Analysis and Forecasting of Rossman Store Sales

Ethan Otto
University of Rochester
Rochester, New York, USA

Tommy Hanson
University of Rochester
Rochester, New York, USA

ABSTRACT

This paper describes the analysis of the application of the Extreme Gradient Descent Decision Tree Boosting algorithm to the time series data featured in the Rossmann Store Sales competition on Kaggle.com. We use XGBoost to both help determine feature importance as well as build a predictive model for store sales up to six (6) weeks into the future. Our best model was able to make it to the 80th percentile on the private leaderboard. This project was done as part of the undergraduate course, CSC 240 taken at University of Rochester during Spring 2019 with Professor Thaddeus Pawlicki PhD.

CCS CONCEPTS

• **Computing Methodologies → Machine Learning**; *Modeling and Simulation*; Artificial Intelligence.

KEYWORDS

Datasets, Time Series Analysis, Decision Trees, Model Prediction, XGBoost

ACM Reference Format:
Ethan Otto and Tommy Hanson. 2019. Analysis and Forecasting of Rossman Store Sales. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied. Because of the importance of overall sales, these predictions should also be able to describe what factors were most impactful. Using XGBoost we can not only build our predictive model, but also explain the effect of different features on the model by rating their feature importance. Additionally, prior to building the model we did some preliminary analysis to look for trends over time and the effects of the two promotions included in the data.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 DATA EXPLORATION AND ANALYSIS

2.1 FEATURE IMPORTANCE

Decision trees have a built in method for evaluating the importance of the features they're trained on. The decision tree can effectively bypass the nodes that use a certain feature. Then it can calculate the difference in gain between the model with and without that feature. A surprising finding was that the most important feature is the competition distance rather then the store id. This implies that a new competitor opens up or goes out of business, the stores sales change drastically. So much so that knowing the store's history is not as valuable as knowing the competitor info.

Feature	Gain
Competition Distance	.2
Store	.175
Promo	.17
Day of Week	.074
Competition Open Since Year	.06
Competition Open Since Month	.055
Store Type	.05
Day	.049

Table 1: Feature Importances for xgboost Baseline Model. This provides us an estimate of how much information gain each feature provides the model.

2.2 Trends and Seasonality

We wanted to break down the trends and seasonality of our dataset. Facebook offers an open source time series analysis algorithm Facebook Prophet prophet. It's uses an additive regression model to find trends on yearly, monthly, and daily time intervals and adds them to forcasts future points. We picked one of the stores from the dataset and trained facebook prophet on the sales feature. The additive components are shown in fig 1. This allowed us to collect some key insights on the sales. Namely: the weekly and monthly trends are can be stronger then the yearly trends, sales increase drastically before Christmas, and some stores are closed on Saturday.

3 DATA PREPROCESSING

Test Data Differences: In the test data there were no closed days and very few na days. There were also only 70 of the stores from the training data. So we assumed that the na days were open and we removed the stores that were not present in the test set.

We converted categorical data into integer labels. We decided to not use one hot encoding since our machine learning algorithm xgboost tends to undervalue sparse variables due to low information gain. Specifically, we split the date attribute into a year number and month number.

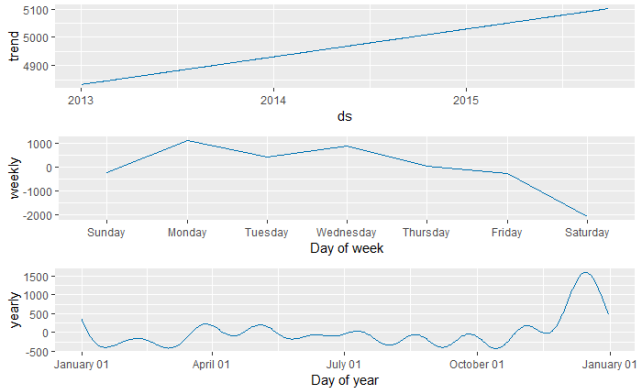

Figure 1: Facebook Prophet components of the yearly, monthly and weekday trends

4 PREDICTION MODEL

The xgboost decision tree algorithm is a favorite amongst Kaggle competitors for it's flexibility, speed, and high performance. For those reasons we focused on this model for the sales dataset. To fully understand xgboost we'll do quick overview of boosted gradient descent and decision trees.

