Google Analytics Customer Revenue Prediction Challenge Report

Sungguk Cha
UNIST
navinad@naver.com

Abstract

The goal of Google Analytics Customer Revenue Prediction Challenge is to predict GStore revenue per customer based on feature data, including geographical location, channel and a number of text and ID features.

My approach for the challenge are: one is to manage feature pool by encoding, adding, subtracting, and abstracting features with data analysis. The other one is to find optimal ensemble coefficient for gradient boosting models.

1. Introduction

The challenge([1]) is to analyze a Google Merchandise Store customer dataset to predict revenue per customer. The outcome will be more actionable operational changes and a better use of marketing budgets for those companies who choose to use data analysis on top of Google Analytics (GA) data.

The key to solve regression problems is data analysis. Based on the analysis, a strategy is to manage features, and engineering deep learning models. This report explains how I managed the features and how did I train the model to improve the performance.

2. Approaches

2.1. Features

Making features of good quality is the most important key in regression challenges. Making more features from present or artificial interpretation, removing some features which are redundant or interrupting, and modifying features such as encoding differently are the ways to set up the features for the regression. Referred ([2]).

2.2. Encodings

I tried three ways of encoding; label encoding frequency encoding and mean encoding. However for final submission, I used label encoding only, because I could not find appropriate way to integrate differently normalized and encoded features. Referred ([3])

2.3. Ensemble

For regression challenges, boosting algorithms XGBoost(2016), LightGBM(2017) and Catboost(2017) are the most powerful methods. Many Kaggle's winner interview reports that the boosting algorithms and the ensemble are the common strategy.

I used four models xgboost, lightgbm and two catboost models. For the catboost models, they have difference in their depth and learning rates.

However, the ratio between boosting models in ensemble matters. For example, for final result, (0.5 * lgbm + 0.1 * xgboost + 0.4 * catboost). I tried to improve performance through tuning the ratio. According to the validation score (rmse of the target), I tune the ratio non-linearly.

I thought feature processing is enough, so ensemble can be the last strategy to improve final score. For ensemble baseline, I referred ([4]).

3. Experiment

3.1. Dataset

Figure 1. shows the data fields. The dataset is real records from given periods, for example, 20160801 to 20170801 for the train set, and 20170802 to 20180430 for the test set.

- channelGrouping: The channel via which the user came to the Store
- date: The date on which the user visited the Store.
- device: The specifications for the device used to access the Store.
- fullVisitorId: A unique identifier for each user of the Google Merchandise Store.
- geoNetwork: This section contains information about the geography of the user.
- sessionID: A unique identifier for this visit to the store.
- socialEngagementType: Engagement type, either "Socially Engaged" or "Not Socially Engaged".

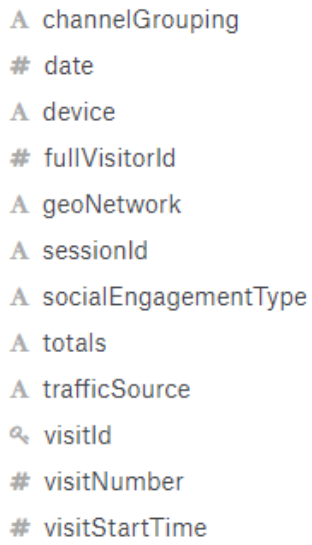

Figure 1. Dataset categories

- totals: This section contains aggregate values across the session.
- trafficSource: This section contains information about the Traffic Source from which the session originated.
- visitId: An identifier for this session. This is part of the value usually stored as the _utmb cookie. This is only unique to the user. For a completely unique ID. It should be used as a combination of fullVisitorId and visitId.
- visitNumber: The session number for this user. If this is the first session, then this is set to 1.
- visitStartTime: The timestamp (expressed as POSIX time).

3.2. Features

Given **60** features, **43** features are added and **33** features are removed (or removed when training). Total **70** features were used to train.

