Sales Forecasting (Retail Sector)

Objective: To forecast the sale of a store and optimize it wrt the market demand. Since store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality and many more.

Goal:

- Explore the data (handle missing values etc).
- Analysis per store type and correlational analysis of stores' activity.
- To define the actual relationship between variables and their associated demand patterns.
- Perform Time Series Analysis (MA, ARIMA, SARIMAX),Machine Learning(Lasso Regression and Random Forest)and Deep learning(Feedforward Neural Network with backpropagation) models.

Dataset: We selected two datasets Walmart Store Sales and Rossmann Store Sales after studying both we decided to finalize the ROSSMANN as the Walmart dataset doesn't contain many features and has a lot of missing values.

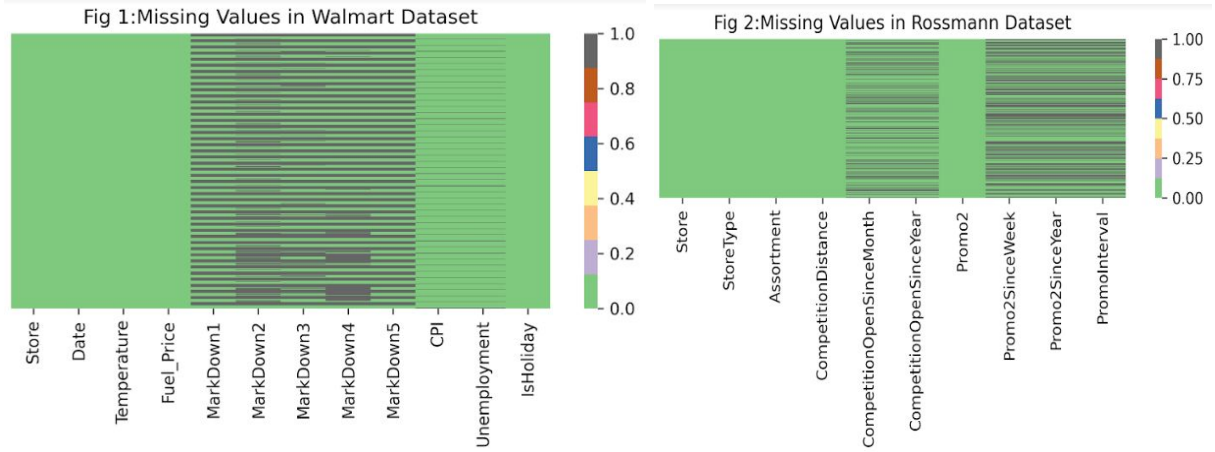
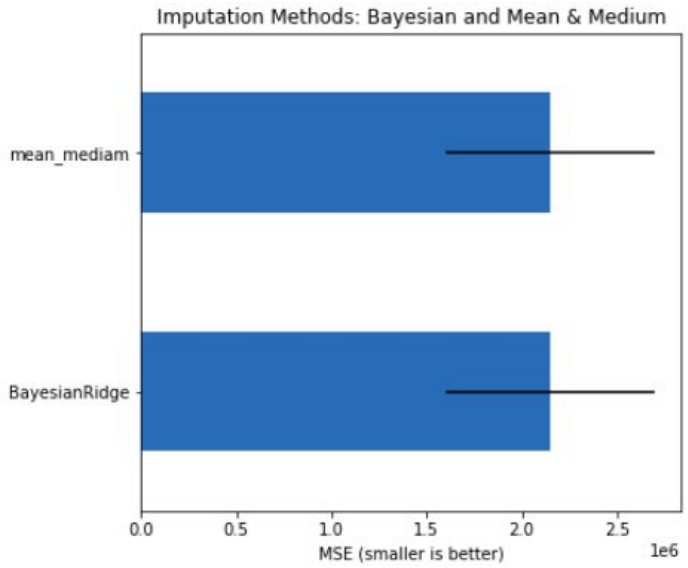


Fig 1 show the missing values in Walmart Dataset and Fig 2 shows missing values in Rossmann Dataset

In ROSSMANN dataset missing values are handled using simple imputers i.e mean & median and also compared with iterative imputers (Bayesian estimator) ,implying the

results are the same(as shown below).



