Rossmann Drug Store Sales Prediction

Part 1: Data Analytics

Part 2: Bayesian Network Structure Learning Analysis

Kajal Rajendra Dhawley(190030618)

1. Introduction:

Rossmann operates over 3,000 drug stores in 7 European countries. 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. We are expecting six weeks of daily sales for 1,115 stores located across Germany. Reliable sales forecasts enable store managers to create effective staff schedules that increase productivity and motivation. By helping Rossmann create a robust prediction model, we will help store managers stay focused on what's most important to them, their customers and their teams.

2. Project Aim:

Our project attempted to apply various machine learning techniques to a real-world problem of predicting drug store sales. To predict the time when there will be more customers and when there will be less. To help the staff prepared beforehand for the demand of goods in sales. Having a strong interest in Tree-Based models; this notebook is meant to show the performance of robust ensemble methods; Random Forest (parallel tree creation), KNN and Linear Regression and evaluate the best performing one for this case.

3. Literature Review:

This paper presents us the use case of data mining for sales forecasting in retail demand and sales prediction. It consists of data from various outlets of major European Pharmacy retailing company. Linear regression and Random Forest regression is used to predict the accuracy and strengthen the sales prediction.[1][4]

This paper presents us the use case of spatial data mining for aggregate sales forecasting in retail location planning. This takes all the geographical, socio-demographical and economic features into consideration which helps us to explore the hidden factors affecting the sales with the help of data.[2]

In this paper represents how to use the entity embedding method to automatically learn the representation of categorical features in multi-dimensional spaces which puts values with similar effect in the function approximation problem. The intrinsic continuity helps reveal the data and NN as well as other standard machine learning algorithms(KNN, Random Forest, Gradient Boosted Trees) to solve the problem.[3]

In this paper, there is explained that recently there have been many models and approaches to generate classifiers and aggregate the results. Boosting and Bagging are the methods of the classification tree. Random forest adds a layer of Bagging, which helps it to use the best node for prediction.[5][6]

From this paper, concepts for Data cleaning and data preparation are explained.[9][10]

4. Data Retrieval

The data utilised for the analysis was collected from Kaggle. Training data is comprised of two parts (train.csv and store.csv). One part is historical sales of each store from date 01/01/2013 to 07/31/2015. This part of data has about 1 million entries contained in train.csv file. Table 1 describes all the fields in train.csv file.

Table 1: train.csv

Field Name	Description
Store	A unique ID for each store: integer number
DayOfWeek	A day in a week: 1-7
Date	in format YYYY-MM-DD
Sales	The turnover for any given day: integer number (This is what to be predicted)
Customers	The number of customers on a given day
Open	an indicator for whether the store was open: 0 = closed, 1 = open
Promo	indicates whether a store is running a promo on that day: 0 = no promo, 1 = promo
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: 1 = school holiday, 0 = not school holiday

The second part of training data is to supplement store information. It has 1115 store info entries, which listed the store type, competitor and a different kind promotion info. Table 2 below describes all the field in store.csv file.

Table 2: store.csv

Field Name	Description
Store	a unique ID for each store: integer number
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
CompetitionOpenSinceMonth	gives the approximate year and month of the time the nearest competitor was opened
CompetitionOpenSinceYear	
Promo2	Promo2 is a continuing and consecutive promotion for some stores: 0 = store is not participating, 1 = store is participating
Promo2SinceWeek	describes the year and calendar week when the store started participating in Promo2
Promo2SinceYear	
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

5. Data Representation Tools:

To achieve the aim of the project, it is essential to handle data in the right way. To accomplish this goal data needs to be analysed in various forms such as tables and graphs. We chose Python 3 for data analysis and interactive, exploratory computing and data visualisation. The various Python libraries used throughout this project are as follows:

- Numpy: NumPy is a python library used for working with arrays. It is an open-source project, and you can use it freely. NumPy stands for Numerical Python. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists because arrays are stored at one continuous place in memory unlike programs so that processes can access and manipulate them very efficiently.

- Pandas: Pandas allows us to read, analyse, and modify data easily. Pandas provide robust data structures and functions designed to make working with structured data fast, easy, and expressive. The primary object in pandas that will be used is the DataFrame, a two-dimensional tabular, column-oriented data structure with both row and column labels.

- Matplotlib: matplotlib is the most popular Python library for producing plots and other 2D data visualizations. Matplotlib is capable of creating most kinds of charts, like line graphs, scatter plots, bar charts and many more.

- Seaborn: Seaborn is built on top of matplotlib but can do quite extraordinary things in the area of visualisation. One example of how Seaborn could be used would be to generate visualisations such as heatmaps.

- Scikit-learn: Scikit-learn is a free machine learning library for Python. It features various algorithms like linear regression, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

6. Data Preparation:

6.1 Importing the dataset:

First, we will import the panda's module and next. We will use the read_csv() to read the CSV files which contains comma-separated values and convert that into a pandas DataFrame.

The code snippet shown in appendices returns the variables store and train where the data frames are now stored. Let's display a few records from this data frame. For this, we will use the head() and tail() method.

Train 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
1017204	1111	2	2013-01-01	0	0	0	0	a	1
1017205	1112	2	2013-01-01	0	0	0	0	a	1
1017206	1113	2	2013-01-01	0	0	0	0	a	1
1017207	1114	2	2013-01-01	0	0	0	0	a	1
1017208	1115	2	2013-01-01	0	0	0	0	a	1

Store DataFrame

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval
0	1	c	a	1270.0	9.0	2008.0	0	NaN	NaN	NaN
1	2	a	a	570.0	11.0	2007.0	1	13.0	2010.0	Jan,Apr,Jul,Oct
2	3	a	a	14130.0	12.0	2006.0	1	14.0	2011.0	Jan,Apr,Jul,Oct
3	4	c	c	620.0	9.0	2009.0	0	NaN	NaN	NaN
4	5	a	a	29910.0	4.0	2015.0	0	NaN	NaN	NaN
1110	1111	a	a	1900.0	6.0	2014.0	1	31.0	2013.0	Jan,Apr,Jul,Oct
1111	1112	c	c	1880.0	4.0	2006.0	0	NaN	NaN	NaN
1112	1113	a	c	9260.0	NaN	NaN	0	NaN	NaN	NaN
1113	1114	a	c	870.0	NaN	NaN	0	NaN	NaN	NaN
1114	1115	d	c	5350.0	NaN	NaN	1	22.0	2012.0	Mar,Jun,Sept,Dec

6.2 Data Cleaning:

6.2.1 Dealing with missing values:

Missing data can cause various problems. It can reduce the statistical power of a study and can produce biased estimates, leading to invalid conclusions. The method we follow in dealing with missing values are discussed below:

- **Count missing values:** First, we count the missing values in each data set using IsNull()function. After running IsNull()function on both datasets store and train, we get the following result:

```
Store Dataset:

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

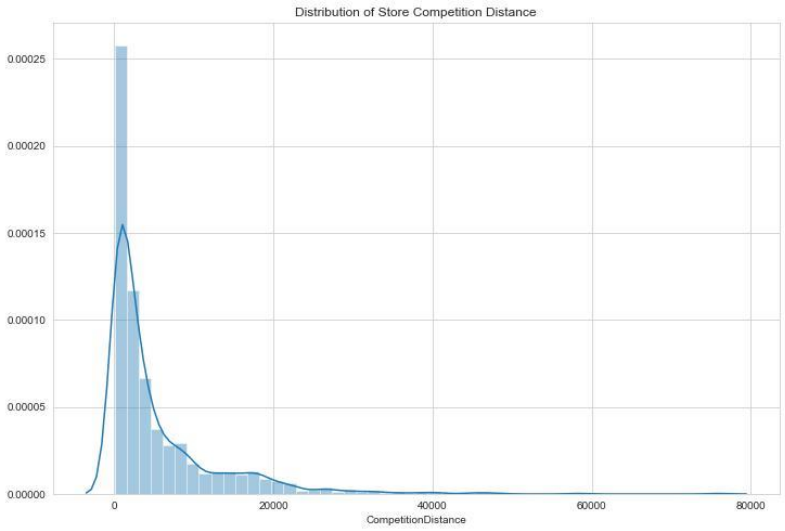
```
Train Dataset:

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

We observed that for dataset train, we get no missing values. For the dataset store, we get missing values for variables 'CompeteionDistance', 'CompetetionOpenSinceMonth', 'CompetitionOpenSinceYear', 'Promo2SinceWeek', 'Promo2SinceYear' and 'PromoInterval'. We calculated the proportion of missing values for each variable. For variable 'CompetitionDistance' only 0.2% (3 out of 1115) values were missing. For variables 'CompetetionOpenSinceMonth' and 'CompetitionOpenSinceYear' around 31.74% (354 out of 1115) values were missing. For variables 'Promo2SinceWeek', 'Promo2SinceYear' and 'PromoInterval' 48.78% (544 out of 1115) values were missing.

- **Features to remove:** After counting missing values we observed that some variables have a high percentage of missing values and they won't be accurate as indicators, so using drop()function, we remove features with more than 30% missing values.

- **Replace missing values:** Missing values in 'CompetitionDistance' feature were only 0.2%, so we decided to replace the missing values. Using Seaborn we obtain the Distribution graph for 'CompetitionDistance' shown in Figure 1.

Figure 1:



We observed that distribution is right-skewed so we will replace missing values with the median.

6.2.2 Transforming Data:

- **Data Extraction**: We wanted to observe the behaviour of sales with respect to Year, Month, Day and Week of the Year so we decided to extract the each of these variables from the date. Refer Appendix for code. The datafarme “train” has 4 new columns namely, Year, Month, Day and WeekOfYear.

- **Joining tables:** We then decided to join our two dataframes train and store before we do further analysis we desire. We used merge() function to join both dataframes and named the output dataframe df. At this point df has 1017209 observations.

