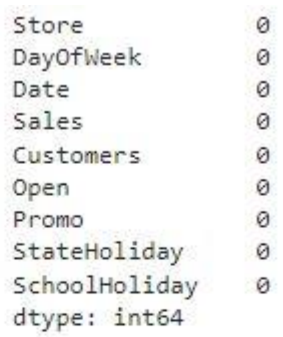
● PromoInterval - describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store.

4. Exploratory Data Analysis

4.1. Data Cleaning and preprocessing.

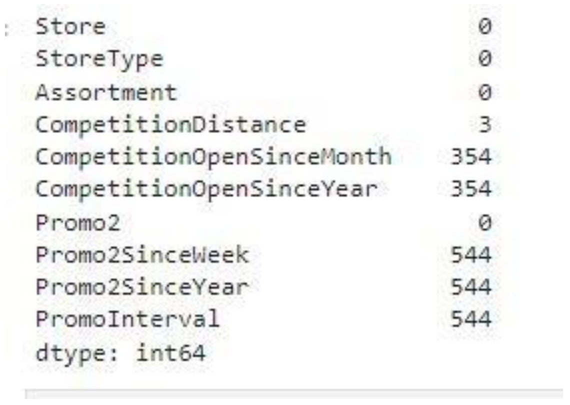
Before doing anything else with data, we need to understand columns with missing or null values. Treatment of null values is important, because they cause a lot of obstacles when feeding data to ML models or even doing EDA.

Null value treatment is considered the first step of data analysis journey, and is usually treated by measures of spread, or if a column contains way too many null values, we need to drop them.
The historical records dataset had no null values.



```
Store            0
DayOfWeek        0
Date             0
Sales            0
Customers        0
Open             0
Promo            0
StateHoliday     0
SchoolHoliday    0
dtype: int64
```

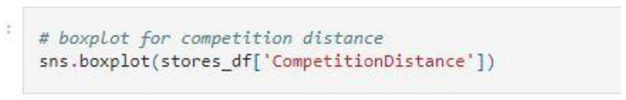
We moved ahead to check the null value in the Store dataset.



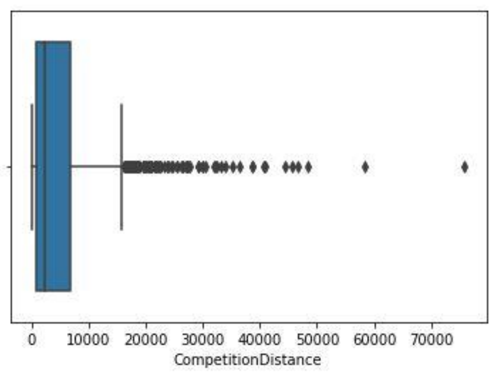
```
Store                          0
StoreType                      0
Assortment                     0
CompetitionDistance            3
CompetitionOpenSinceMonth    354
CompetitionOpenSinceYear     354
Promo2                         0
Promo2SinceWeek              544
Promo2SinceYear              544
PromoInterval                544
dtype: int64
```

Here we can see that CompetitionDistance Column had very few 3 null values.

So for treatment of these three null values, we replaced them with the median value in CompetitionDistance column, since this column has a lot of outliers.



```python
# boxplot for competition distance
sns.boxplot(stores_df['CompetitionDistance'])
```



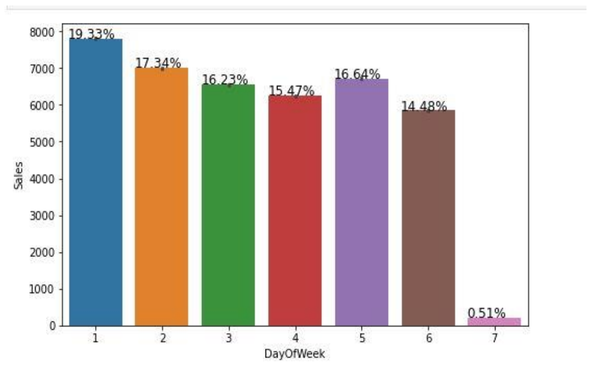
For null values in CompetitionOpenSinceMonth and CompetitionOpenSinceYear, we replaced them with 0, and did the same for all the left columns with null values.

