TeamSR: Kaggle Rossman Store Competition

Contents

1 Libraries to import

Our code uses xgboost. Please go to https://anaconda.org/anaconda/py-xgboost for help installing this library.

2 Overview

This document is designed to help anyone understand the solution developed by two students of NUS to solve the famous Rossman Sales Machine Learning competition on Kaggle. The document starts with discussion about the data organization. We latter discuss the models that were used.

3 Data Processing

Changing the form out our data
First, we added the characteristics of the stores to every data point in train and test datasets. Then we transformed the date to extract the month, the day, the day of the week, the week of the year and the year. We also chose to add a new column that has a 1 if competition exists for a data point or 1 if it doesn't. We made sure to check that the day of the sales is after the day that the competition opened. Finally, we used the PromoInterval column to detect weather there is a promo during the day and stored this new information in a column "IsPromoMonth".

To deal with the closed stores, we adopted an easy solution. We take out all the data points where there are no customers because this means that the store is closed and the sales will be 0. This allows us to train our model only with open stores. After predicting the sales for the test data, we simply replace the sales by 0 if there was 0 customers.
We also had to fill missing values and map string values to numerical representations.

Addition to the data:
When thinking about special days or months for a store, it appeared to us that Fridays and December were likely to behave very differently. For this reason, we added two columns in our data to specify if a sale was happening on a Friday or during the Month of December. This improved our result.
The last addition we did was to specify if the sales was taking place in the beginning of the month (before the 5th day) or at the end (after the 25th day). This allowed us to have better result.

4 Data processing that turned out not good

We initially thought it was relevant to know how long was the competition opened for and since how long a promo had started. He had added this 2 information in our data, but it did not reduce the error (it increased it a little, around 0.005), so we did not use it. Because the time between the sales and the opening of a competing store did not matter, we thought that CompetitionOpenSinceYear, CompetitionOpenSinceMonth could be removed. It turned out that the model was learning better with this information so we kept it.

We also notice that our **train data only covered 3 months.** We thought that we should therefore drop the Month and Year column because it was not relevant for the test data, which was outside these months or year. When testing, it turned out the error was smaller when keeping them so we did.

5 Data Summary

Kept		**Dropped**
Store	DayOfWeek	PromoInterval
Customers	Open	Date
Promo	StateHoliday	
CompetitionOpenSinceYear	StoreType	
CompetitionDistance	Assortment	
CompetitionOpenSinceMonth	SchoolHoliday	
Promo2SinceWeek	Promo2	
Promo2sinceYear		

added	**Description**
Year	The year of the sales
Month	The month of the sales
Day	The day of the sales
WeekOfYear	The weak number of the sales
Competition	1 if competition exists at the day of the sales, 0 otherwise
Friday	1 if sale on a Friday, 0 otherwise
EarlyInMonth	1 if sales before the 5th of the Month, 0 otherwise.
LateInMonth	1 if the sales after the 25th of the month, 0 otherwise.
December	1 if the sales is in December, 0 otherwise.
IsPromo2Month	1 if the sales is in a month during which there is promo (using PromoInterval), 0 otherwise.

6 Models Used

We decided that every group member would train different models and we would average the best results.

Member 1 (A0174733B)	**Member 2 (A0175196U)**
SVR	Catboost (Gradient Boosting)
Building Neural Network with Keras	Random Forest
Random Forest	
XGBoost	

6.1 Models that failed

SVR was long to implement because it required a lot of data processing. We had to use OneHotEncoder and dummies. The best results we could get was using a rbf kernel but the results were not satisfactory; the error was around 0.6, which is twice as much as our best model.

Neural Network using Keras was a waste of time. We had to import this library that is not available in Anaconda. We don't know if this would have given good results because we only got NaN as values. We decided to stop spending time to fix it and concentrate on other methods.

6.2 Satisfactory Methods (but not the Best)

Random Forest was for most of this project the model that gave us the best results. We tuned very carefully the parameters and got an error as low at 0.375.

Gradient Boosting was a very promising method but it did not perform better than random forest. However, when we submitted an average between gradient boosting and random forest we gained one position on the leaderboard.

6.3 Final Method

Xgboost was our final solution. We used it because it is an optimized version of Random Forest. As expected, it's performance was at first close to our previous models. However, by tuning the parameters, we reduced a little our error and we made it from approximatively from the 30th place to the 20th place. We made sure not to overfit the model to our data.

We could get in sample errors of 0.01 with xgboost, however it did not generalize to the test dataset. We used validation to decide when to stop training our model. We trained it using 99% of the data and kept 1% for validation.
The model stops learning when the error in the validation set does not decrease during a hundred steps. The maximum amount of learning steps allowed is 10'000. This causes our in-sample error to be higher but the generalization was better.

Evaluation	**Score**
Training data	0.0366
Validation data	0.0504
Test data	0.36557

Although the model performed well on the validation set, it's error in the test set is 7 times bigger. We believe it is hard to get a better out of sample solution because the training data only covers a very restricted period of time.

Our initial goal was to **combine the solution** of our different models, to avoid overfitting and generalize better. This worked well for random forest and gradient boosting. However, none of our previous solutions worked well enough to be averaged with xgboost to reduce the test error. Therefore, we rely only one the solution obtained by xgboost.

7 Files included with this submission

- README.pdf
- submission.csv
- hw3-rossman.ipynb
- Extra_codes
  - cat_boostrand_forests ipynb
  - SVR ipynb
  - random_forest ipynb
  - keras ipynb

8 Statement of Individual Work

We, A0175196U, A0174733B, certify that we have followed the CS 3244 Machine Learning class guidelines for homework assignments. In particular, we expressly vow that we have followed the Facebook rule in discussing with others in doing the assignment and did not take notes (digital or printed) from the discussions. We've also followed Kaggle competition rules.