- **Drop data that can cause bias:** After exploring data we find out that Open variable in dataframe has 0 values that means the store was closed on corresponding day, so the sales were also 0 for that day. We decided to remove this part of dataset. Open only has only one value i.e 1 so its not a variable anymore so we drop it. Now we are left with 844392 observations. Then we checked any open stores with 0 sales. We find out 54 such cases out of 844392. Then we decided to drop this part of data as well to avoid bias.

6.2.3 Feature Engineering

Modifying Datatypes: We decided to see variable types of dataframe and we get following observations after running the info() function on dataframe df:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 844338 entries, 0 to 1017190
Data columns (total 16 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Date                  844338 non-null  datetime64[ns]
 1   Store                 844338 non-null  int64
 2   DayOfWeek             844338 non-null  int64
 3   Sales                 844338 non-null  int64
 4   Customers             844338 non-null  int64
 5   Promo                 844338 non-null  int64
 6   StateHoliday          844338 non-null  object
 7   SchoolHoliday         844338 non-null  int64
 8   Year                  844338 non-null  int64
 9   Month                 844338 non-null  int64
 10  Day                   844338 non-null  int64
 11  WeekOfYear            844338 non-null  int64
 12  StoreType             844338 non-null  object
 13  Assortment            844338 non-null  object
 14  CompetitionDistance   844338 non-null  float64
 15  Promo2                844338 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(11), object(3)
memory usage: 109.5+ MB
```

After looking at the variable types nominal values are printed for variables StateHoliday, StoreType and Assortment using set()function.

output:

```
Out[90]: ({0, '0', 'a', 'b', 'c'}, {'a', 'b', 'c', 'd'}, {'a', 'b', 'c'})
```

As we can see that for variable StateHoliday has 0 as integer as well as string type. We convert all integer types of 0 to string type using astype(str) function.

- **Add new features/variables:** We decided to add new features to our dataset to improve our accuracy of prediction. We added AvgSales (monthly average sales) and AvgCustomer (monthly average customers) for each store using mean()function. To do this we used groupby() function to group Year, Month and Store variables, so that while calculating average, sales don't get repeat along with these grouped variables.

- **Transform variable:** As we observed before StateHoliday variable has 4 different types of values (a = public holiday, b = Easter holiday, c = Christmas, 0 = None). We decided to keep only 2 values (0 - not a state holiday; 1- is on a state holiday).

7 Data Exploration and Visualisation:

Prior to any modelling , a great amount of attention was devoted to data exploration . We believe that this step was crucial in identifying factors affecting sales . Extensive exploration

of the data was carried out as we believe exploration and visuals helps us understand the data and grasp critical information that could have been easily missed .

7.1 Sales Distribution

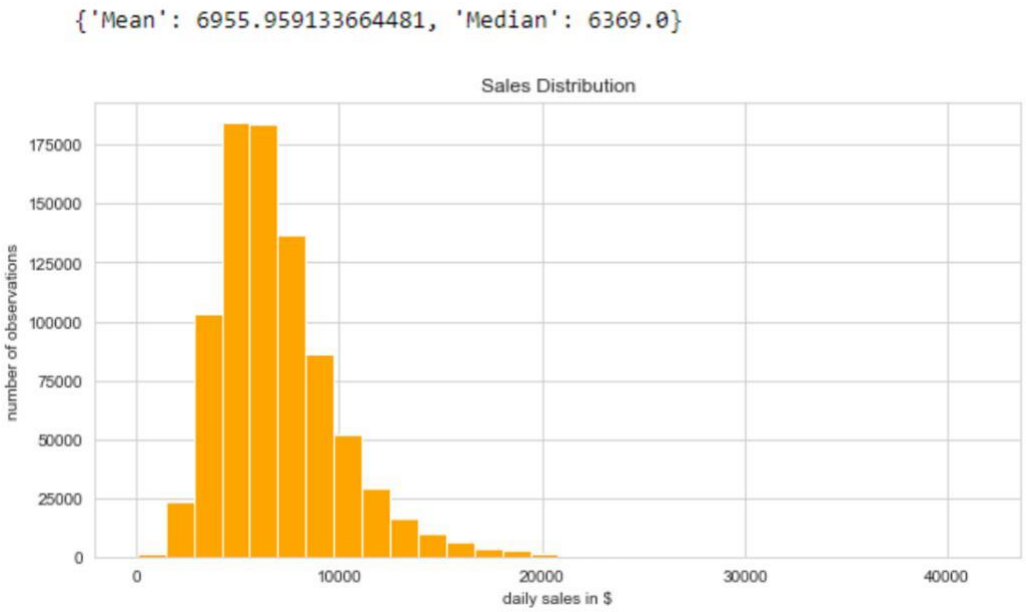


Figure 1. Sales Distribution

- The big amount of sales on certain days is either due to promotional purposes , the type of the store being big and popular or just not having near enough competition and being the monopoly in its region.
- There are 0 of the time having 0 sales , those cases could happen for some shops having external events affecting it.(an incident, a manifestation etc.)
- Also here a mean of 6955 versus 6369 in median is a very good sign that there are no extravagant values affecting the general distribution of Sales.

7.2 Customer Distribution

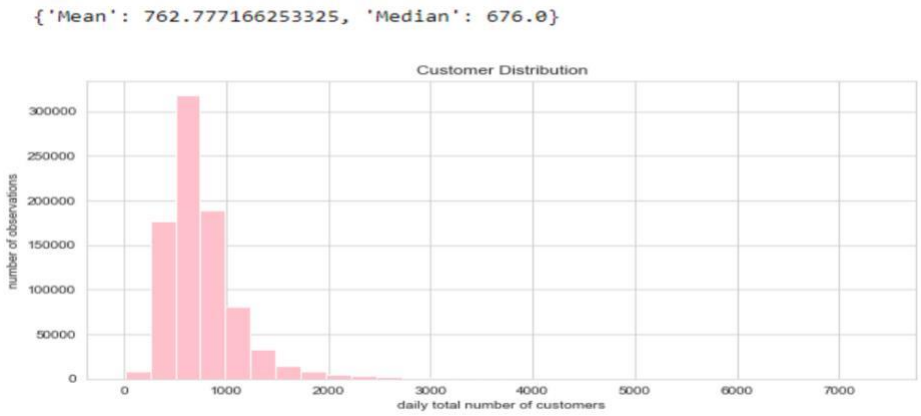


Figure 2. Customer Distribution

- We can see similar patterns with the customers distribution and the Sales distribution, in fact the correlation factor of 0.82 explains that there is a strong positive correlation between Sales

and Customers. In general, the more customers you have in a store, the higher your sales for the day.

- We see that on a specific day there was a huge amount of customers in a store, this was due to a big promotion going on. Those specific values are affecting the mean which concludes the difference between a mean of 762 and a median of 676.

7.3 Sales Over Time

Figure 3. Sales Over Weeks of a Year.

In the above bar plot graph between Weeks of a year and total sales it is observed that maximum sales is observed in the 51st week of the year i.e. Christmas week and there is also considerable increase in sales in 18th ,31st , 49th and 52nd week where sale exceeds 8000 .

Figure 4. Sales Pattern in 3 years (2013, 2014, 2015)

In sales over 3 years graph we can see that almost same pattern of purchase is followed every year. There is peak in sales somewhere near May and in the month of December .

7.4 Sales vs Competition Distance

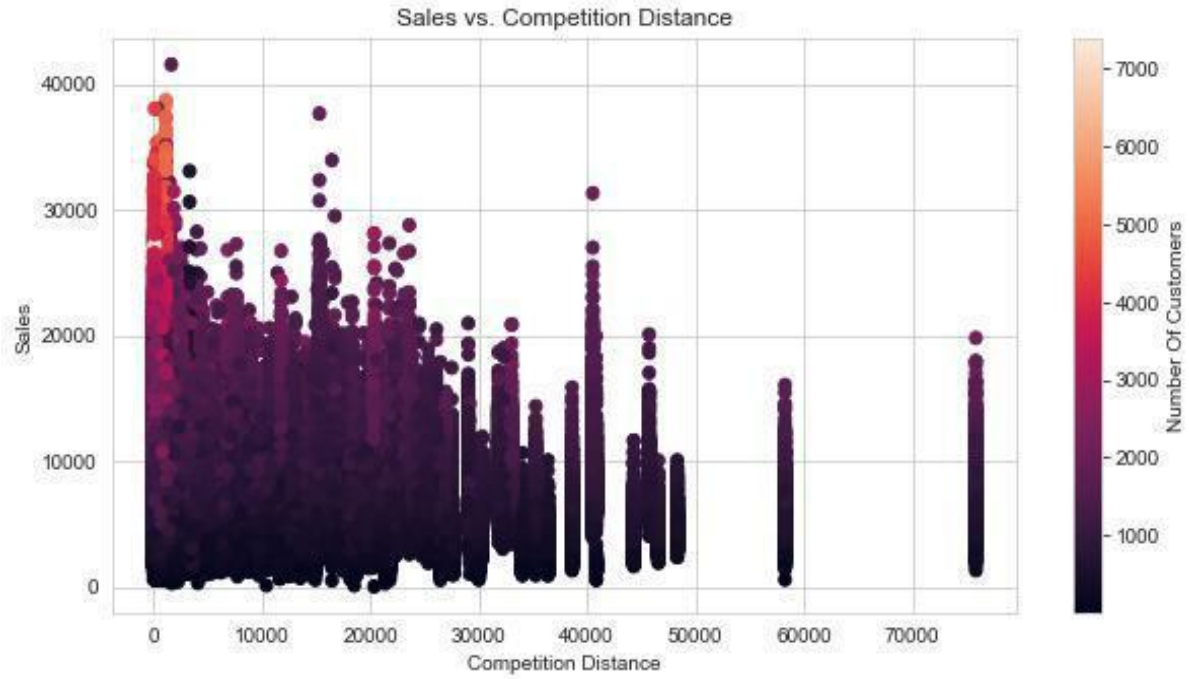


Figure 5. Sales vs. Competition Distance

- This scatter plot graph is used to observe the sales and competition distance with respect to number of customers.
- Ideally more the competition distance , more is the average sale due to the fact that when there are no competition nearby , stores tend to sell more and have more customers because they are almost a monopoly in their region . But here , contradictory observation is seen in the sales pattern with respect to competition . From the graph it is found that there is maximum sales in 0 competition distance i.e above 4000 with highest number of customers i.e around 6000 . Also there is considerable increase in sales with more competition distance which shows that competition distance is positively related to sales but the relation is not so strong.

7.5 Impact of promotion on sales over days of a week

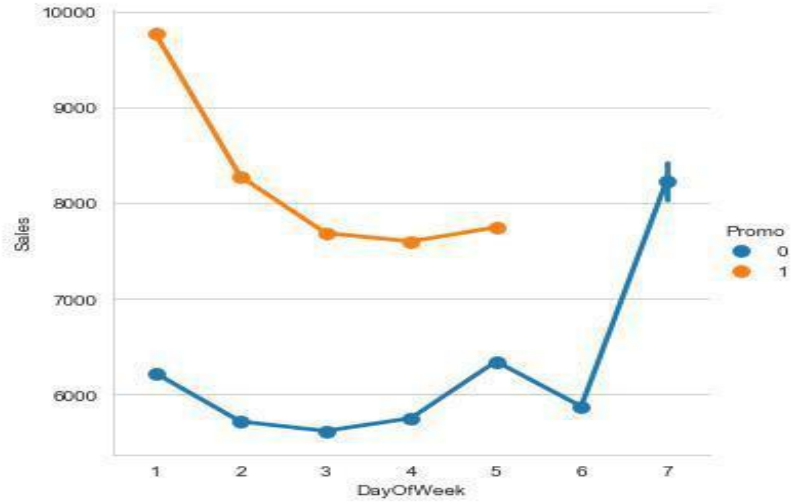


Figure 6. Sales over days of a week when promotion is applied.