The reason why we did that is, these columns actually impact sales, and whether or not sales is running or there is a promo interval can actually cause sales to fluctuate, so it would be wise not to replace them with anything else.

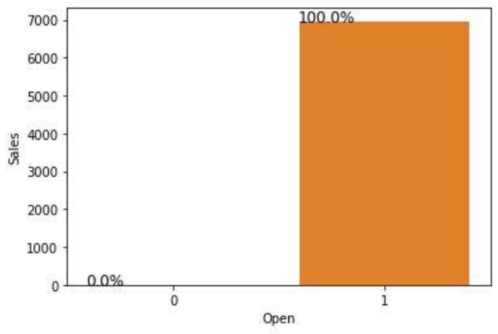
4.2. **Exploratory Data Analysis.**

After treating null values in Stores data, we compared it with historical data columns, and merged both the datasets on stores column for further data exploration.

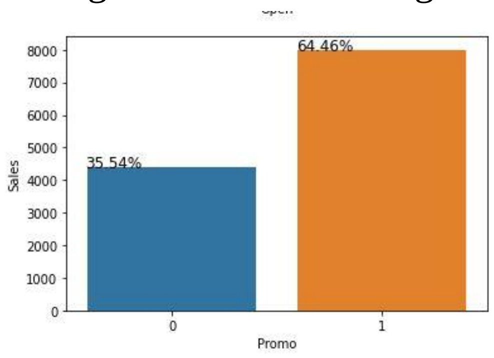
First plot we drew was for sales against days, and went on to plot more like that.



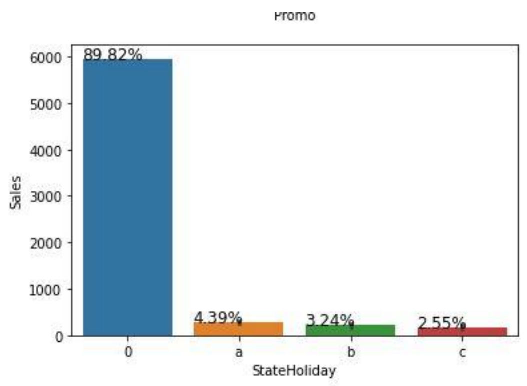
Sales against Open or Closed:



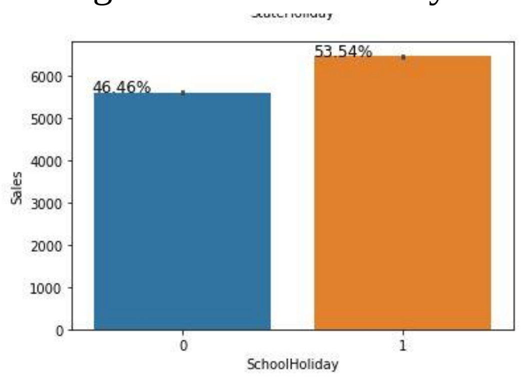
Sales against Promo running or not.



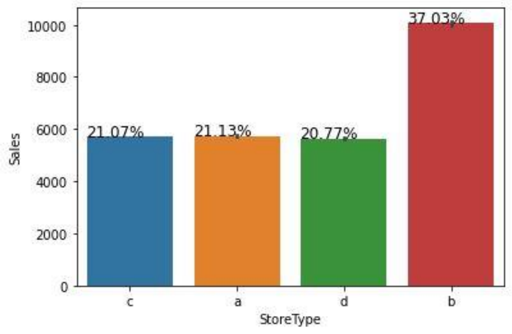
Sales against State Holiday.