Rossmann Dataset Overview:

	Dataset	Variables	No.of Variables	No. of rows
1	train.csv	Store, DayOfWeek, Date, Sales, Customers, Open, Promo,StateHoliday,SchoolHoliday	9	1017209
2	test.csv	Id,Store,DayOfWeek,Date,Open,Promo, StateHoliday,SchoolHoliday	8	41088
3	store.csv	Store,StoreType,Assortment,CompetitionDistance,CompetitionOpenSinceMonth,CompetitionOpenSinceYear,Promo2,Promo2SinceWeek,Promo2SinceYear,PromoInterval	10	1115

Feature Engineering Intro:

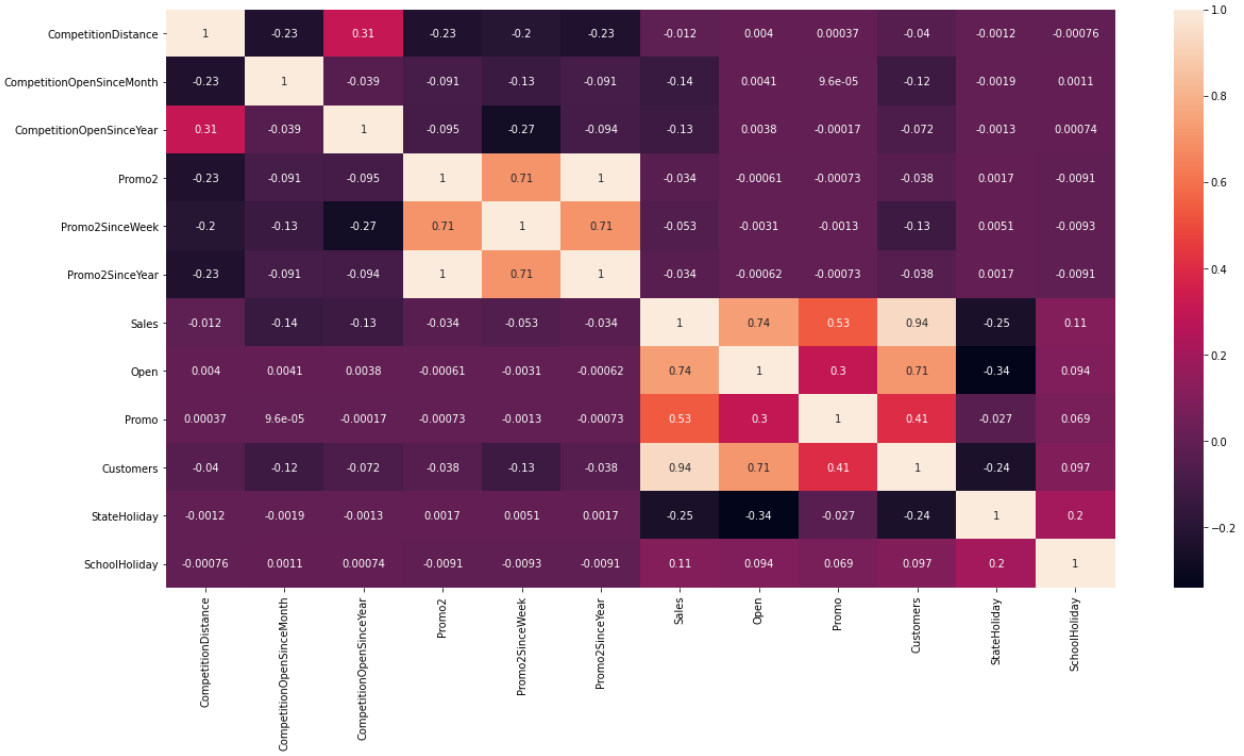
Step 1: Firstly we analysis the data and detect missing values and handle them with appropriate methods.

Step 2:

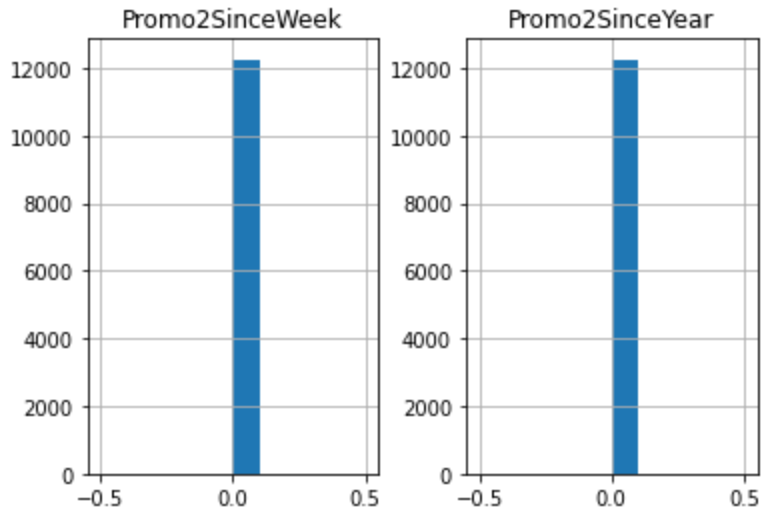
- Filled missing values with proper method:
    - Removed missing observations in train.csv when Open=0 because when the store is closed there is zero number of sales.
    - Handled missing observations in store.csv with different methods like :
        - Simple Computer with strategy mean for column ‘ComeptitionDistance’.
        - Simple Imputer with strategy median for column ‘CompetitionOpenSinceMonth’ and ‘CompetitionOpenSinceYear’.
        - And for column ‘Promo2SinceWeek’, ‘Promo2SinceYear’ and ‘Promo2Interval’ we filled them with 0 as ‘Promo2’ has 0 value.
- Split data by Stores to predict sales of individual stores.
- Split data by Dates, Monthly and Year to check seasonality. (For Time Series model)

