Rossmann Drug Store Sales Prediction

Machine Learning Project Report

Vishesh Ruparelia
IMT2016006

Sarthak Soni
IMT2016089

Shivam Kumar Singh
IMT2016114

***Abstract*—Our project attempted to apply various machine learning techniques to a real-world problem of predicting drug store sales. Rossmann, Germany's second-largest drug store chain, has provided past sales information of 1115 Rossmann stores located across Germany. We preprocessed, feature-engineered the data, and examined different statistical / machine learning analysis for forecasting sales of each store. Then, we compared the methods' predictive power by computing Root Mean Square Percentage Error (RMSPE). We found that Gradient Boosting model performed the best with a RMSPE score of 0.06 on the test data set. Rossmann Store Sales**

I. Introduction

The objective of this project is to predict 6 weeks of daily sales for 1115 Rossmann stores located across Germany using the data which was provided by Rossmann through Kaggle. The motivation of this project is the following: by reliably predicting daily sales, store managers may decrease operational costs, and create effective staff schedules (ex. by assigning more staffs during busy days). Also, we would like to identify which type of techniques are both efficient and effective in a real-world sales prediction task.

II. Data

Data sets are given by Rossmann through Kaggle. There are three files: "test.csv", "train.csv", and "store.csv". "train.csv" contains 1001599 observed daily sales of different stores from 2013 to 2015. "store.csv" contains supplementary information for each store (1115 lines because there are 1115 stores). "train.csv" and "store.csv" both contain Store id that we can use to join the train and store data sets. Finally, "test.csv" has 1115 lines of daily sales of different stores but the value of Sales column is omitted. We are expected to predict the value of Sales column for the test set ("test.csv") by using the training set ("train.csv") and store data ("store.csv"). The following table describes fields in "test.csv" and "training.csv":

Store	**unique id for each store (positive integer ranging from 1 to 1115)**
DayOfweek	**the day of week (positive integer ranging from 1 to 7 where 1, 2, ..., 7 corresponding to Monday, Tuesday, ...,Sunday)**
Date	**the date (string of format YYYYMMDD)**
Sales	**the total amount of revenue for this given day**
Customers	**the number of customers on the given day**
Open	**an indicator of whether the store was open on the given day (0 if closed, 1 if open)**
Promo	**an indicator of whether the store was running a promotion on the given day**
StateHoliday	**an indicator of whether the given day was a state holiday (0 if not a state holiday, a if public holiday, b if eastern holiday, c if christmas)**
SchoolHoliday	**an indicator of whether the store was affected by a nearby school holiday on the given day (0 if not affected, 1 if affected)**

A. Preprocessing

- We first merged "train.csv" and "store.csv" by "Store" because we can predict daily sales better with more data related to the sales. We also merged "test.csv" and "store.csv" by "Store".
- We splitted "Date" field into three fields: "year", "month", and "day" to better account for the effects each components of date have.

- We computed average sales for each store to create a new field "Average.Sales". This variable seemed to be an indicator of how well a store will perform in future. This makes sense because a store with a strong past performance is likely to perform well in future.

B. Exploratory Data Analysis

To check any distinctive relationships between variables before applications of prediction models, we drew a correlation coefficient plot.

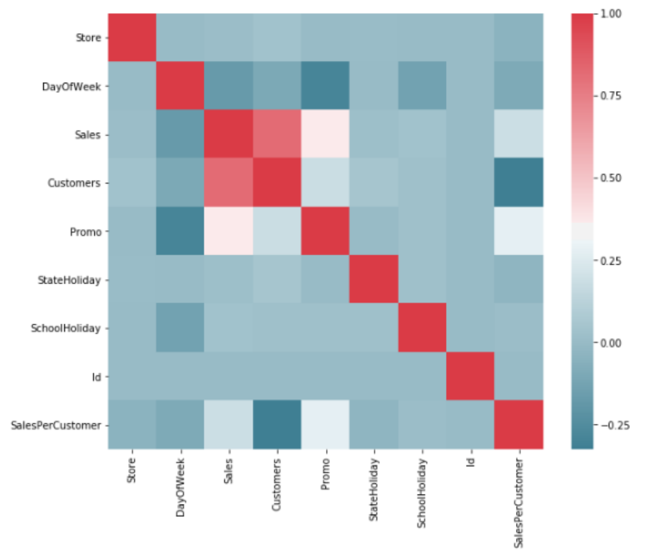

Fig. 1. Correlation Matrix

Analysis of the columns mainly revolved around getting its relationship with 'Sales'.

- "DayOfWeek" and "Sales" are negatively correlated. This implies that, as **"DayOfWeek" approaches to Sunday, sales would decrease** because drugstores would mostly close on Sunday.