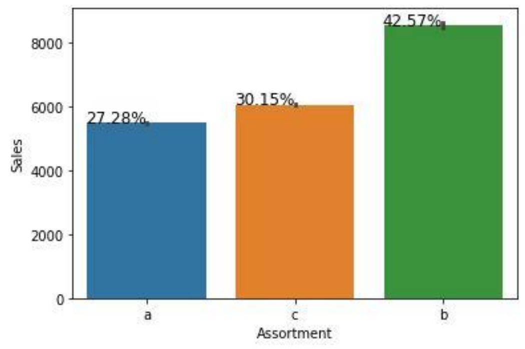
Sales against School Holiday



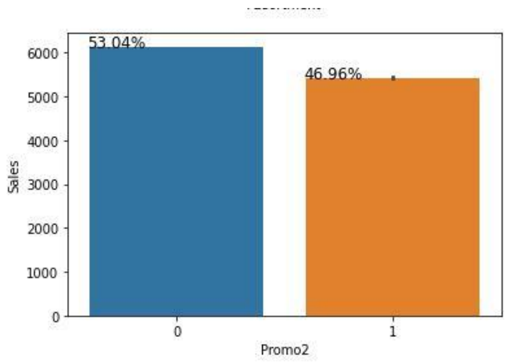
Sales against StoreType



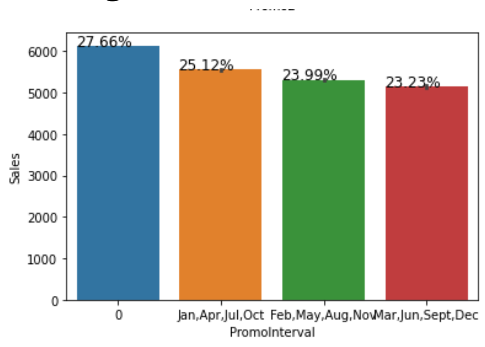
Sales against Assortment type



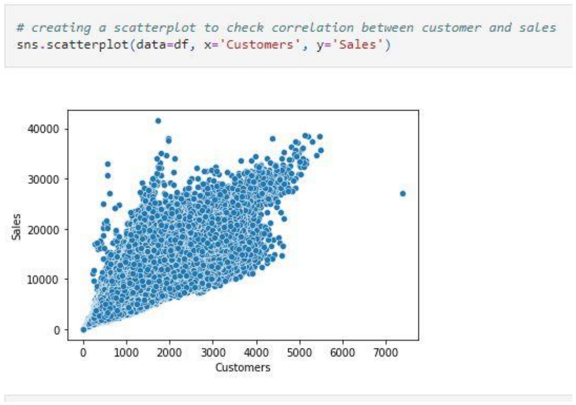
Sales against whether Promo2 is running or not.



Sales against PromoInterval

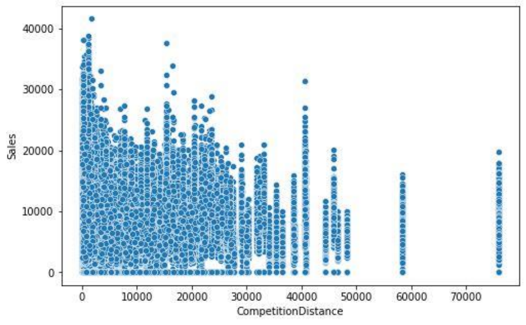


Thereafter we went ahead to create a scatterplot to understand the relationship between customers and Sales.



```
# creating a scatterplot to check correlation between customer and sales
sns.scatterplot(data=df, x='Customers', y='Sales')
```

then , a scatterplot for competition distance and sales.



After that we compared our assumed hypothesis with what insights we obtained from these plots.

Here, is how it went:

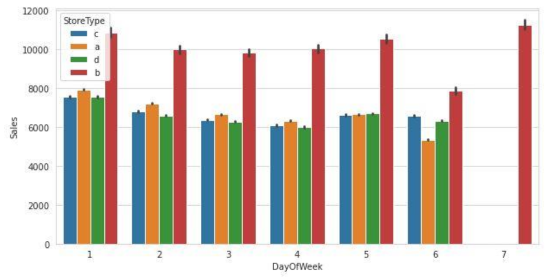
Insights from above plots:

- It verifies our assumption that Mondays are days with most sales, and Sundays are ones with least.

- Stores with assortment level b offers contribute the highest average sales.

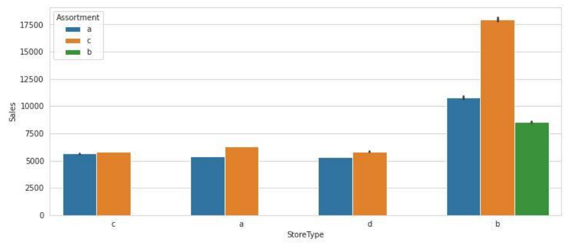
- Stores perform a lot well when Promo is running.
- Seems like less stores are open on State Holidays, hence less average sales are recorded.
- On school holidays, sales are comparatively higher.
- Store type b has the highest average sales.
- Average sales are low when Promo2 is running, we will try finding reasons for that, one assumption is stores were participating less.
- As assumed, customer and sales have positive correlation, except for some rare outliers.
- Opposite to intuition, Sales and Competition distance are actually negatively correlated, and stores having less competition distance actually have more sales.