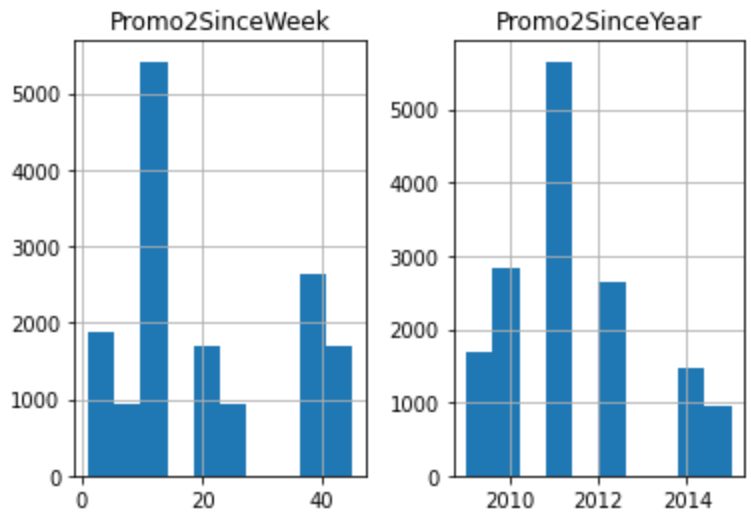
Step 3: Then the data with no null values are taken for encoding for which we first change the object data type to the relevant one and then applied different feature engineering methods like label encoding and one hot encoding method to increase the number of features so total 23 new features are created.

Step 4: We also did some visualization techniques to visualize what we are doing like using a heat map. We checked the correlation between the columns which was helpful for checking the relation between columns.



Step 5: Data also contains some values which were dependent on other columns so we visualize that too and in the same way we checked for particular year and its sale.





Models Implemented:

- Time Series Models: AR, MA and ARIMA
- Machine learning models: Lasso Regression and Random Forest
- Deep Learning model: Deep Neural network (Feedforward Neural Network with backpropagation)

Evaluation Matrix:

After applying different models we have to check the accuracy of the model and for that we are using RMSE and adjusted R^2 score.

R2 score will be used for Time Series models because Adjusted R2 will help in evaluation when there are more no. of features.

Data Pre-Processing:

1)Label Encoder/One Hot Vector: In our dataset, there are categorical variables and to apply the ML models, we need to transform these categorical variables into numerical variables. And this problem is solved in the feature engineering section.

2)Normalization:
There are two columns i.e CompetitionDistance, Customers whose values are not in the standard range with comparison to the overall dataset.

In this project, these two features are normalized using the MinMaxScaler method (sklearn library) instead of standardization (StandardScaler) because, on standardization, most features values are negative that impact badly on the model.

3)Feature Selection:

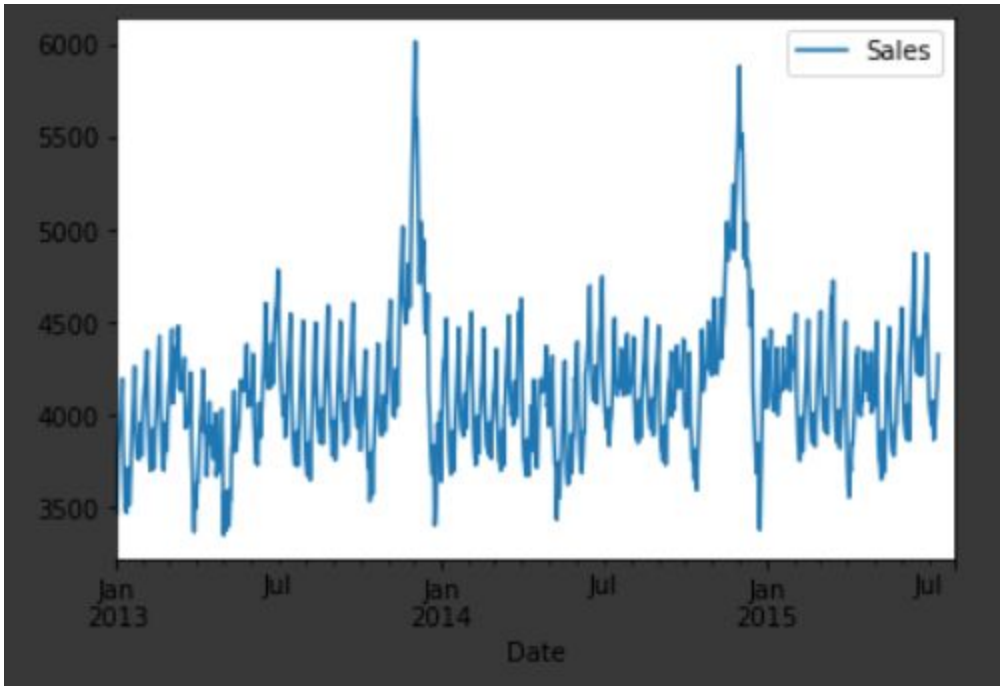
During Feature Engineering, lots of new features are created but all features are not required or all features are not important.

The Sklearn library SelectFromModel is used for feature selection and base estimator is LassoRegression which will return 20 features which have more of an impact on the target variable.

4)Split Dataset: In this process, 80% of the data was split for the train data and 20% of the data was taken as test data.

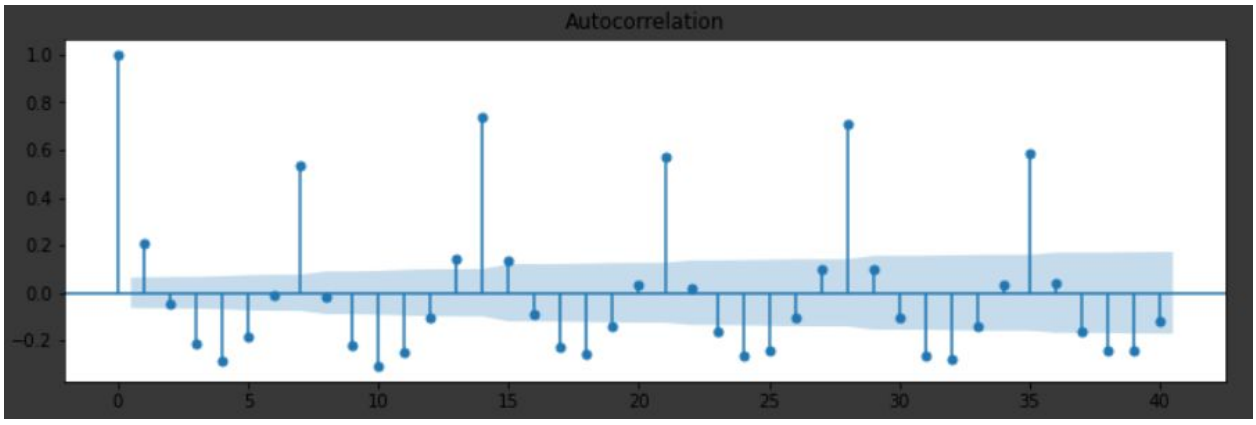
Time Series Analysis

A moving average is a calculation used to analyze data points by creating a series of averages of different subsets of the full data set.