```
Number of Stores opened on Sundays:33
```

- We see the dramatic change when we compare having promotion Promo=1 to not having promotion Promo=0 and can conclude that a store that have promotion on a given day shows considerable increase in sales .
- We also see that there are no promotions during the weekend as very few stores opens on Sundays (only 33) so if anyone needs anything urgently and don't have the time to get it during the week, he will have to do some distance to get to the open ones even if it's not close to his house.
- After attempting to look at Sales behaviour in a week over the Years and over the months, concluded that the pattern doesn't change, which means all the time there is a peak on Mondays with promotions, a tiny peak on Friday before the weekend and a big peak on Sunday because of closed stores.

7.6 Sales By StoreType and Assortment

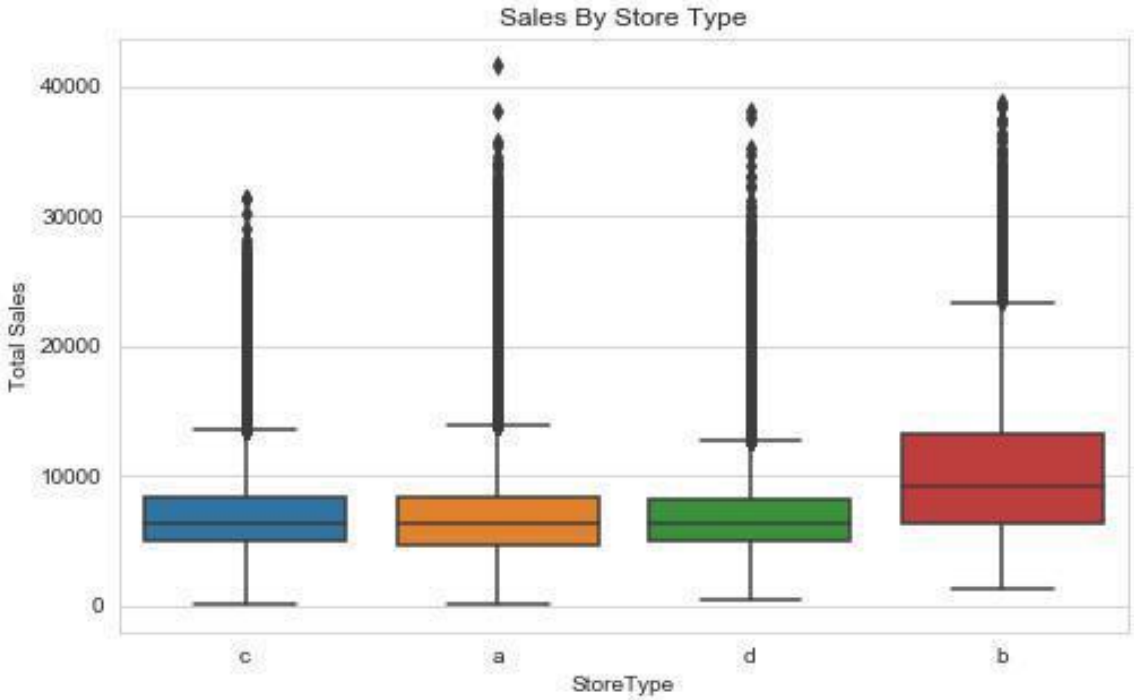


Figure 7. Sales By Store Type

- Here , we use boxplot to observe the Sales in 4 Type of Stores (a,b,c,d) . As it can be seen , b type of store has higher sales compared to other store type.

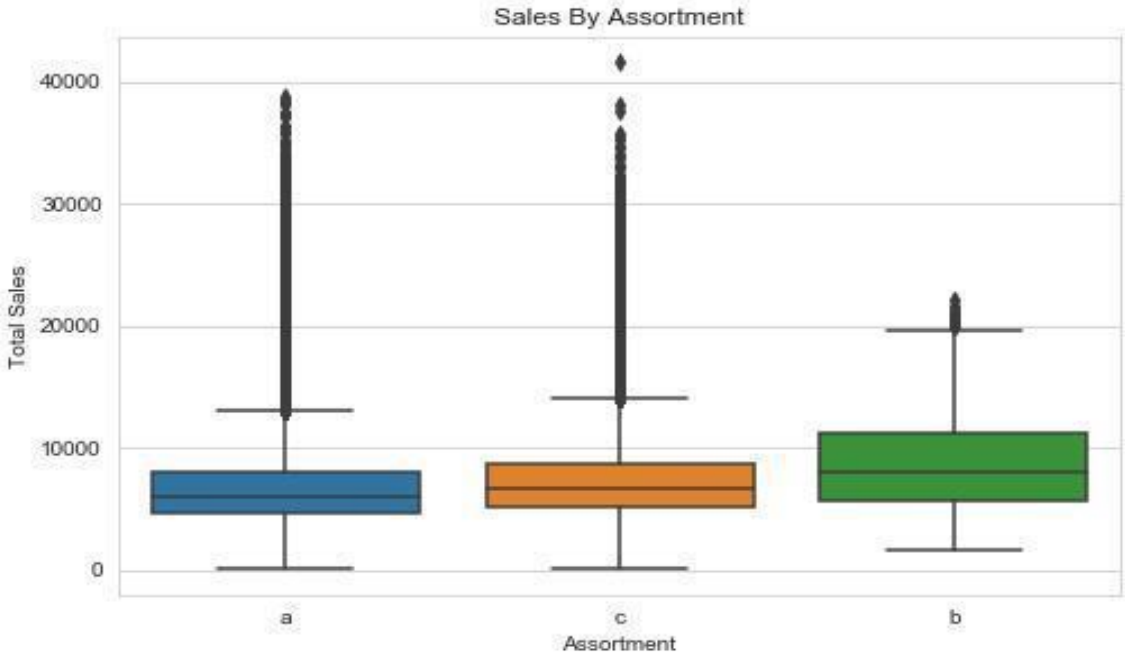


Figure 8. Sales By Assortment

Here boxplot is used to observe the Sales in 3 type of Assortment (a = basic , b = extra ,c = extended) . As it can be seen that B type of assortment has higher sales compared to other types , as B is of type "extra", it is the middlepoint for customers between not too much variety like C assortment and not too basic like A assortment and this is what is driving the high traffic in such type of stores resulting in higher sales.

7.7 Assortment-wise sales over months with respect to store type.

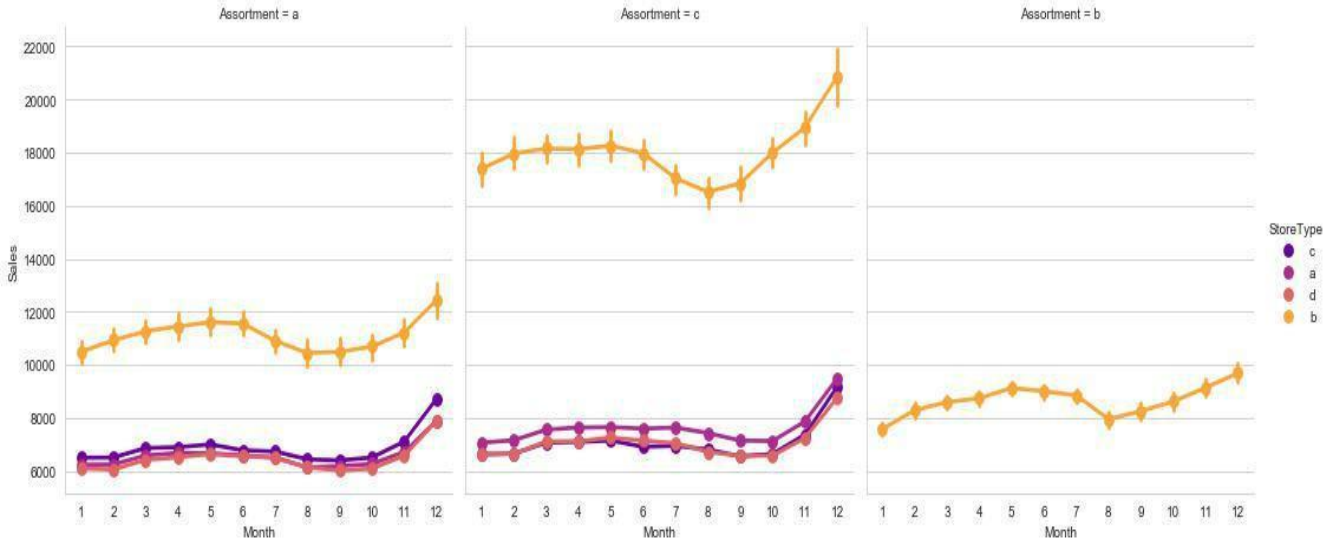
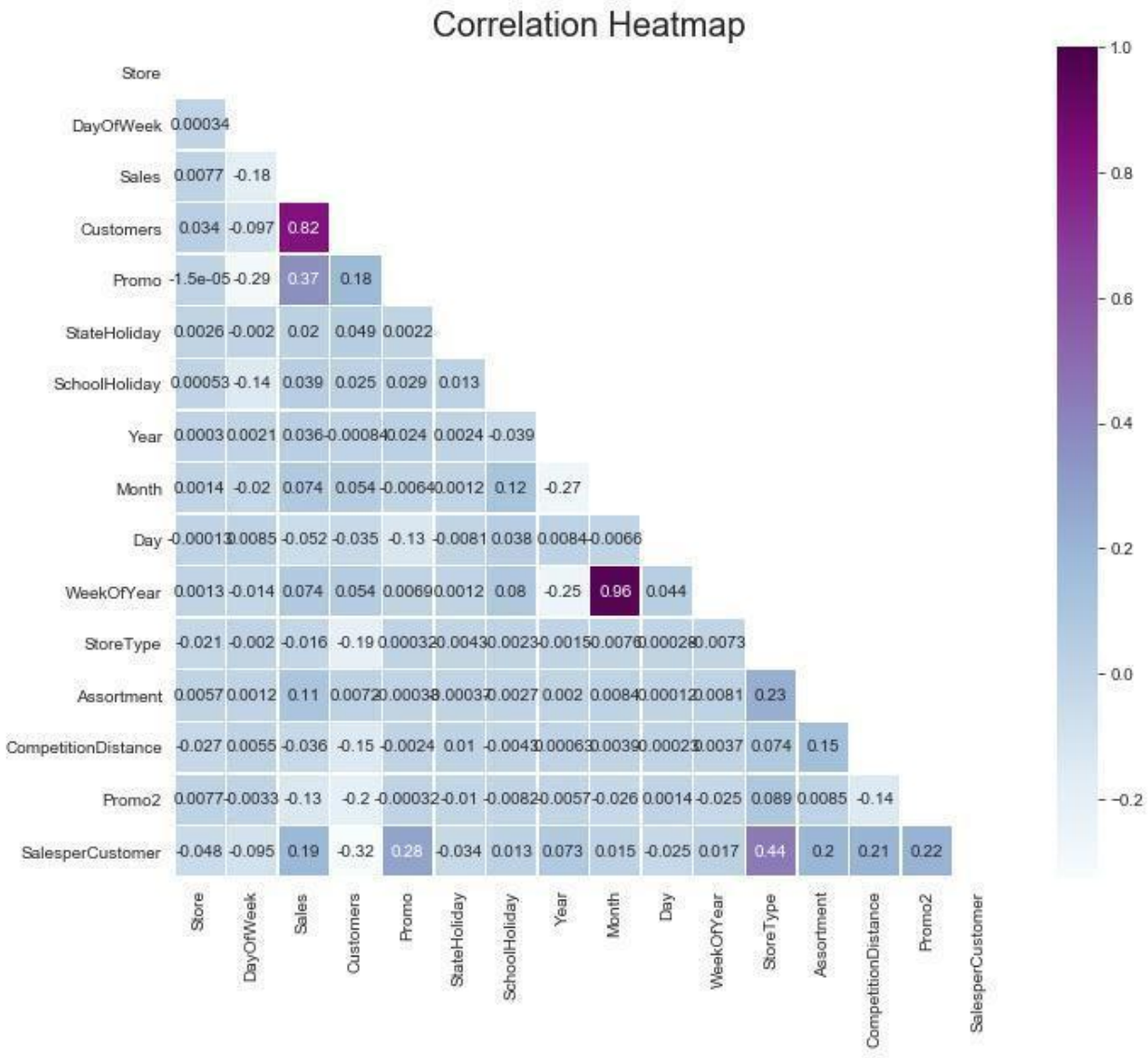


Figure 9. Assortment-wise sales over months with respect to store type.

- We can clearly see here that most of the stores have either a assortment type or c assortment type. Another important factor here is the fact that store type b is the only one who has the b assortment type. If we compare these findings with Sales By Store Type chart , we see that B type store has higher sales compared to other store type .

7.8 Correlation Heatmap



- We can see that Sales and Customers are positively correlated (0.82) as more the customers in the store , higher will be the sales for that day.
- Sales per Customer and Promo (0.28) actually correlate positively, since running a promotion increases the sales.
- Sales per Customer also correlates with Competition Distance (0.21) , higher the competition distance more will be the sales per customer as it won't be having near enough competition in its region.
- Additionally, the effect of Promo2 to Sales per Customer like we said above as well(0.22), it did provoke a change in the buying pattern and increased it when continuous promotion were applied.
- Sales per Customer also correlates with StoreType and Assortment , which is concluded from previous observations .

8 Modelling:

We used three different approaches in order to predict store sales with respect to time. Each method was trained on data from a single store, and then used to predict the sales for that particular store. This process was subsequently repeated for every store. We had divided the dataset into training and test sets in a 70-30 ratio before building any of the models. Every model was initially built using a part of the training data and the other part was used for validation. The model that gave the best validation score was run against the test set.

8.1 Linear Regression

Linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables. It is used to fit a predictive model to a set of observed values. This is useful if the goal is prediction or forecasting or reduction.

8.1.1 Model Implementation

Equation of linear regression

y=c+m1x1+m2x2+...+mnxny=c+m1x1+m2x2+...+mnxn

yy is the response

cc is the intercept

m1m1 is the coefficient for the first feature

mnmn is the coefficient for the nth feature

The mm values are called the model coefficients or model parameters.

We started the implementation of Linear regression by splitting our variable into Training and Testing sets. It is split in such a way that 70% of data is used for training whereas 30% for testing.

We need to import the libraries from sklearn_model selection. By default, the library consists of line in the dataset which passes through origin. A constant is added such that it is used to X_train dataset. Root Mean Square Error (RMSE) as well as Mean Absolute Percentage Error (MAPE) are also calculated for better accuracy.

OUTPUT :