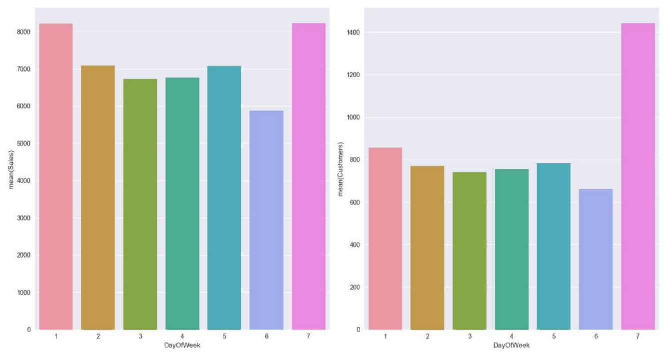

Fig. 2. Average Sales and Average Customers for Open Stores by Day of Week

- As can be seen in the chart above, there are **three peaks** in the average sales and average no. of customers in the week. The first two are on Mondays and Fridays, which is probably due to these days being the start and end of the work week. The highest peak, however, is on Sundays, which is probably due to the fact that most other stores are closed. An interesting observation here is that there is a larger difference between the average no. of customers and average sales on Sundays as compared to other days of the week. While the average sales is approximately the same on Sundays and Mondays, there is a drastic difference in the no. of customers, which hints at a large number of **window shoppers on Sundays**.

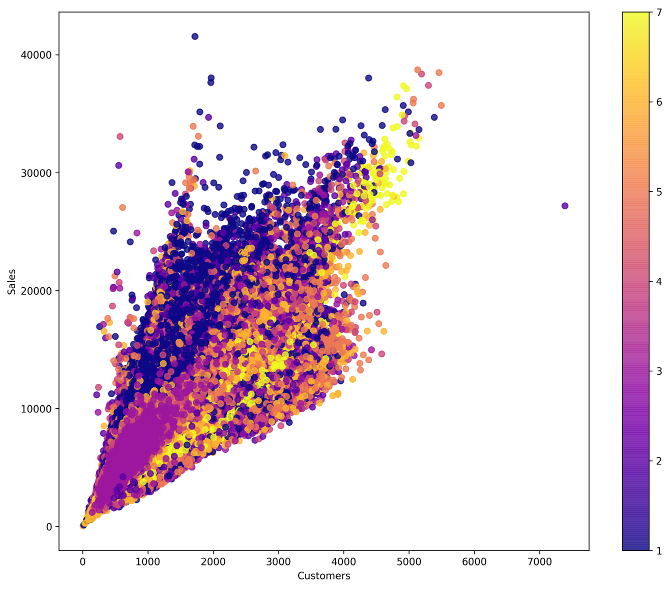

Fig. 3. Correlation Matrix

- From above we see that there are various gradients for sales on different days.This graph helped us realize why **line of best fit** will not capture the dependence between sales and customers.Our assumptions got stronger with further graphs.

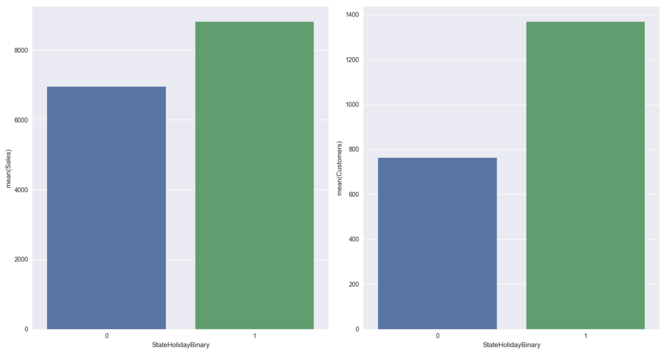

Fig. 4. Mean customers vs State Holiday

- Since most stores are closed on public holidays and closed stores can be ignored when making predictions, the average sales figures and average number of customers on state holidays is plotted, but only for open stores in the figure below. It is evident that when a **store is open on state holidays, it attracts more customers**.

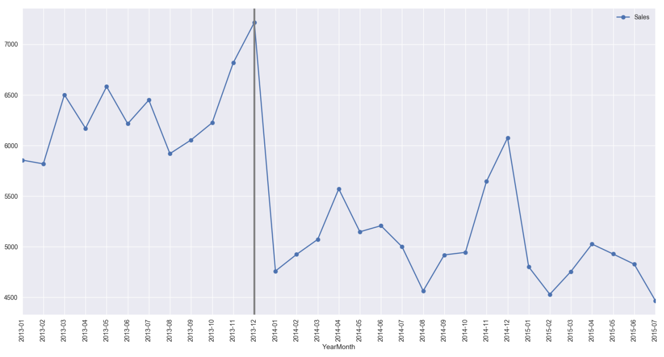

Fig. 5. Mean customers vs State Holiday

- In order to verify the **effect of competition on store sales**, a plot was generated selecting one particular store (Store 6) and studying its sales before and after competition opened nearby. The results are given in the plot above. The effect is immediately evident, with **sales plunging right as the competition opens.**

Our exploratory data analysis notebook consists of analysis of more features.