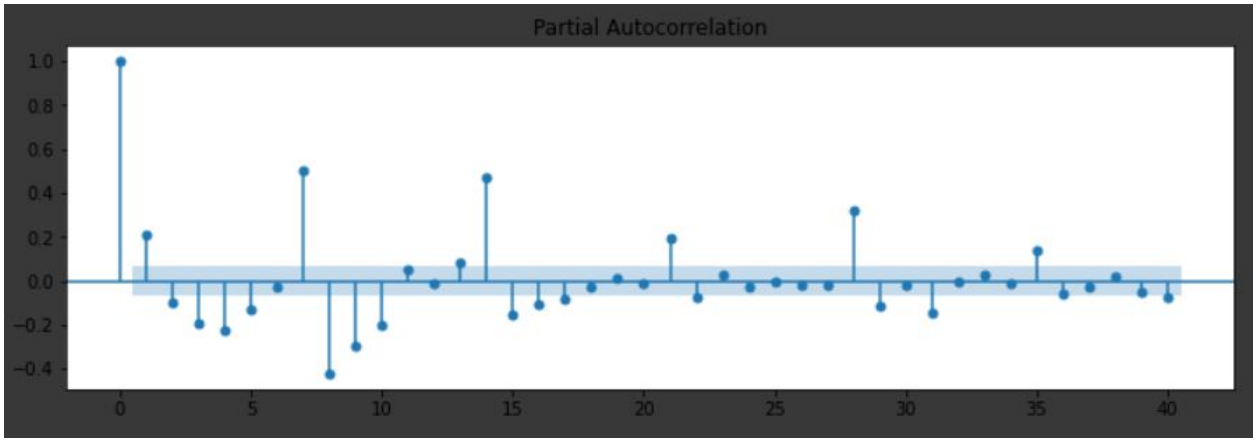
ACF and PACF Plots

ACF is an (complete) auto-correlation function which gives us values of auto-correlation of any series with its lagged values.



PACF is a partial auto-correlation function that finds correlation of the residuals (which remains after removing the effects which are already explained by the earlier lag(s)) with the next lag

value hence ‘partial’ and not ‘complete’ as we remove already found variations before we find the next correlation. So if there is any hidden information in the residual which can be modeled by the next lag, we might get a good correlation and we will keep that next lag as a feature while modeling. Remember while modeling we don’t want to keep too many features which are correlated as that can create multicollinearity issues. Hence we need to retain only the relevant features.



Time series algorithms are used extensively for analyzing and forecasting time-based data. ARIMA and SARIMAX class of models which are based on describing autocorrelations in the data are used.

a) ARIMA (Autoregressive Integrated Moving Average)

Three components p, d and q are required to build the ARIMA model:-

p - Number of autoregressive lags

d - Order of differencing required to make the series stationary

q - Number of moving average lags

One of the requirements for ARIMA is that the time series should be stationary. A stationary series is one where the properties do not change over time. We used the Augmented Dickey-Fuller test.

The Augmented Dickey-Fuller test is a type of statistical unit root test. The test uses an autoregressive model and optimizes an information criterion across multiple different lag values.

The null hypothesis of the test is that the time series is not stationary, while the alternate hypothesis (rejecting the null hypothesis) is that the time series is stationary.

```
ADF Test Statistic : -6.218236994150324
p-value : 5.300862570845989e-08
#Lags Used : 21
Number of Observations Used : 920
Critical Values:
1% -3.4374778690219956
5% -2.864686684217556
10% -2.5684454926748583
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit root and is stationary
```

The p-value now is below the significance level, indicating that the series is stationary.

ARIMA Model Results

Dep. Variable:	D.Sales	**No. Observations:**	705
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-6470.442
Method:	css-mle	**S.D. of innovations**	2331.759
Date:	Wed, 30 Sep 2020	**AIC**	12948.885
Time:	17:27:50	**BIC**	12967.117
Sample:	1	**HQIC**	12955.930

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.3131	0.436	-0.718	0.473	-1.168	0.542
ar.L1.D.Sales	0.0127	0.038	0.336	0.737	-0.061	0.087
ma.L1.D.Sales	-1.0000	0.004	-259.522	0.000	-1.008	-0.992

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	78.9383	+0.0000j	78.9383	0.0000
MA.1	1.0000	+0.0000j	1.0000	0.0000

b) SARIMAX (Seasonal Autoregressive Integrated Moving Average Exogenous)

A Seasonal ARIMAX model is formed by including additional seasonal terms in the ARIMAX models.

SARIMAX Results

Dep. Variable:	Sales	**No. Observations:**	942
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 12)	**Log Likelihood**	-8541.663
Date:	Wed, 30 Sep 2020	**AIC**	17093.327
Time:	17:27:55	**BIC**	17117.498
Sample:	0	**HQIC**	17102.546
	- 942		

Covariance Type: opg

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.2014	0.032	6.352	0.000	0.139	0.264
ma.L1	-1.0000	0.717	-1.396	0.163	-2.404	0.404
ar.S.L12	-0.0826	0.042	-1.980	0.048	-0.164	-0.001
ma.S.L12	-0.9991	0.720	-1.388	0.165	-2.410	0.412
sigma2	5.329e+06	1.35e-07	3.96e+13	0.000	5.33e+06	5.33e+06

Ljung-Box (L1) (Q):	0.11	**Jarque-Bera (JB):**	80.19
Prob(Q):	0.74	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.85	**Skew:**	-0.72
Prob(H) (two-sided):	0.16	**Kurtosis:**	3.06