In our further journey to understand our data, we created a barplot using sns.barplot to show sales of each store type on each day of the week.



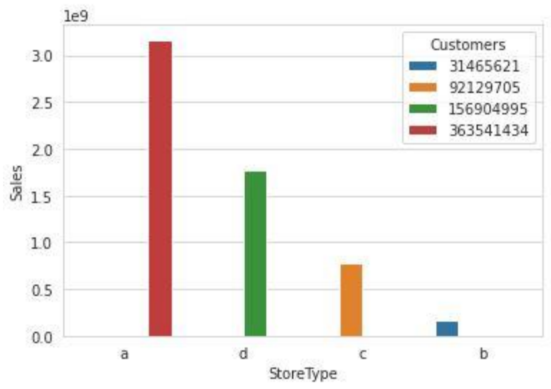
It can be observed that except store type b, almost all other store type were mostly closed on Sundays

Exploring relationship between StoreType and Assortment with reference to Sales:



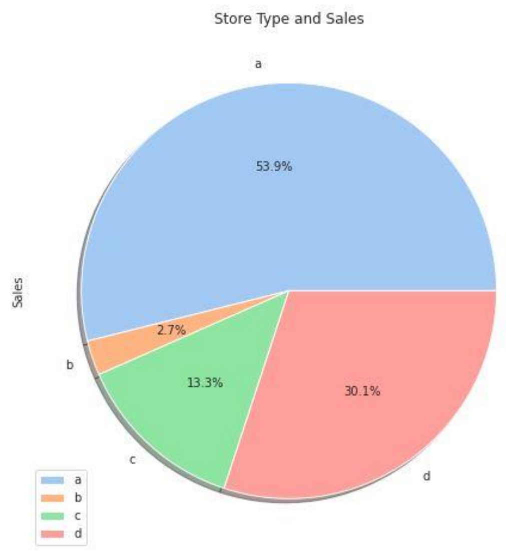
This explains why storetype b had the best average sales of all other store types, it has three assortments available, and also this was the only storetype open even on sundays.

Sales and customer distribution over store types:



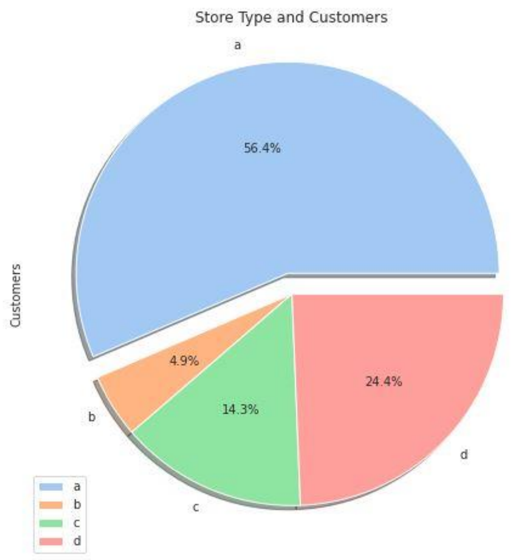
Found that Store a has maximum number of customers and most revenue overall as well.

We went ahead to deep dive into these data with the help of pie charts.



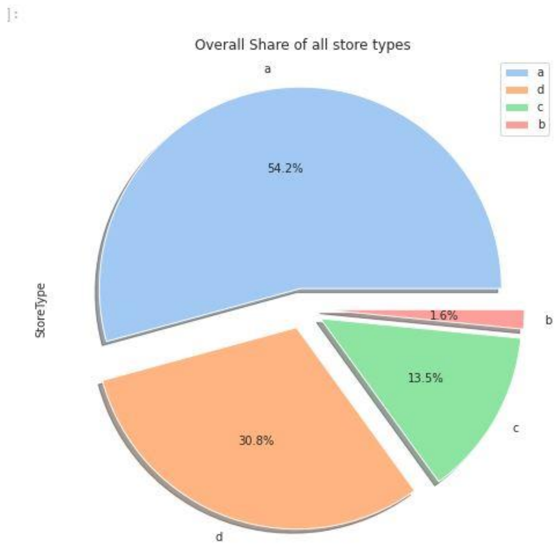
Storetype a accounts for around 54 percent of all Sales happening in data.