```
Regresion Model Score : 0.7486872962950808 , Out of Sample Test Score : 0.7482884273948069
Training RMSE : 1552.8927784361235 Testing RMSE : 1556.0327257032725
Training MAPE : 16.996208120525296 Testing MAPE : 17.053657649224167
```

8.2 Random Forest Regression

- Random forest is a supervised learning algorithm. It consists of both classifications as well as regression. However, It is generally used for classification problems.
- As we know that a forest is made up of trees and more trees means more rich forest. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution utilizing voting.
- With the help of this model, we will predict the sales based on the dataset with various variables.

8.2.1 Model Implementation:

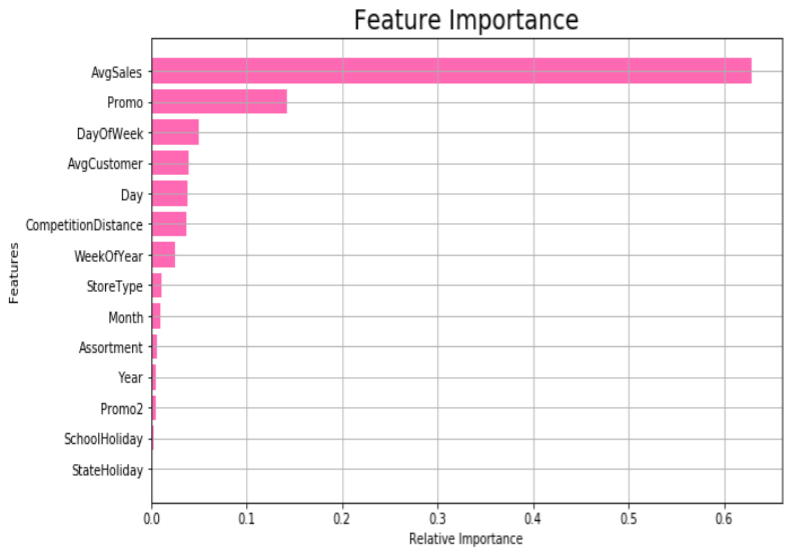
- By using the decision tree we found which features are more important for the sales. With the help of feature importance we found out that AvgSales and Promo are the main factors that affect the sales.
- These features which are important are shown below.
- Random forest gives us the training accuracy of 0.91 with the testing accuracy of 0.84.
- This states that this model is the most accurate model among the rest of the models that are taken into consideration.

OUTPUT:

```
Regresion Model Score : 0.987805201715366 , Out of Sample Test Score : 0.919520314533873
Training RMSE : 342.0751763971179 Testing RMSE : 879.8533062133778
Training MAPE : 3.419915422308864 Testing MAPE : 9.029892309616939
```

8.2.3 Feature Importance

- Random Forests are biased in favour of categorical variables with different number of levels while calculating the feature importance.
- The feature importance as ranked by the Random Forest is as shown in Figure below.



	importance
AvgSales	0.629408
Promo	0.141839
DayOfWeek	0.050408
AvgCustomer	0.039626
Day	0.037671
CompetitionDistance	0.036589
WeekOfYear	0.024763
StoreType	0.011059
Month	0.009620
Assortment	0.005950
Year	0.005153
Promo2	0.004339
SchoolHoliday	0.002688
StateHoliday	0.000887

8.3 KNN (K –nearest Neighbour Algorithm)

- The basic theory behind KNN is that in this dataset, it finds a group of k samples that are nearest to unknown samples (e.g., based on distance functions). From these k samples, the label (class) of unknown samples are determined by calculating the average of the response variables (i.e., the class attributes of the k nearest neighbor).
- As a result, for this classifier, the k plays an important role in the performance of the KNN, i.e., it is the key tuning parameter of KNN. The parameter k was determined using a bootstrap procedure. In this study, we examined k values from 1 to 30 to identify the optimal k value for all training and test sample sets along with the model score.

OUTPUT:

```
Regresion Model Score : 0.6524756271336151 , Out of Sample Test Score : 0.6276810088437411
Training RMSE : 1826.1097834629309 Testing RMSE : 1892.451376032131
Training MAPE : 21.949007513402734 Testing MAPE : 22.78498743408952
```

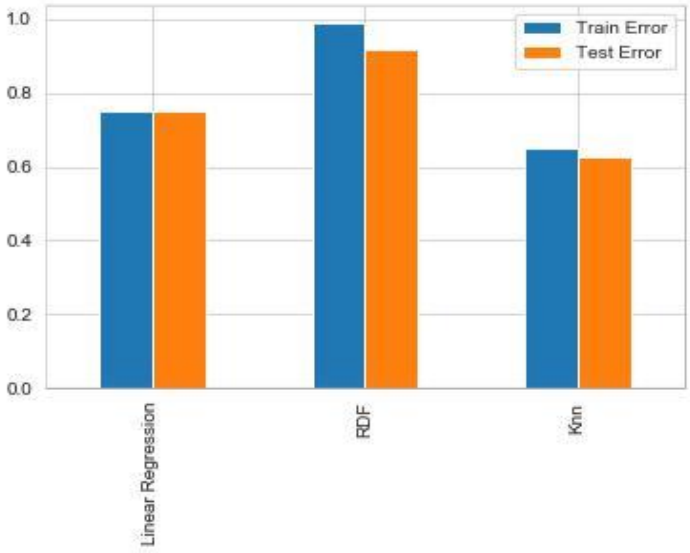
```
Regresion Model Score : 1.0 , Out of Sample Test Score : 0.6474933693252644
Regresion Model Score : 0.9124957388330857 , Out of Sample Test Score : 0.6738642587944125
Regresion Model Score : 0.8558174129627617 , Out of Sample Test Score : 0.654988933079603
Regresion Model Score : 0.8074537574447423 , Out of Sample Test Score : 0.6305046218343373
Regresion Model Score : 0.7661210746885609 , Out of Sample Test Score : 0.6116717543473327
Regresion Model Score : 0.7332587315281955 , Out of Sample Test Score : 0.5996704879753836
Regresion Model Score : 0.7088888389971022 , Out of Sample Test Score : 0.5927066713295375
Regresion Model Score : 0.6912462761266187 , Out of Sample Test Score : 0.5896593844072082
Regresion Model Score : 0.6789068106839262 , Out of Sample Test Score : 0.5900888561809591
Regresion Model Score : 0.6711675120464007 , Out of Sample Test Score : 0.5930780574676395
Regresion Model Score : 0.6671065326830901 , Out of Sample Test Score : 0.5980299698529088
Regresion Model Score : 0.6654764516937683 , Out of Sample Test Score : 0.6036507549411689
Regresion Model Score : 0.6655973352279224 , Out of Sample Test Score : 0.6091409302025252
Regresion Model Score : 0.6665783708087091 , Out of Sample Test Score : 0.614369258506029
Regresion Model Score : 0.6676648899612242 , Out of Sample Test Score : 0.6188506104781967
Regresion Model Score : 0.6686478819810026 , Out of Sample Test Score : 0.6222962862910699
Regresion Model Score : 0.6689565632948903 , Out of Sample Test Score : 0.6245360973069839
Regresion Model Score : 0.6684172577789975 , Out of Sample Test Score : 0.6257855842941429
Regresion Model Score : 0.6672759733713085 , Out of Sample Test Score : 0.625884671433985
Regresion Model Score : 0.6656352012710725 , Out of Sample Test Score : 0.6256456867217889
Regresion Model Score : 0.6636807709149197 , Out of Sample Test Score : 0.6249714668195651
Regresion Model Score : 0.6616916084413502 , Out of Sample Test Score : 0.6242817823442275
Regresion Model Score : 0.6597483841088833 , Out of Sample Test Score : 0.6237416157605714
Regresion Model Score : 0.6579271623505407 , Out of Sample Test Score : 0.6233051902723898
Regresion Model Score : 0.6563877755644907 , Out of Sample Test Score : 0.6231003368164729
Regresion Model Score : 0.6551298375475234 , Out of Sample Test Score : 0.6231849945095151
Regresion Model Score : 0.6541583464711902 , Out of Sample Test Score : 0.6233862852722492
Regresion Model Score : 0.6535182976545827 , Out of Sample Test Score : 0.6238512020311512
Regresion Model Score : 0.6529102853941815 , Out of Sample Test Score : 0.6242451453462318
  I
```

8.4 Model Comparison

- We compared the RMSPE and MAPE values of our models Linear Regression, KNN and Random Forest. For our validation set, we found that the error values of the Linear Regression where relatively more as with the accuracy score of the model was a bit lower than when implemented the same model using Random Forest .
- Random Forest model gave a RMSPE and MAPE values that are 1% of our baseline models. This 1% error will actually translate to a large number of sales, thus it is a huge improvement over our baseline models.

- When we compared the training and testing sets for all the models, we found of the accuracy of Random Forest Regression exceeds the value of Linear Regression model and KNN is having the lowest value as per the graph below.

	Train Error	Test Error
Linear Regression	0.748687	0.748288
RDF	0.987805	0.919520
Knn	0.651492	0.627701



9 Challenges:

- During this whole data prediction there were multiple challenges and successes that we came across. In this dataset there were columns which had more missing values and unwanted column due to which we dropped them.
- In the data visualization part, initially we used chart studio to plot the graphs and in chart studio there is limit in the amount of data samples used to plot the graph i.e. 5000 data samples and due to this less data the graph was unable to provide proper result. To overcome this challenge, we used matplotlib and seaborn to plot the graph instead of chart studio and all the data from the dataset were taken into consideration. After doing so, the graph was properly plotted by providing us a proper visualization of the dataset.
- There were multiple variables which also has categorial data in the dataset, it makes the process more complex and tedious. Due to which we converted this categorial data into Integer values.

10 Business Applications:

- By using the results that are obtained from the dataset. It can be easily predicted when there will be more demand of goods. If the demand is considered beforehand and all the stocks are available during the sales. Then there will be boost in the sales and it will create more benefit for the stores as well as the vendors.
- There will be increase in the chances of employment during the month when there is maximum sales, as there will be more requirements of staff to help the stores during crucial time it will lead to the employability of the people who needs it.

Key Findings :

- It was found that there is a strong relation between the sales and customers as well as promos and sales.
- We learned that Random Forest gives us higher precision and accuracy.

11 Conclusion and Future Scope:

Conclusion

The Rossmann Store Sales problem is a very interesting data science problem to solve. We observed that the problem is more focused on feature engineering and feature selection than on model selection. We spent around 70% of our time analysing the data for trends in order to make feature selection easier. We attempted several models using different features and assumptions. We concluded that ensemble learning improves the prediction accuracy considerably. It was found that sales in the month of December was maximum which led to increase on the workload of the staff. Due to which they can prepare the schedule of their work in the month of December beforehand to reduce the burden during that period. We learned that Random Forest gives us higher precision and accuracy. However, the other two models where more elaborative about the correlation between variables and gave an insight to feature description and selection.

Future Scope

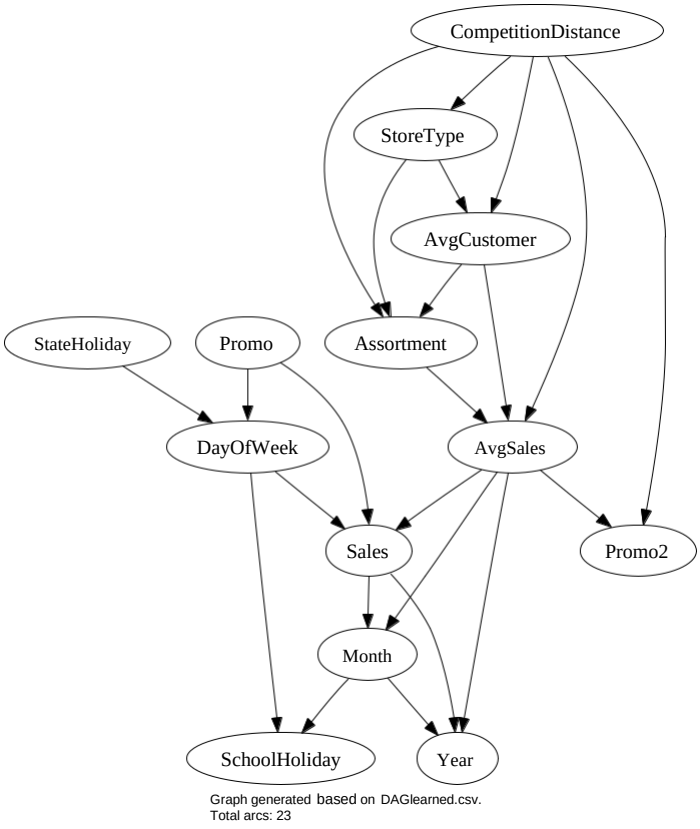
The factors which we can be improved are as follows:

- In addition to predicting sales predicting the demand is another solution which would immensely benefit stores. Prior knowledge of what products will be in demand will help stores stock up on right item.
- Identifying which customers will react positively to certain promos and offers to ensure they receive them. Conversely identifying the customers who do not like the certain promos will help reduce sending those promos.
- This solution involves identifying the appropriate discounts for different items so, as to maximize the revenue. Identifying the right product will help generate profit as well as clear excess stock.

12 Bayesian network structure learning analysis

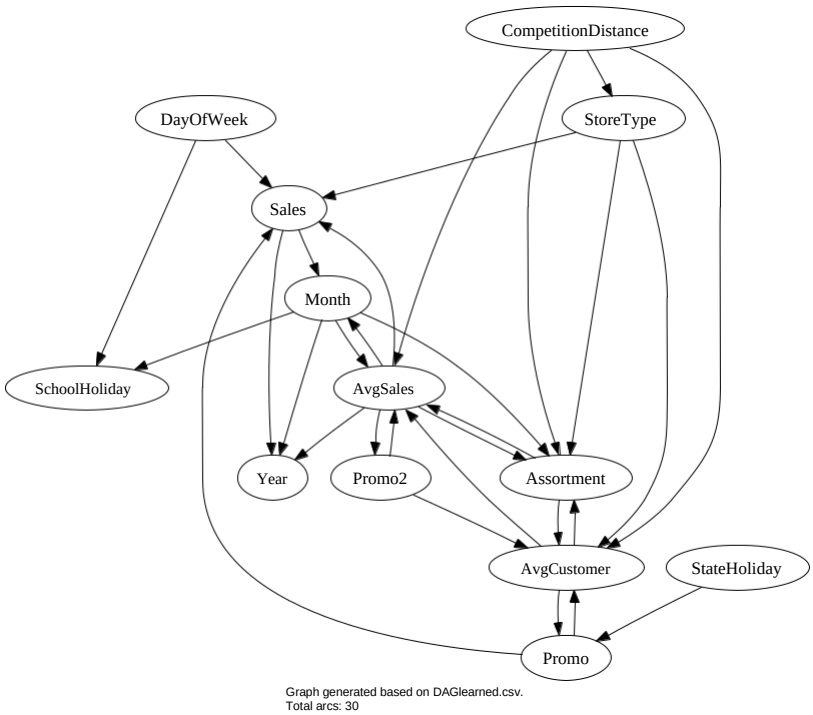
DATASET: Rossman Store Sales

1 DAGlearned graph



_____ Evaluation _____
Nodes: 13
Sample size: 904263
TrueDAG arcs: 30
TrueDAG independencies: 48
LearnedDAG arcs: 23
LearnedDAG independencies: 55

2 Knowledge Based Graph:

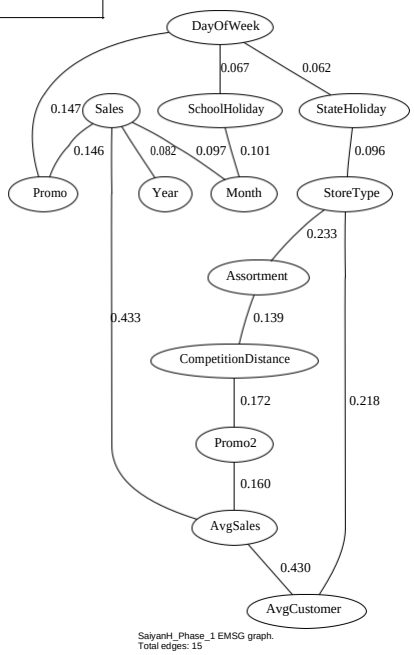


Answer 1: We took the following steps to produce Knowledge based graph:

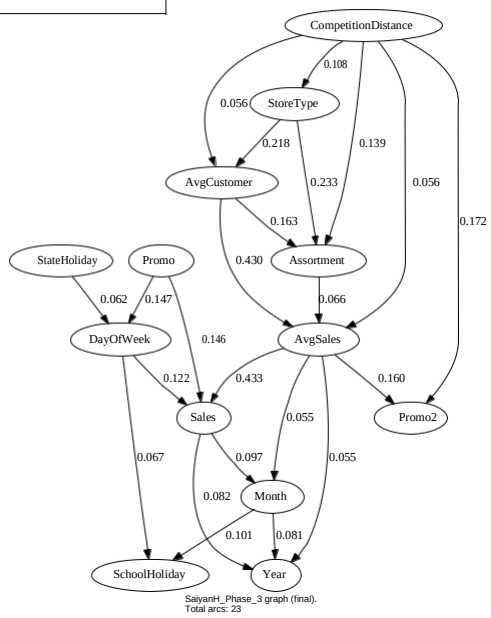
1. We categorized each variable into Cause, effect or both.
2. Then created a 2 column table in which first column contains all the cause variables and second column containing effect or both variables.