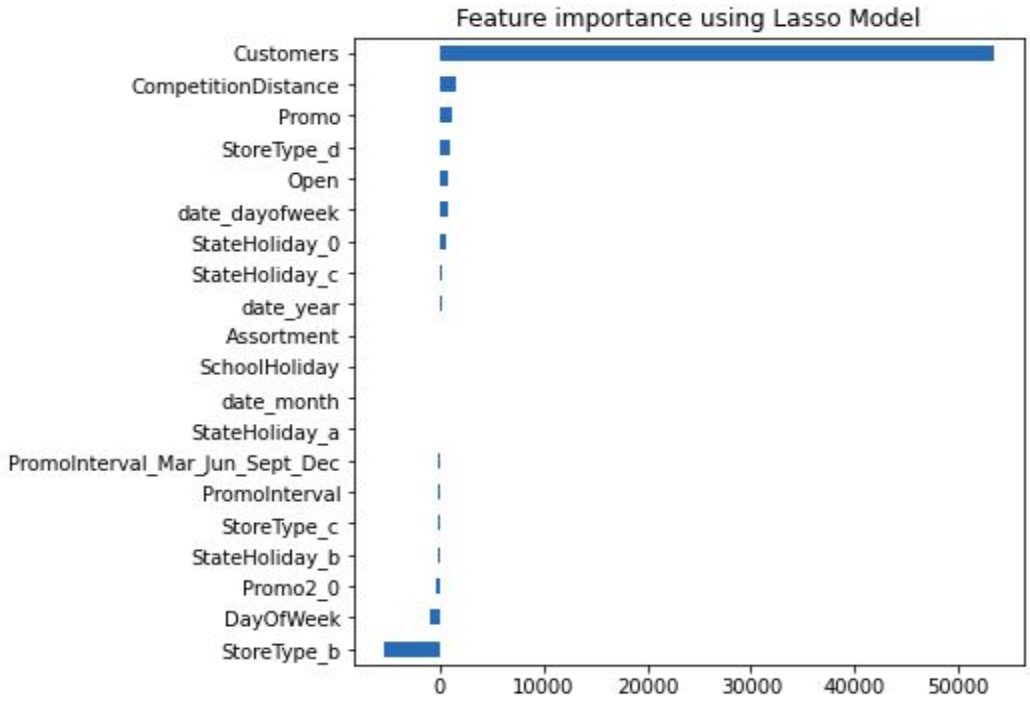
Machine Learning Models:

1) Lasso Regression: LASSO stands for Least Absolute Shrinkage and Selection Operator.

Lasso regression is one of the regularization methods that create parsimonious models in the presence of a large number of features, where large means either of the below two things:

1. Large enough to enhance the tendency of the model to over-fit. Minimum ten variables can cause overfitting.
2. Large enough to cause computational challenges. This situation can arise in the case of millions or billions of features

Feature Importance Graph:



Result:

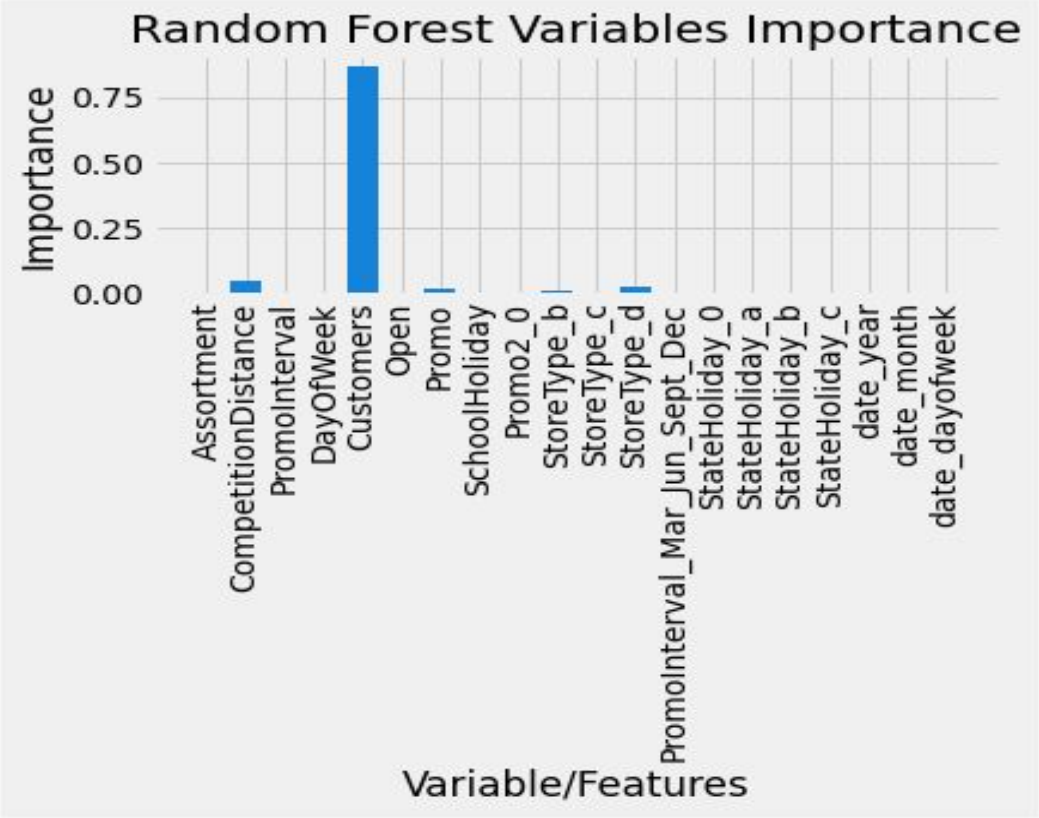
Adj. R2 Score (Training)	0.8955 or 89.55%
Adj. R2 Score (Testing)	0.8967 or 89.67%