Visualizing number of customers in each store type:



Here also, StoreType a has highest share of customers, i.e, around 56 percent, after that comes store type d, then c and b respectively

Visualizing overall share of stores in entire data:



We went ahead to derive conclusions from these pie charts:

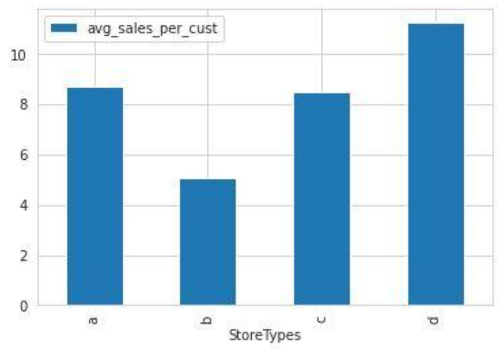
Some conclusions from pie charts:

- Even after having only 1.6% of share of total number of storetype StoreType b has around 4.9 percent customer share and about 3 percent share in overall sales, that might be one reason why average sales for StoreType b was higher than others, also StoreType b is only storetype with all three assortments.
- StoreType a has the highest overall share of around 54%, and it contributes about 54% to overall sales with 56% of customer share. That is a fair reason to conclude why it has the highest revenue numbers.

- Store type c is comparatively linear comprising around 14% of all store counts with around the same percentage share of customers and slightly less share to overall sales around 13%.

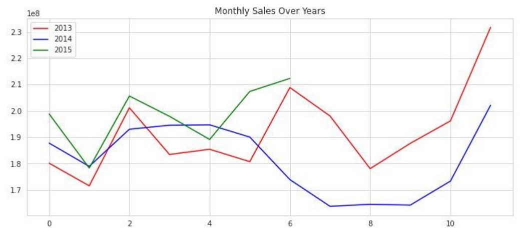
- StoreType d has second highest share in overall number of stores with around 31% share as well as it has 24% share of customers which contributes to 30% of all sales. It can be concluded that per customer sales for store type d must be highest.

Barplot showing average sales per customer for all 4 storetypes:



We can see Average sales per customer for StoreType d is 11.25, maximum.

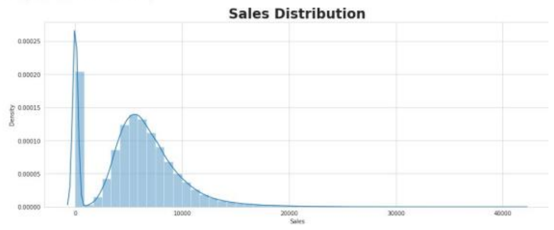
After exploring these much into data, we went ahead to create a line chart showing monthly sales of all three years:



There is a significant drop in sales for around two months in 2014 data, it is due to shops closed for refurbishment for two months in 2014.

Then, a distplot showing sales distribution was created.

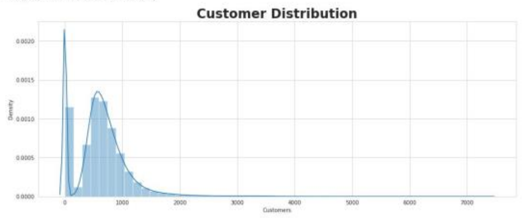
```
Out[ ]: Text(0.5, 1.0, 'Sales Distribution')
```



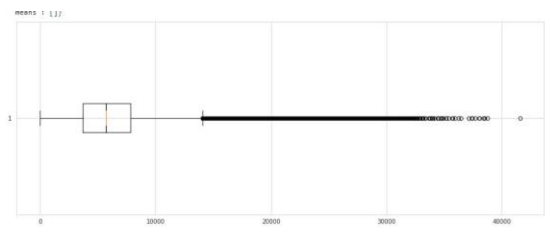
The drop in sales is because of those stores which were closed for refurbishment, this drop was seen in monthly sales over years in 2014 as well.