3. Then for every row we started eliminating variables from column 2 by checking dependency of variable in column 2 with variable in column 1.
4. The step 3 was taken on the basis visualisations, knowledge gained through reading papers on retail systems prediction and subjective reasoning of group members (sometimes through voting in case of disagreements).
5. After eliminations we convert the table into csv file.

3 Three SaiyanH graphs:

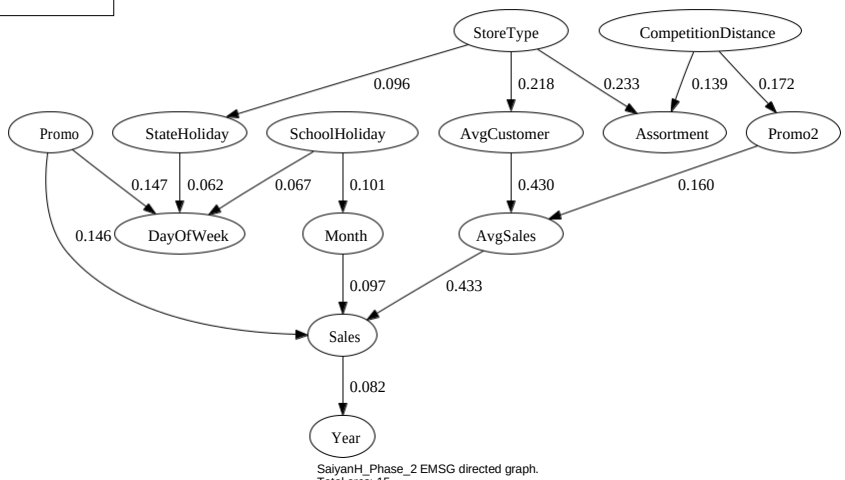
Phase 1:



Phase 3:



Phase 2:



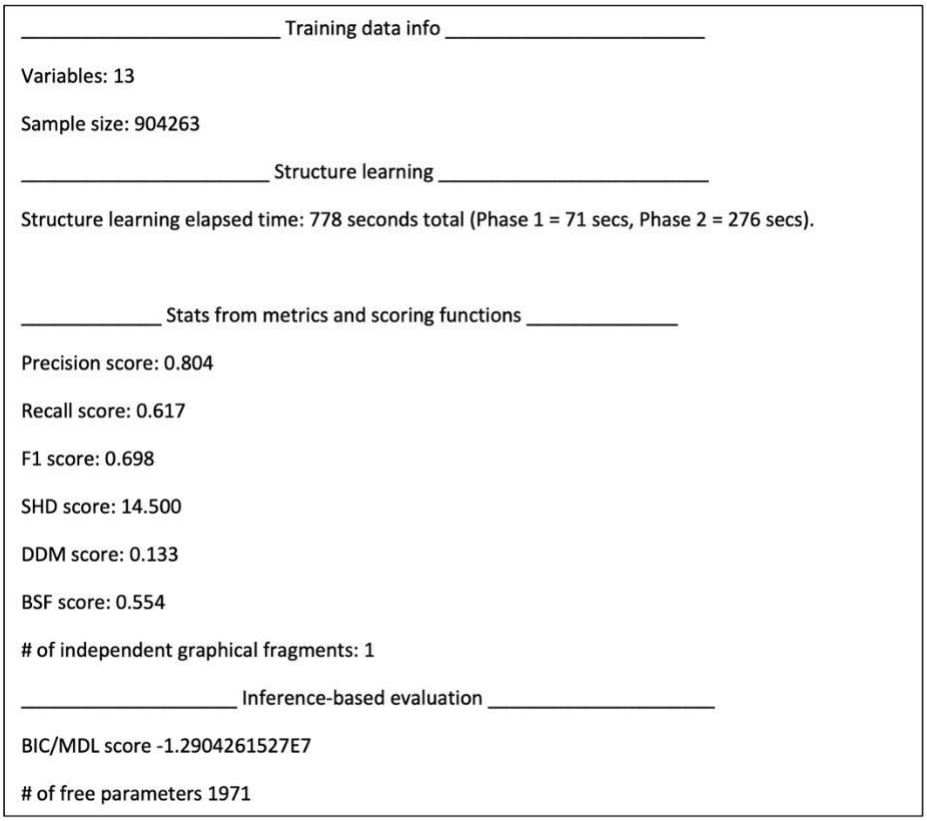
Answer 2: The Difference in graphs produced during Phase 1 and Phase 2 is that First phase is an undirected graph called as Extended Maximum Spanning Graph (EMSG) that is created using pair-wise associational score Mean/Max/MeanMax Marginal Discrepancy (MMD) whereas Phase 2 graph is a directed acyclic graph generated by applying Constraint-based learning to undirected graph in phase 1 . Conditional independence test is applied to all pairs of nodes conditional on the remaining nodes i.e in set of triples and classifies each triple into conditional dependence , independence or insignificance to determine orientation of edges . In order to determine the starting node to begin orienting the edges , nodes are ordered based on total MMD score they share with their neighbours and the node with highest score serves as the starting node , then edges from that node are evaluated .

Answer 3: The difference in graphs produced during Phase 2 and Phase 3 is that Second phase graph is produced by applying Constraint-based learning to undirected graph in Phase 1 whereas in phase 3 , search method is applied to determine best graph . It uses BIC to score graphs and the graph with highest BIC score is selected . Second phase graph act as a starting graph for search in phase 3 . The search starts with Hill-Climbing approach where all the neighbouring graphs are explored and whenever the graph with highest BIC score is explored , it is replaced with the previous graph . This process continues until no neighbour with highest BIC score exists . This process may lead to possibility of local maximum . So to escape from this , Tabu search is performed where neighbours of graph with minimal BIC are explored to determine the graph which improves BIC and thereby replacing it with the previous one . This process continues until all possible graphs are explored .

Answer 4:

Files	Number of Scores
marginalDep.csv	78
conditionalDep.csv	58
conditionalIndep.csv	39
conditionalInsignificance.csv	761

In marginalDep.csv, MMD score is calculated in all pairs of nodes and in conditionalInsignificance.csv, conditional independence tests is applied across all pairs of nodes conditional on remaining nodes i.e in set of triples. That is why there is huge difference in the number of scores in both the files.



Answer: 5 Our Score with sample size 904263 (≈ 904 ≈ 1000). We find 4 case studies that return result for sample size = 1000K. Relating our scores to Fig 2 of research paper:

Case Study	F1 Score:	SHD Score:	BSF Score
Alarm	0.8	20	0.8
Asia	0.9	1	0.9
Property	0.8	10	0.8
Sports	0.9	2	0.9
Rossman (Expected)	0.85	8.25	0.85
Rossman (Actual)	0.698	14.5	0.554

SHD score is higher than expected and F1 and BSF score is lower than expected. This could be because SHD returns classification accuracy which in our case is more than expected. The other 2 metrics will balance the score so their score is less than expected.

Answer 6:

Our Structure learning elapsed time: 778 seconds

Our System Properties:

Speed = 2.5GHz | # of cores = 2 | Intel Turbo Boost Technology = No

Our Sample Size = 904263 (≈ 904 ≈ 1000)

Table from research paper with 1000K samples:

Sr. No.	Nodes	True edges	Max in-degree	# of parameters	Runtime
1	27	31	3	3056	1400
2	13	30	-	1971	778

The results from Table 2 of research paper shows that runtime increases rapidly with the number of nodes and the sample size of the input data. We compared our results (Sr. No. = 2) to one of the observations (Sr. No. = 1). As in our case nodes get almost halved so we expected the runtime to be decreased with same proportion. And the result was same as we expected, our run time is close to half the run time of the referred observation. System properties can also affect the runtime.

Answer 7: Step 4:

BIC/MDL score -1.2904261527E7

Step 2:

BIC/MDL score -1.3174211298E7

$$= (|) - (\quad _2 \quad \overline{\quad\quad 2} \quad)$$

$$F = \sum ((\quad -1) \quad)$$

Drop in BIC score tells us when to stop adding arcs. BIC score decreases when new arc enters a node that already has enough arcs it should have. When new arc enters the variable having sufficient arcs, its states ( ) are also multiplied to states of its new parent plus the states of other parents, producing more free parameters. Then dimensionality of model increases exponentially. So penalty increases.

Model obtained in step 2 (DAGtrue) has more such cases than model obtained in step 4 (DAGlearned), so DAGlearned gave more BIC score.

With sample size BIC score becomes less important. It comes closer to LL score. So we don't see much difference as expected.

Answer 8: Step 4:

# of free parameters 1971

Step 2:

# of free parameters 24638

Model obtained in step 2 has nodes with more parent nodes. Free parameter (F) increases if node has more parent nodes because states of node multiplies to the states of all parent nodes.

Model obtained in step 4 has nodes with less parent nodes so less free parameters.

13 References:

[1] Ankur Jain, Mamballykalathil N. Menon, Saurabh Chandra, Sales Forecasting for Retail Chains, Published 2015.

[2] Simon Scheider et. all Maike Krause-Traudes. Spatial data mining for retail sales forecasting. 11th AGILE International Conference on Geographic Information Science, 2008.

[3]Entity Embeddings of Categorical Variables Cheng Guo∗ and Felix Berkhahn† Neokami Inc. (Dated: April 25, 2016)

[4] Brian Knott, Hanbin Liu, Andrew Simpson, 'Predicting Sales for Rosmann Drug store', Published 2015.

[5]A. Liaw and M. Wiener (2002). Classificationand Regression by randomForest. RNews, 2(3):18–22.
[6]L. Breiman. Random forest. Mach. Learn., 4:5–32, 2001.

[7] Rossmann,https://www.kaggle.com/c/rossmann-store-sales.

[8] L.M. Liu, and Z. Kong, Data Mining in Sales Forecasting[J], Business Times, 2007, pp.8-9.

[9] Chu Xu, Ihab F. Ilyas, Sanjay Krishnan and Jiannan Wang, "Data cleaning: Overview and emerging challenges", Proceedings of the 2016 International Conference on Management of Data, pp. 2201-2206, 2016.

[10] Data preparation for neural network data analysis, Stanislav I. Koval, 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus).

14 APPENDICES

DATA :

train.csv :

	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
1017204	1111	2	2013-01-01	0	0	0	0	a	1
1017205	1112	2	2013-01-01	0	0	0	0	a	1
1017206	1113	2	2013-01-01	0	0	0	0	a	1
1017207	1114	2	2013-01-01	0	0	0	0	a	1
1017208	1115	2	2013-01-01	0	0	0	0	a	1

store.csv :

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2SinceYear	PromoInterval
0	1	c	a	1270.0	9.0	2008.0	0	NaN	NaN	NaN
1	2	a	a	570.0	11.0	2007.0	1	13.0	2010.0	Jan,Apr,Jul,Oct
2	3	a	a	14130.0	12.0	2006.0	1	14.0	2011.0	Jan,Apr,Jul,Oct
3	4	c	c	620.0	9.0	2009.0	0	NaN	NaN	NaN
4	5	a	a	29910.0	4.0	2015.0	0	NaN	NaN	NaN
1110	1111	a	a	1900.0	6.0	2014.0	1	31.0	2013.0	Jan,Apr,Jul,Oct
1111	1112	c	c	1880.0	4.0	2006.0	0	NaN	NaN	NaN
1112	1113	a	c	9260.0	NaN	NaN	0	NaN	NaN	NaN
1113	1114	a	c	870.0	NaN	NaN	0	NaN	NaN	NaN
1114	1115	d	c	5350.0	NaN	NaN	1	22.0	2012.0	Mar,Jun,Sept,Dec

CODE :

The full notebook is available on github through the following link:
https://github.com/salonij/RossmannDrugStore

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style("whitegrid")
import sklearn
import warnings
warnings.filterwarnings("ignore")
```