2) Random Forest Regressor:

The random forest regressor consists of many decision trees as a base estimator. It uses bagging and features randomness when building each tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

In our model, 150 decisions are created with max_depth 20.

Feature Importance graph:



Result:

Adj. R2 Score (Training)	0.9921 or 99.21%
Adj. R2 Score (Testing)	0.9812 or 98.12%

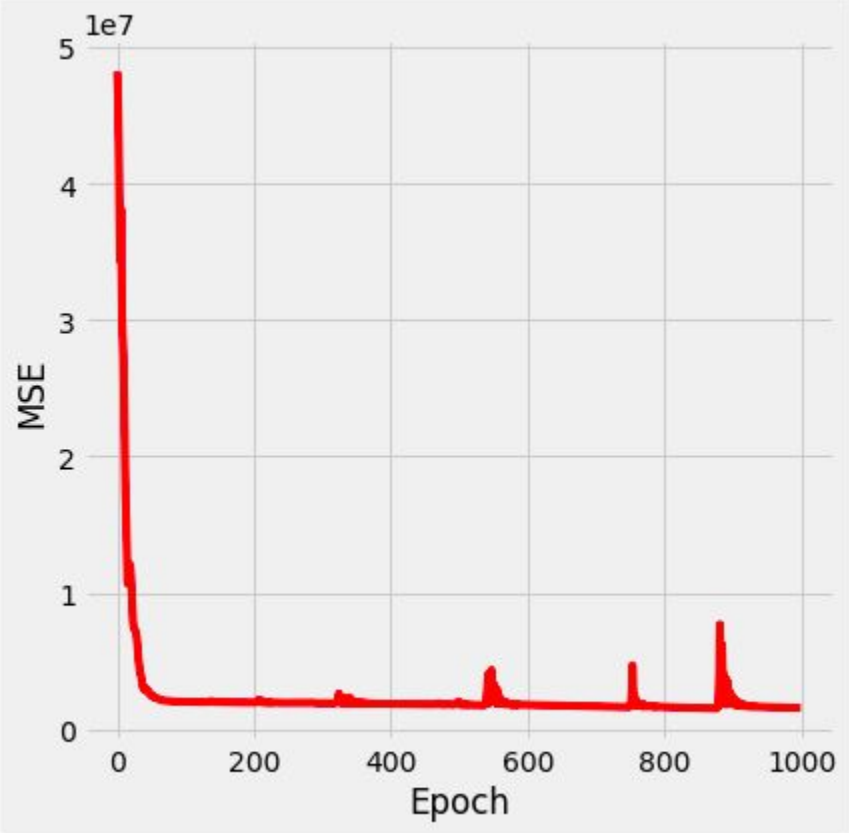
3) Deep Neural Network:

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship.

In our DNN, There is one input layer than 4 hidden layers and each layer consist of 100 neurons. In the end, there is one linear layer that gives the final output.

Optimization algorithm: Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. Adam is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks.

Loss Function: Mean Squared Error (MSE) loss function is used for this DNN.
Loss Graph per epoch (max 1000): DNN is training 1000 times i.e 1000 epoch and results shown in the below graph.