4.1 Decision Trees

Decision tree algorithms partition the training dataset by a series of questions about a particular feature. Each node of the tree is a question about a feature (ex: is age > 18). Each question divides the dataset into small groups creating a tree like structure. This continuous until the data doesn't have any more questions to run though creating a leaf node. Labels of the features belonging to the leaf are averaged and stored as the guess for that leaf node. To predict a label given an object's features you can feed the object into the root node and follow the questions down to a leaf node. Then use the average stored in the leaf as the predicted value for the label.

When building a decision tree, the algorithm needs a metric to evaluate the information gain which is how similar the new split groups are compared to the old group. Typically, we use either the gini coefficient or entropy. We also need to decide on where to split the groups. For categorical features the groups are split on each category. For a numerical feature, the feature values are sorted, and a split point is tested between each point. The split point with the highest gain is selected as the best split point for that feature.

When a new node is created from a set of data the algorithm iterates through all the features, calculates the information gain on each split point and selects the feature and split point with the highest information gain. This done recursively for each node in the tree until the maximum number depth is reached or until there is only one data point left.

4.2 Boosted Gradient Descent

Decision trees are very powerful when used as an ensemble. An ensemble is when many models together toward making a more robust prediction. Boosted gradient descent relies on creating many small trees and sequentially uses the output from one tree to build the next tree.

The process starts with building a normal decision tree using the features and labels. Then the residuals are calculated from the first tree. Another tree is grown to predict the residuals of the first tree using the calculated residues instead of labels. The next tree is grown using the residuals of the second tree an so on. Lastly, the models is evaluated by aggregating the responses from each of the trees. This is called gradient descent because growing decision trees off the residuals of the last model approximates the gradient descent algorithm.

4.3 XGBoost

The most popular implementation of BGD is XGBoost. It's much faster then other GBD algorithms, has built in parallelization, and optimized for scalability. Unfortunately the mechanics of xgboost cannot be easily summarized. However details can be found in the original 2016 paper by Tianqi Chen and Carlos Guestrin XGBoost: A Scalable Tree Boosting System [1].

Experiment and Feature Engineering Like most datasets, we might be able to extract additional insights by engineering new features from the given ones. We'll be experimenting with two sets of additional features that respectively focus on exploiting the reopening effects and momentum effects in our data.

5 FEATURE ENGINEERING

5.1 Reopening Effects

We observed that when a store reopens after a long-closed period that the sales seem to increase in some stores. The 2nd place winner also mentioned this effect in his interview with Kaggle [2]. The intuition is that some people will This led us to experiment with 4 additional boolean features, Opened, Closed, TommorowClosed, LongClose, and Open. Opened and Closed captured if the store was closed and open the day before. TomorrowClosed indicates whether the store will be closed tomorrow. LongClose and LongClose and LongOpen indicates if the store will be closed for more than 1 days and if the store just Opened after being closed for more than 1 day.

5.2 Momentum Effects

Momentum is an object's tendency to continue it's motion. It's is very frequently used in time series forecasting and it's likely be a indicator of future sales. For instance, when stores sales are rising, they are likely to continue rising. The easiest way to capture momentum is with moving averages. We made four moving averages features: $SMA_Q, SMA_{300}, CMA_Q, CMA_{300}$. Respectively they are Sales moving average over the last quarter year and year, and Customer moving average over the last quarter year and year.

The moving averages cannot be accurately calculated for the test data since it is a 60 day period without sales or customer data. In order to use these features we need to approximate what the moving averages are for this period. We considered implementing time series forecasting methods on the moving averages like linear regression or ARIMA. However for the limit scope of this project we decided on simply inserting the most recent moving averages in the the test data. This may not be accurate for the quarterly moving averages but it still might produce useful results.

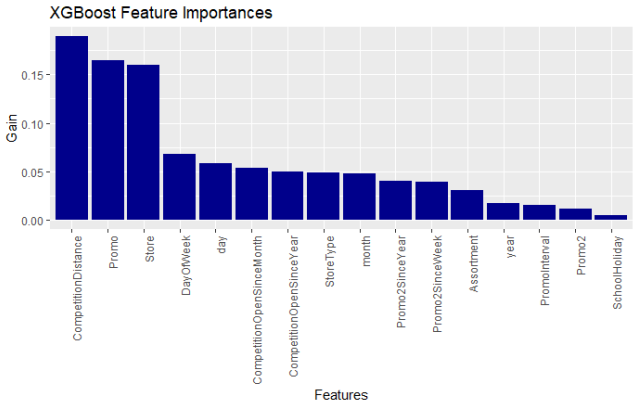

Figure 2: Feature importances for the baseline model.