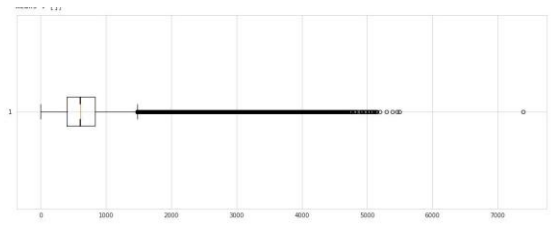
Created a distplot to visualize customer distribution;

```
Out[ ]: Text(0.5, 1.0, 'Customer Distribution')
```



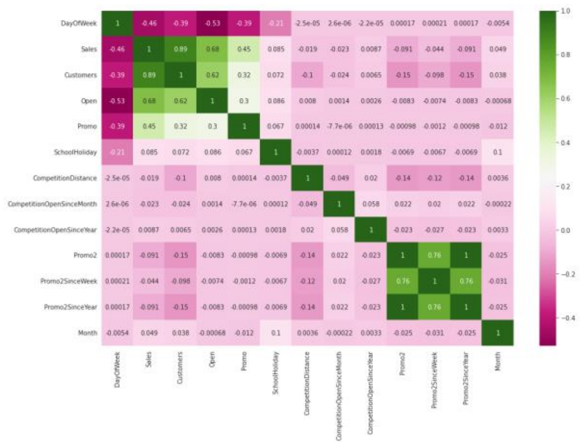
After seeing these displots it was obvious that there are some outliers in both the distplots, we confirmed that with help of boxplots:





After understanding that sales and customers have a lot of outliers, we moved ahead to create a correlation heatmap with the help of sns.heatmap feature of seaborn library.

Correlation: Correlation is a statistical term used to measure the degree in which two variables move in relation to each other. A perfect positive correlation means that the correlation coefficient is exactly 1. This implies that as one variable moves, either up or down, the other moves in the same direction. A perfect negative correlation means that two variables move in opposite directions, while a zero correlation implies no linear relationship at all.



Key takeaways from heatmap:

- DayOfWeek and sales are negatively correlated, that makes sense, since overall average sales decreases as week moves from Monday to Sunday.
- Customers and Sales have high correlation.
- Open and Sales have good correlation.
- Promo and sales have good correlation as well.
- Competition Distance and Sales has negative correlation.
- Multicollinearity can be seen in data, with Columns Promo2, Promo2SinceWeek, Promo2SinceYear being highly collinear to each other.

5. **Data Manipulation and Feature Engineering:**

Data manipulation involves manipulating and changing our dataset before feeding it to various regression machine learning models. This involves keeping important features, outlier treatment, feature scaling and creating dummy variables if necessary.

Feature Engineering:

● Some stores were closed due to refurbishment and some on account of week off or holidays. Those stores on those dates generated zero sales and hence removing the rows was important to avoid confusion by the algorithms and then removing the feature altogether because it wasn't providing any value in prediction of the sales.

● There were features that like Competition Open since Month and Year. It was combined to count the total months since the nearest competition was opened.

● Promo2SinceWeek, Promo2SinceYear indicated promotion 2 opened between week and year. These features were combined to count the total months since promotion 2 is run.

● PromoInterval indicated the months for promotion 2 renewal. Hence, the sale month was compared against the interval and a new feature was created to determine whether the promo2 was renewed in that month.

Outlier Detection: In statistics, an outlier is a data point that differs significantly from other observations. Outliers can occur by chance in any distribution, but they often indicate either measurement error or that the population has a heavy-tailed distribution. Z-score is a statistical measure that tells you how far a data point is from the rest of the dataset. In a more technical term, Z-score tells how many standard deviations away a given observation is from the mean. z = (x-mean)/standard deviation More than 3 standard deviations was considered as an outlier.

Exploring the outliers dataframe, some important insights were generated:

● The data points with sales value higher than 28000 are very low and hence they can be considered as outliers.

● The outliers had day of the week as 7 i.e. Sunday and the store type for those observations were 'b'.

● Other outliers had promotion running on that day.

● It can be well established that the outliers are showing this behavior for the stores with promotion = 1 and store type B. It would not be wise to treat them because the reasons behind this behavior seems fair.