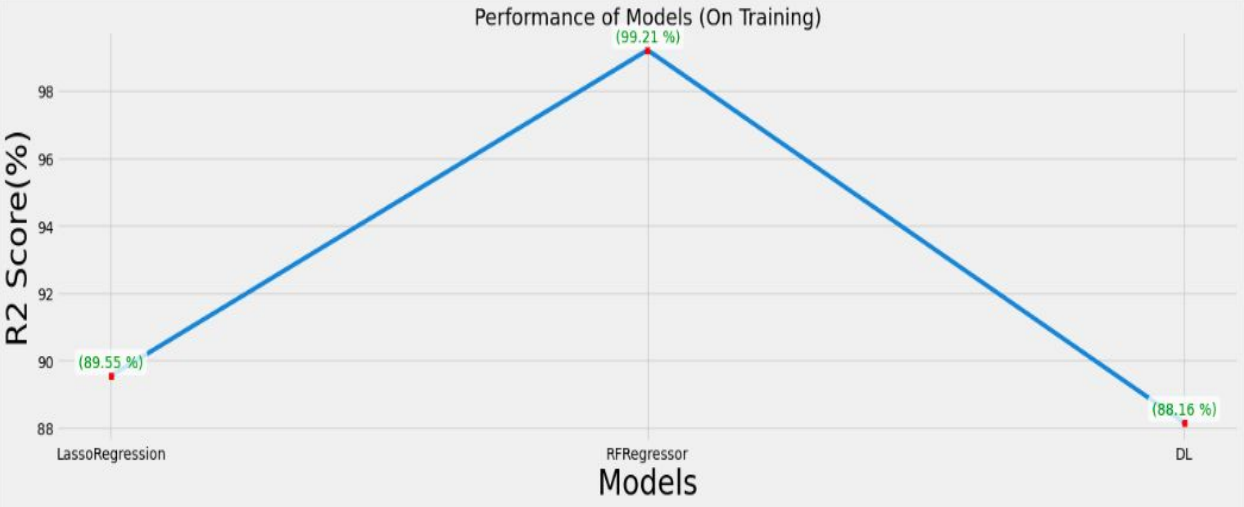
Result:

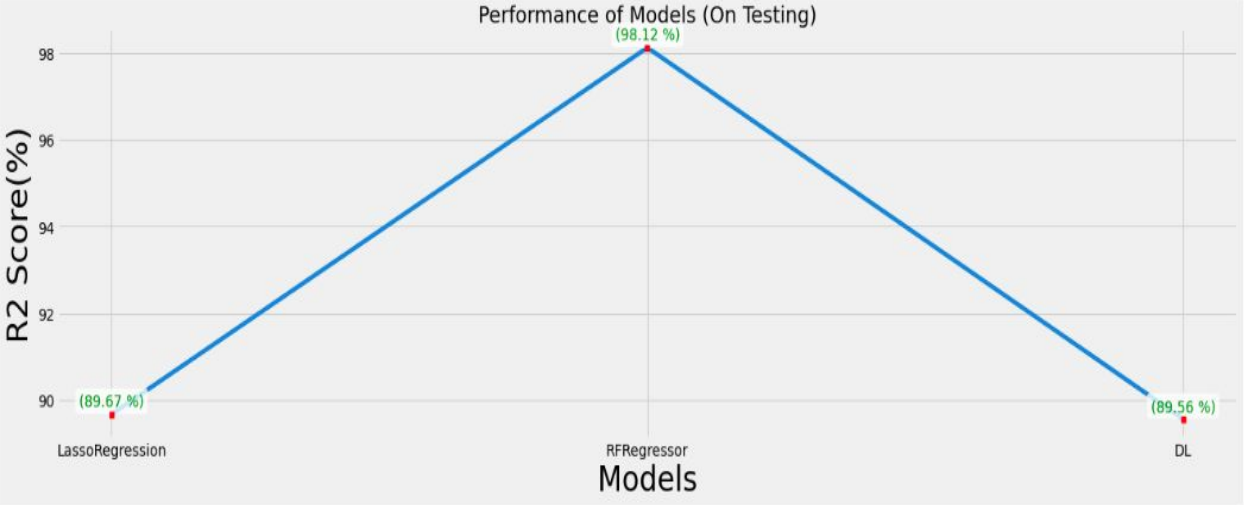
Adj. R2 Score (Training)	0.8816 or 88.16%
Adj. R2 Score (Testing)	0.8956 or 89.56%

Over-All Evaluation:

The below graph shows that Random Forest Regressor performs better as compared to other models.

For DNN, we have done hyperparameter tuning manually, but these are not the optimal values due to the hardware limitations we have not to reach.





Contributors:

[Akhitha Babu]
[Sajal Sharma]
[Panwar Abhash Anil]