Load Dataset

```
#reading data files
store=pd.read_csv("store.csv")
train=pd.read_csv("train.csv",parse_dates = True, index_col = 'Date')
```

```
store.head().append(store.tail())
```

```
train.head().append(train.tail())
```

Dealing with missing values

```
#Count missing values in store dataset
print("Store Dataset:\n\n",store.isnull().sum())
```

```
#Count missing values in train dataset
print("Train Dataset:\n\n",train.isnull().sum())
```

```
#we can see that some features have a high percentage of missing values and they won't be accurate as indicators,
#so we will remove features with more than 30% missing values.
store = store.drop(['CompetitionOpenSinceMonth', 'CompetitionOpenSinceYear','Promo2SinceWeek',
                    'Promo2SinceYear', 'PromoInterval'], axis=1)
```

```
# CompetitionDistance is distance in meters to the nearest competitor store
# let's first have a look at its distribution
sns.distplot(store.CompetitionDistance.dropna())
plt.title("Distributin of Store Competition Distance")
```

The distribution is right skewed, so we'll replace missing values with the median.

```
# replace missing values in CompetitionDistance with median for the store dataset
store.CompetitionDistance.fillna(store.CompetitionDistance.median(), inplace=True)
```

Data Extraction

```
# extract year, month, day and week of year from "Date"
train["Year"]=train.index.year
train["Month"]=train.index.month
train["Day"]=train.index.day
train["WeekOfYear"]=train.index.week
train = train.reset_index()
```

Join tables

```
df = pd.merge(train, store, how='left', on='Store')
df.head()
```

Drop subsets of data which might cause Bias

```
# where stores are closed, they won't generate sales, so we will remove this part of the dataset
df = df[df.Open != 0]
```

```
# Open isn't a variable anymore, so we'll drop it
df = df.drop('Open', axis=1)
```

```
# see if there's any opened store with zero sales
print("Number of stores with zero sales:",df[df.Sales == 0].shape[0])
```

```
Number of stores with zero sales: 54
```

```
# remove this part of data to avoid bias
df = df[df.Sales != 0]
```

Handling Categorical Data

```
df.info()
```

```
# see what's in nominal varibles
set(df.StateHoliday), set(df.StoreType), set(df.Assortment)
```

```
# StateHoliday indicates a state holiday - a = public holiday, b = Easter holiday, c = Christmas, 0 = None
# convert number 0 to string 0
df.loc[df.StateHoliday == 0,'StateHoliday'] = df.loc[df.StateHoliday == 0,'StateHoliday'].astype(str)
```

```
# 0 - not a state holiday; 1- is on a state holiday
df['StateHoliday'] = df.StateHoliday.map({'0':0, 'a':1 ,'b' : 1,'c': 1})
```