● Being open 24*7 along with all kinds of assortments available is probably the reason why it had higher average sales than any other store type.

● If the outliers are a valid occurrence it would be wise not to treat them by deleting or manipulating them especially when we have established the ups and downs of the target variable in relation to the other features. It is well established that there is seasonality involved and no linear relationship is possible to fit. For these kinds of dataset tree based machine learning algorithms are used which are robust to outlier effect.

Feature Scaling: Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is done to prevent biased nature of machine learning algorithms towards features with greater values and scale. The two techniques are: Normalization: is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling. [0,1] Standardization: is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation. [-1,1] Normalization of the continuous variables was done further.

One hot encoding: For categorical variables where no such ordinal relationship exists, the integer encoding is not enough. We have categorical data integers encoded with us, but assuming a natural order and allowing this data to the model may result in poor performance. Many of the features such as DayofWeek, StoreType and Assortments were categorical in nature and had to be one hot encoded to continue.

6. Modeling:

Train - Test - Split:

After setting Store and Date column to index, we divided dataset into two parts:

Train DF: It consisted of all data for the dates from 1st January 2013 to 15th June, 2015.

Test DF: It consisted of all data for the dates from 15th June, 2015 to 31st July, 2015.

We saved separated csvs for for test and train data to my drive.

Factors affecting in choosing the model: Determining which algorithm to use depends on many factors like the problem statement and the kind of output you want, type and size of the data, the available computational time, number of features, and observations in the data, to name a few.

The dataset used in this analysis has:

● A multivariate time series relation with sales and hence a linear relationship cannot be assumed in this analysis. This kind of dataset has patterns such as peak days, festive seasons etc which would most likely be considered as outliers in simple linear regression.

● Having X columns with 30% continuous and 70% categorical features. Businesses prefer the model to be interpretable in nature and decision based algorithms work better with categorical data

Applying Models:

1. **Baseline Model 1 - Decision Tree:**

A baseline is a simple model that provides reasonable results on a task and does not require much expertise and time to build. It is well established that there is seasonality involved and no linear relationship is possible to fit. For these kinds of datasets tree based machine learning algorithms are used which are robust to outlier effects which can handle non-linear data sets effectively.

Decision Tree is a Supervised learning technique that can be used for both Classification and Regression problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node.

Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The results show that a simple decision tree is performing pretty well on the validation set but it has completely overfitted the train set with a test R^2 of 0.91. It's better to have a much more generalized model for future data points. Businesses prefer the model to be interpretable in nature in order to understand the patterns and strategize accordingly unlike any scientific facility where the results matter much more than interpretability. If interpretability is important then sticking with tree based algorithms when most of the features are categorical; is beneficial and using tuned

Hyperparameters to grow the tree deep enough without overfitting.

2. **Model 2: Random Forest:**

Random forests are an ensemble learning method for classification and regression that operates by constructing a multitude of decision trees at training time. For regression tasks, the output of the random forest is the average of the results given by most trees. In simple terms, random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random Forest Regressor results were much better than our baseline model with a test R^2 of 0.950859

Random Forest Hyperparameters:

● max_depth- The max_depth of a tree in Random Forest is defined as the longest path between the root node and the leaf node

● min_sample_split- a parameter that tells the decision tree in a random forest the minimum required number of observations in any given node in order to split it. The default value of the minimum_sample_split is assigned to 2.

This means that if any terminal node has more than two observations and is not a pure node, we can split it further into subnodes.

● max_leaf_nodes- This hyperparameter sets a condition on the splitting of the nodes in the tree and hence restricts the growth of the tree. If after splitting we have more terminal nodes than the specified number of terminal nodes, it will stop the splitting and the tree will not grow further.

● min_samples_leaf- This Random Forest hyperparameter specifies the minimum number of samples that should be present in the leaf node after splitting a node.

● n_estimators- the number of trees

● max_sample (bootstrap sample)-The max_samples hyperparameter determines what fraction of the original dataset is given to any individual tree.

● max_features- This resembles the number of maximum features provided to each tree in a random forest.

Randomized searchcv searches on hyper parameters to fit and score various models and get the best estimator.

In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n_iter

3. **Model 3: LGBM:**

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel, distributed, and GPU learning.
- Capable of handling large-scale data.