III. Evaluation Metrics

As the metrics that Kaggle uses to compute test error is the **Root Mean Squared Percentage Error** (RMSPE), we used the same metrics to compute our validation error. It's the following: $RMSPE = \sqrt{\frac{1}{n}\frac{\sum(y_i - \hat{y}_i)^2}{y_i^2}}$ where y_i is a store's sales on a single day, $\hat{y}_i$ is the daily sales prediction for the store on the day, and n is the

IV. Models

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.

A. Linear Regression

This model is a rudimentary first look into the large scale trends. We did not expect it to capture the granular movements of the sales numbers(as seen in Fig. 6) and indeed it didn't, reporting an average **RMSPE of 30.2%**. We observe that there are minimal inter-year trends and therefore can safely disregard them in future considerations.

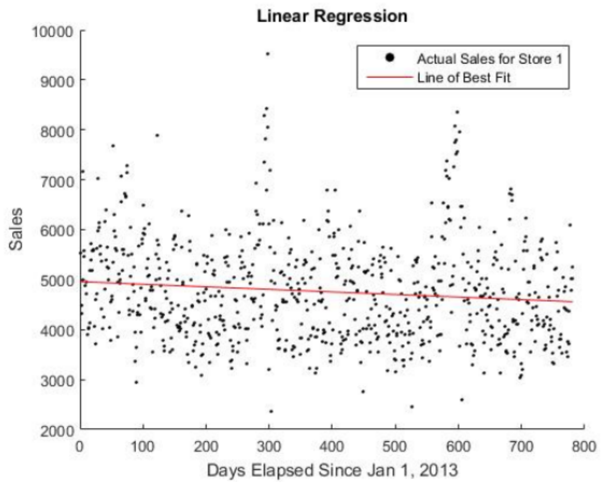

Fig. 6.

B. Mean and Median submission

Predict sales as a historical **mean** for a given store, day of week, and promo.Using values of previous weeks finding out mean for each of the days given whether the store is running a promo on that day or not.Got an **RMPSE of 0.17881**. Followed a similar procedure for **median** and got an **RMPSE of 0.19862**.

C. Random Forest

Random Forest is trained on the data and the performance of the algorithm is recorded by varying the number of trees used. Cross validation error is used to record the performance.

- Its advantages:
  Random forest run times are quite fast, and they are able to deal with unbalanced and missing data. The process of averaging or combining the results of different decision trees helps to overcome the problem of over-fitting. They also do not require preparation of the input data. You do not have to scale the data.
- Its Disadvantages:
  The main drawback of Random Forests is the model size. You could easily end up with a forest that takes hundreds of megabytes of memory and is slow to evaluate. They get a bit harder to interpret than regular decision trees, since we are constructing forest of more than 50 decision trees.

As evident from the results (Fig. 7), the performance of the algorithm **saturates after the number of trees is around 100** which is used to calculate the error on the test data on the Kaggle Competition. Random Forests are biased in favor of categorical variables with different number of levels while calculating the feature importance. One-hot encoding described in Section Data Pre-processing helped us overcome this problem. Random forest was performed with **log transformation on Sales** to not be as sensitive to high sales. The feature importance as ranked by the Random Forest is as shown in Fig. 8

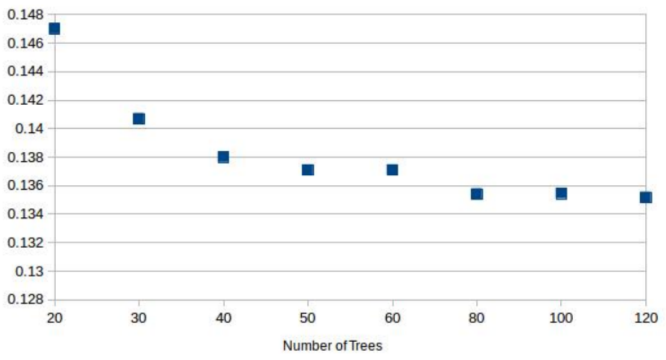

Fig. 7.