Model	Train	Validation	Test Pub	Test Pri
Baseline	0.084967	0.098388	0.10684	0.12010
Reopening	0.084725	0.103853	0.10581	0.12666
MA	0.085853	0.116491	0.10632	0.13201
Complete	0.086072	0.110990	0.10632	0.13201

Table 2: The root mean square percent error from the data sets for the four models

5.3 Models

We built four models: one with both feature sets, two with the feature sets separately, and a baseline model with no feature sets. For each model we had a training set set of roughly 600,000 instances, a randomly selected validation set of 10,000, and the fixed test set of 40,000. The xgboost algorithm stopped training after 3200 trees were grown. We used code from a kaggle kernal as a template for our models [3].

6 RESULTS

Kaggle scores the test data predictions with a private and a public score. The public score uses 33% of the test points and the private score uses 66%. table 2 summarizes the error for the 3 models on the train, validation and test sets. The reopening effects model has lowest public score, and the none model has the lowest private score. The most impressive is the private score for the basline model which was able to bring us up to the 80th percentile on the leaderboard.

Also note that the complete model's and the moving averages model's scores are identical. This can be explained by comparing the feature importances of the none model and the moving averages model. Fig 4 and fig 5 are the importances of the none model and the moving averages model. Adding the SMAQ seems to absorb a lot of the gain from the other features. The xgboost model might be over focusing on SMAQ and ignoring the features with a more subtle relationship with the sales.

7 FUTURE RESEARCH

Our moving averages features model did not perform nearly as well as expected falling below the baseline model for the private score. Considering the reported usefulness of moving averages, we expected the model to definitively improve by using them. Studying better methods of estimating the moving averages in the test day or using different look back intervals might yield better results.

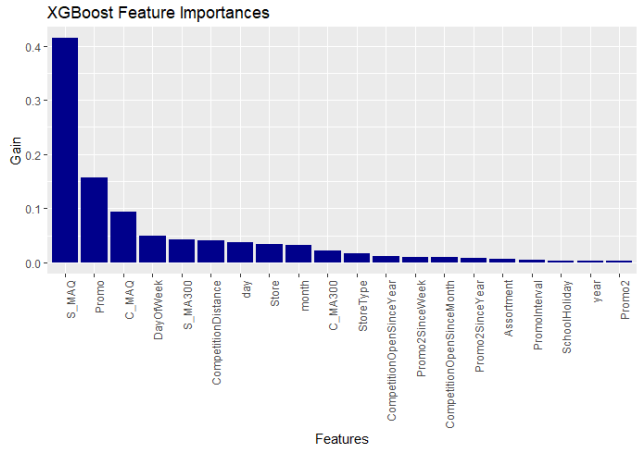

Figure 3: Feature Importances for the moving averages model. Notice how SMA_Q dominates all the other features. The model might be focusing too much on SMA_Q feature and ignoring less obvious patterns.

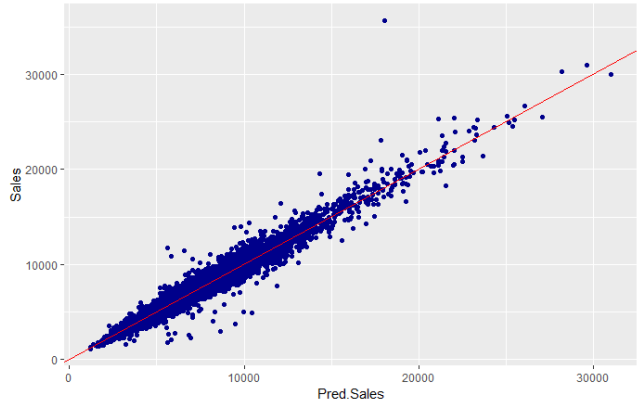

Figure 4: Baseline Model Fit Validation Results

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). ACM, New York, NY, USA, 785-794. DOI: https://doi.org/10.1145/2939672.2939785

[2] Kaggle Team. 2016. Rossmann Store Sales, Winner's Interview: 2nd place, Nima Shahbazi. (February 2016). Retrieved April 28, 2019 from http://blog.kaggle.com/2016/02/03/rossmann-store-sales-winners-interview-2nd-place-nima-shahbazi/

[3] Ben Hamner. Benchmark fork. Retrieved May 1, 2019 from https://www.kaggle.com/seiteta/xgb-rossmann

[4] Facebook. Prophet: Forcasting at scale. Retrieved May 2, 2019 from https://facebook.github.io/prophet/