Adding additional features

```
df1 = df.copy()
```

```
# calculate weekly average sales
sales = df1[['Year','Month','Store','Sales']].groupby(['Year','Month','Store']).mean()
sales = sales.rename(columns={'Sales':'AvgSales'})
sales = sales.reset_index()
```

```
df1['sales_key']=df1['Year'].map(str) + df1['Month'].map(str) + df1['Store'].map(str)
sales['sales_key']=sales['Year'].map(str) + sales['Month'].map(str) + sales['Store'].map(str)
```

```
# drop extra columns
sales = sales.drop(['Year','Month','Store'], axis=1)
# merge
df1 = pd.merge(df1, sales, how='left', on=('sales_key'))
```

```
#create a variable that calculates Monthly average number of customers for each store.
cust = df1[['Year','Month','Store','Customers']].groupby(['Year','Month', 'Store']).mean()
cust = cust.rename(columns={'Customers':'AvgCustomer'})
cust = cust.reset_index()
```

```
df1['cust_key']=df1['Year'].map(str) + df1['Month'].map(str) + df1['Store'].map(str)
cust['cust_key']=cust['Year'].map(str) + cust['Month'].map(str) + cust['Store'].map(str)
```

```
# drop original feature Customers
df1 = df1.drop('Customers', axis=1)
```

```
# drop extra columns
cust = cust.drop(['Year', 'Month', 'Store'], axis=1)
```

```
# merge
df1 = pd.merge(df1, cust, how="left", on=('cust_key'))
```

```
# drop extra columns
df1 = df1.drop(['cust_key','sales_key','Store','Date'], axis=1)
```

Data Exploration and Visualization :

```
dfv = df.copy()
```

Sales Distribution

```
{"Mean":np.mean(dfv.Sales),"Median":np.median(dfv.Sales)}
```

```
{'Mean': 6955.959133664481, 'Median': 6369.0}
```

```
plt.figure(figsize=(10,5))
plt.hist(x=dfv.Sales, bins=30,color = "orange")
plt.ylabel('number of observations')
plt.xlabel('daily sales in $')
plt.title('Sales Distribution')
```

Customer Distribution

```
{"Mean":np.mean(dfv.Customers),"Median":np.median(dfv.Customers)}
```

```
{'Mean': 762.777166253325, 'Median': 676.0}
```

```
plt.figure(figsize=(10,5))
plt.hist(x=dfv.Customers , bins=30,color = "pink")
plt.ylabel('number of observations')
plt.xlabel('daily total number of customers')
plt.title('Customer Distribution')
```

Sales Over Weeks

```
plt.figure(figsize=(15,5))
sns.barplot(x=dfv['WeekOfYear'],y=dfv['Sales'],data=dfv)
plt.xlabel('Week Of Year')
plt.ylabel('Total Sales')
plt.title('Sales Over Weeks')
```

Sales Over Time

```
store1_2015 = dfv.query('Store == 1 and Year == 2015')
store1_2013 = dfv.query('Store == 1 and Year == 2013')
store1_2014 = dfv.query('Store == 1 and Year == 2014')
```

```
plt.figure(figsize=(12,5))
sns.lineplot(x=store1_2013.Date, y=store1_2013.Sales, data=store1_2013)
sns.lineplot(x=store1_2014.Date, y=store1_2014.Sales, data=store1_2014)
sns.lineplot(x=store1_2015.Date, y=store1_2015.Sales, data=store1_2015)
plt.title('Sales Over Time')
```

Sales vs. Competition Distance

```
plt.figure(figsize=(10,5))
plt.scatter(x=dfv.CompetitionDistance, y=dfv.Sales , c=dfv.Customers)
plt.ylabel('Sales')
plt.xlabel('Competition Distance')
plt.title('Sales vs. Competition Distance')
cbr= plt.colorbar()
cbr.set_label('Number Of Customers')
```

Impact of promotion on sales over days of a week

```
print ("Number of Stores opened on Sundays:{}" .format(dfv[dfv.DayOfWeek == 7]['Store'].unique().shape[0]))
```

```
Number of Stores opened on Sundays:33
```

```
sns.factorplot(data = dfv, x ="DayOfWeek", y = "Sales",
                hue='Promo'
               ,sharex=False)
```

Sales By Store Type

```
# StoreType - differentiates between 4 different store models: a, b, c, d
plt.figure(figsize=(8,5))
sns.boxplot(x=dfv.StoreType, y=dfv.Sales, data=dfv)
plt.ylabel('Total Sales')
plt.title('Sales By Store Type')
```

Sales By Assortment

```
# Assortment - describes an assortment level: a = basic, b = extra, c = extended
plt.figure(figsize=(8,5))
sns.boxplot(x=dfv.Assortment, y=dfv.Sales, data=dfv)
plt.ylabel('Total Sales')
plt.title('Sales By Assortment')
```

Assortment-wise sales over months with respect to store type.

```
]: sns.factorplot(data = dfv, x = 'Month', y = "Sales",
                  col = 'Assortment',
                  palette = 'plasma',
                  hue = 'StoreType')
```

Correlation HeatMap

```
# Converting categorial features Assortment and StoreType
dfv['Assortment']=dfv['Assortment'].astype('category').cat.codes
dfv['StoreType']=dfv['StoreType'].astype('category').cat.codes

# Adding new feature SalesperCustomer to get better correlation with StoreType and Assortment
dfv['SalesperCustomer']=dfv['Sales']/dfv['Customers']
```

```
corr = dfv.corr()
```

```
mask = np.zeros_like(corr, dtype = np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize = (11, 9))
sns.heatmap(corr, mask = mask, annot= True,
            square = True, linewidths = .5, ax = ax, cmap = "BuPu")
plt.title("Correlation Heatmap", fontsize=20)
```

Store Sales Prediction

```
1 # split features and labels
2 X = df1.drop('Sales', axis=1)
3 y = df1. Sales
```

```
1 # get dummy variables for categorical features for linear regression
2 xd = X.copy()
3 xd = pd.get_dummies(xd)
```

```
1 # split training and test datasets
2 from sklearn.model_selection import train_test_split
3 xd_train,xd_test,yd_train,yd_test = train_test_split(xd,y,test_size=0.3, random_state=1)
```

Linear Regression

```
1 from sklearn.linear_model import LinearRegression
2 lin= LinearRegression()
3 linreg = lin.fit(xd_train, yd_train)
```

```
# definte RMSE function
from sklearn.metrics import mean_squared_error
from math import sqrt

def rmse(x, y):
    return sqrt(mean_squared_error(x, y))

# definte MAPE function
def mape(x, y):
    return np.mean(np.abs((x - y) / x)) * 100

# get cross validation scores
yd_predicted = linreg.predict(xd_train)
yd_test_predicted = linreg.predict(xd_test)

print("Regresion Model Score" , ":" , linreg.score(xd_train, yd_train) , "," ,
      "Out of Sample Test Score" ,":" , linreg.score(xd_test, yd_test))
print("Training RMSE", ":", rmse(yd_train, yd_predicted),
      "Testing RMSE", ":", rmse(yd_test, yd_test_predicted))
print("Training MAPE", ":", mape(yd_train, yd_predicted),
      "Testing MAPE", ":", mape(yd_test, yd_test_predicted))

regression_train = linreg.score(xd_train, yd_train)
regression_test = linreg.score(xd_test, yd_test)
```

Random Forest Regression

```
# label nominal variables for tree based regression
xl = X.copy()

from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
xl.StateHoliday = label.fit_transform(xl.StateHoliday)
xl.Assortment = label.fit_transform(xl.Assortment)
xl.StoreType = label.fit_transform(xl.StoreType)
```

```
xl_train,xl_test,yl_train,yl_test = train_test_split(xl,y,test_size=0.3, random_state=1)
```

```
from sklearn.ensemble import RandomForestRegressor
rdf = RandomForestRegressor(n_estimators=30)
rdfreg = rdf.fit(xl_train, yl_train)

```

```
#There are multiple features present in the dataset
#This DataFrame will help us to obtain the importance of each features.

feature_importances = pd.DataFrame(rdf.feature_importances_,
                                   index = xl_train.columns,
                                    columns=['importance']).sort_values('importance', ascending=False)
```

```
feature_importances
```

```
print("Regresion Model Score" , ":" , rdfreg.score(xl_train, yl_train) , "," ,
      "Out of Sample Test Score" ,":" , rdfreg.score(xl_test, yl_test))

yl_predicted = rdfreg.predict(xl_train)
yl_test_predicted = rdfreg.predict(xl_test)

print("Training RMSE", ":", rmse(yl_train, yl_predicted),
      "Testing RMSE", ":", rmse(yl_test, yl_test_predicted))
print("Training MAPE", ":", mape(yl_train, yl_predicted),
      "Testing MAPE", ":", mape(yl_test, yl_test_predicted))

Forest_train = rdfreg.score(xl_train, yl_train)
Forest_test = rdfreg.score(xl_test, yl_test)
```

KNN Regression

```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors = 30)
knnreg = knn.fit(xd_train, yd_train)
```

```
print("Regresion Model Score" , ":" , knnreg.score(xd_train, yd_train) , "," ,
      "Out of Sample Test Score" ,":" , knnreg.score(xd_test, yd_test))

yd_predicted = knnreg.predict(xd_train)
yd_test_predicted = knnreg.predict(xd_test)

print("Training RMSE", ":", rmse(yd_train, yd_predicted),
      "Testing RMSE", ":", rmse(yd_test, yd_test_predicted))
print("Training MAPE", ":", mape(yd_train, yd_predicted),
      "Testing MAPE", ":", mape(yd_test, yd_test_predicted))

Knn_train = knnreg.score(xd_train, yd_train)
Knn_test =  knnreg.score(xd_test, yd_test)
```

```
for x in range(1,30):
    knn = KNeighborsRegressor(n_neighbors = x)
    knnreg = knn.fit(xd_train, yd_train)
    print("Regresion Model Score" , ":" , knnreg.score(xd_train, yd_train) , "," ,
      "Out of Sample Test Score" ,":" , knnreg.score(xd_test, yd_test))
```

Comparison between Models

```
#Accuracy Score Comparison between Linear Regression, Random Forest and KNN

import pandas as pd

train_error=[regression_train,Forest_train,Knn_train]

test_error=[regression_test,Forest_test,Knn_test]

col={'Train Error':train_error,'Test Error':test_error}
models=['Linear Regression','RDF','Knn']
dfm=pd.DataFrame(data=col,index=models)
dfm
```

```
#Plotting the table using Bar Plot
dfm.plot(kind='bar')
```