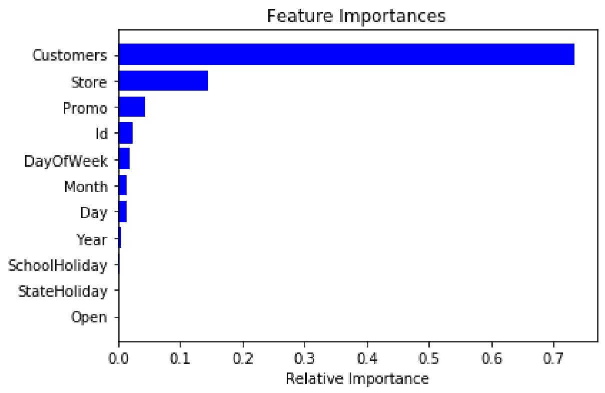

Fig. 8. Feature Importance

D. XgBoost

The main component of XGBoost is its tree ensembles which sum up the predictions of multiple trees. The boosting procedure in XGBoost is an additive process where a trained model is added to the prediction at each step. In our implementation, we used the **RMPSE as the loss function for optimization**. The similarity between Random Forest and Gradient Boosting is that they both generate decision trees. However, Random forest uses bootstrapping method for training while Gradient Boosting builds decision tree over the residuals. In other words, Gradient Boosting Tree generates the decision tree as it optimizes on a proper loss function. This refers that **we can establish our own loss function for optimization, which we couldn't do in Random Forest**.
We set our loss function to be our error metric, RMPSE. Our decision trees here are regression trees. Each regression tree divides the data point at each non-leaf node according to a constraint on a feature. The leaf node value of a data point decides its score, and a data point is predicted by averaging the scores of the leaf node it belongs to. We ran a **lot of versions of XgBoost** and here the observations are as follows:

- The store's opening/closing dates does not affect the store's performance. For example, a **store that was closed yesterday will not get more sales today because of that**.
- DayOfMonth has an effect on sales, for example, the **sales is higher on paydays**.
- Our accuracy improved after we added 2 columns that are **AvgSalesPerCustomer and MedSalesPerCustomer These columns were added to capture the willingness of customers to buy once they entered a store.**
- We determined the **performance of a store via profits** i.e. higher the profits, better the performance. Profits are directly related to sales, therefore more sales will result in more profits. What are sales dependent on? Customers. Therefore, **logically more customers visiting a store should result in more sales. But the real world does not work that way. Lots of customers just browse or window-shop and not contribute to sales. Therefore, we concluded that, the best performance measure for a store should be sales per customer or ratio of sales to customer**. For example, if 10 customers spend a total of Rs. 10, then sale to customers ratio is Rs. 1 per customer. If you increase this ratio, then the store's profit will increase as a result and vice versa. Therefore, if you somehow manage to increase from Rs. 1 per customer to Rs. 2 per customer, or increase the customer numbers at the Rs. 1 per customer, your store's profitability also increases.

E. Stacking XgBoost Models

After building several models with the XGBoost library, we obtained a very good RMSPE score but it had plateaued. At this point, we turned to the **ensemble learning approach as suggested by Prof. GSR to further improve the RMSPE score**. This time, we attempted a custom built static combiner that **combines the predictions of two XGBoost models using a weighted average** (i.e. y pred = y pred1 * w1 + y pred2 * w2). Once we obtained the correct w and correction factor values using the local RMSPE score for the last six weeks of the training data, we tweaked the values manually to get the best possible score on Kaggle as converging the values for w and correction score using the training data for several iterations would be too computationally expensive and would take days on our personal laptops. After tweaking the values we saw the most accurate model was the one which had only last 6 weeks of data. We realized the data for the last 6 weeks captured the sales trends much better than the complete 3 year data.
In practice, **it might be better to use a backward sliding window of 6 weeks** from the last day of the training set to create different test sets for validation. This will provide more scope for the true weights to converge and prevent overfitting. However, **this process involved re-training the two XGBoost models that are being combined for every 6 week test period** and then converging the values of w and correction factor over several iterations, which would

take too long on the **computation power of our personal laptops as informed to Prof. GSR**.

V. 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 analyzing 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. We learned the **benefits of boosted trees and their applications in machine learning**. However, complicated ensemble learning approaches may not be practically applicable in many scenarios as it tends to over-fit the training data.
In the future, we want to implement the backward sliding window of six weeks to converged to the values of the weights and correction factor in our ensemble of boosted trees as opposed to the current single validation set as suggested by Prof. GSR. Since the data is based on time, simply using random sub-sampling will not be the best approach as future data needs to be predicted from historical data and not just a random selection of training instances. We also look to improve the existing models, experimenting with different sets of features.
We were ranked **3rd on Public leader board and 5th on Private leader board**.

Our Final scores for different model submissions are as follows-

Models	**Private Score**	**Public Score**
Mean	0.17881	0.18774
Median	0.19862	0.20667
Random Forest	0.26853	0.30427
XgBoost	0.10753	0.13376
Weighted XgBoost	0.06347	0.06257