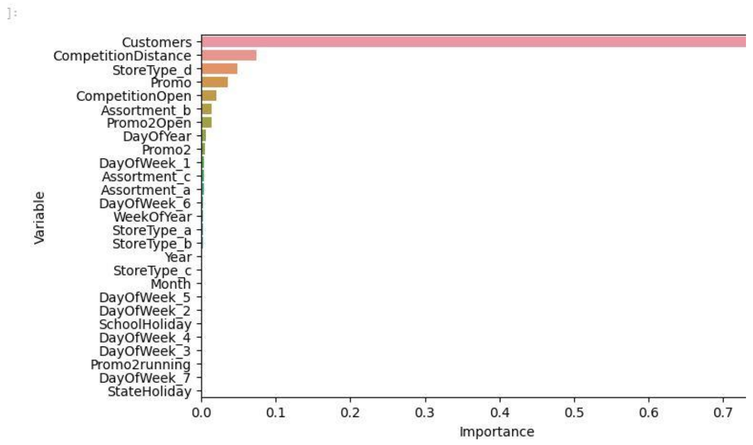
For LGBM regressor model with default parameters, r2 score on testing data was 89%, a fairly underfit as compared to Random Forest regressor.

4. **Model 4: Random Forest Hyperparameter Tuned Model :**

Following are best parameters for Random Forest Model:
max_depth = **None**, max_features = 'auto', min_samples_leaf=1, n_estimators=150

The maximum R^2 was seen in the tuned Random Forest model with the value 0.9510325693402717 which was only 0.018 % improved from a simple random forest model. This indicates that all the trends and patterns that could be captured by these models without overfitting were done and the maximum level of performance achievable by the model was achieved.

Random Forest Hypertuned Model Feature Importances:



The most important features in predicting the Sales were Customers, CompetitionDistance, StoreType D and Promo.

7. Model Performance and Evaluation:

Random Forest vs Baseline Model Model Performance - Improvement of 4.115 % was seen in Random Forest against Decision Tree.

Random Forest Tuned vs Baseline and Random Forest Models Model Performance - Improvement of 4.134 % was seen in Random Forest Tuned against Decision Tree. - Improvement of 0.018 % was seen in Random Forest Tuned against Simple Random Forest.

Observation:

1. We observed maximum r2 for Random forest Tuned, which was only 0.018% more than the simple random forest model.

2. LGBM with its default parameters gave least r2.

8. Conclusion:

We had sales data for around 2.5 years and we had to predict sales prices for the next six weeks. After cleaning data, we took all the data points with some sales to split data into Test and Train data.

Thereafter, we performed StandardScaling techniques to get scale data and started fitting models on it.

We selected the decision tree as our base model, and then went ahead with RandomForest and LGBMRegressor models.

Random Forest with its parameters tuned, gave us the maximum r2 score, although it was least different from Random Forest model with default Parameters.

Some important conclusions drawn from the analysis are as follows:

1. There were more sales on Monday, probably because shops generally remain closed on Sundays which had the lowest sales in a week. This validates the hypothesis about this feature.

2. There is a high correlation between Customers and Sales.

3. Stores in close proximity had more sales than those with large competition distance. This can be due to location and accessibility factors.

4. Store type B though being few in number had the highest sales average. The reasons include all three kinds of assortments specially assortment level b which is only available at type b stores and being open on Sundays as well.

5. Outliers were genuine in our dataset, and that's why instead of treating them we let them there, and went with models which are not impacted by them.

Suggestions/Recommendations

- Although this data is a bit old, given for years 2013, 2014 and 2015. Since then the drug industry has changed dramatically, especially after Covid. One can easily assume, stores even large distances have great sales.

- Still, if relevant, more stores should be encouraged to have more assortment levels, as it leads to more sales.

- Drugs are an essential commodity, so if possible, stores try to keep them open even on holidays, even if it is for a very short time.

- StoreType B should be increased in number.

9. Difficulties:

- Building these types of models requires high RAM and very huge computation abilities.

10. ***References:***

- ScikitLearn Documentation
- MachineLearningMastery
- Seaborn documentation
- GeekForGeek
- AnalyticsVidhya
- TowardsDataScience
- KrishNaik
- AlmaBetter