3.2.1 Making more features

- 'id_incoherence': Most visitId values are the same as the date. For a few logs which visitId != date, this feature is set to be true.
- '#visitId': It is the number of the count per each visitId.
- 'weekday'
- 'time': value between 0 and (24 * 60 * 60) 86400.
- 'sources': adding two columns 'trafficSource.source' + 'geoNetwork.country'. For example of the result, 'google' + 'Turkey'.
- 'campaign, medium': 'trafficSource.campaign' + 'trafficSource.medium'
- 'browser, category': 'device.browser' + 'device.deviceCategory'
- 'browser, os': 'device.browser' + 'device.operatingSystem'
- 'devicecategory, channelgrouping': 'device.deviceCategory' + 'channelGrouping'
- ...

In order to correlate some columns, I made some arbitrary features like 'browser, category' which is the combination of the two columns, 'device.browser' and 'device.deviceCategory'. Total 43 features are added. id_incoherence, #visitId, weekday and time features are added. The other 39 features are the combinations of existing columns.

3.2.2 Removing some features

The total number of logs, dataset, is about 904k for trainset, and 805k for testset. Regarding the number of the logs, it is very reasonable to ignore categories which appeared less than 1k times in order to reduce the number of categories to train as much as possible. I cleared up many categories, replacing the categories with an arbitrary NaN.

Among 60 columns of training set, I filtered 31 columns. The removed features are constant or close to constant in which constant columns are useless in regression.

During training, 'date', 'fullVisitorId(index)', 'totals.transactionRevenue', 'visitId', 'visitStartTime', 'month', 'day', 'help' and 'sessionID' are excluded. 8 more features are excluded in learning.

3.3. Encodings

Table 1. rmse results per encoding

	RMSE result
Label encoder	1.4675
Frequency encoder	1.4545
Mean encoder	1.4470

3.3.1 Label encoding

I used label encoding functions of 'scikit' library. After feature preprocessing, I converted every feature into string, then label-encoding, and lastly, convert the result into floating point type. It has a downside that the encoding order for the features matters.

3.3.2 Frequency encoding

For each column(feature), normalize each category according to the number of the appearance. In a sense that it implies a little interpretation of the data, while label encoding encodes in an arbitrary way, it has better performance than label encoder.

3.3.3 Mean encoding

Mean encoding encodes each label by target mean. Because it is directly correlated to the target, the result was the most powerful in terms of rmse result. However, as most users(fullVisitorId) has zero target(no revenue transaction), it can result in overfitting depends on the training dataset.

3.4. Boosting Algorithms and Ensemble

Except adding more features by data analysis and optimizing boosting algorithm, I thought the ensemble of the gradient boost models can be a key to improve the perforamance. As a result, I got RMSE 1.2911 reaching 12% of the leader board (397th/3411), when the top 10th has RMSE 1.2828 in Nov 7, 2018.

3.4.1 Ensemble coefficient optimization algorithm

s1 to s4 are the scores from validation. Algorithm1 shows the method.

Data: Max, s1, s2, s3, s4
initialization;
ns = []
ns.append((1/s1, 1))
ns.append((1/s2, 2))
ns.append((1/s3, 3))
ns.append((1/s4, 4))
min = 1.0
max = Max
sort_increasing_order_along_axis1(ns)
M = ns[3][0] // Max value
m = ns[0][0] // Min value
i = 1
while *i is less than 5* **do**
    v := ns[i][0]
    res[ns[i][1]] = (m * (M - v) + M * (v - m)) / (M - m)
    res[ns[i][1]] = exp(res[ns[i][1])
    i += 1
end
return res

Algorithm 1: Ensemble coefficient tuning algorithm

Fig. 2 shows an example of the result. Normalizing the values, use them as the coefficients for the ensemble. I could find the optimized Max value (2.0, 2.25, 2.5) through binary search between 1.5 to 3.5.

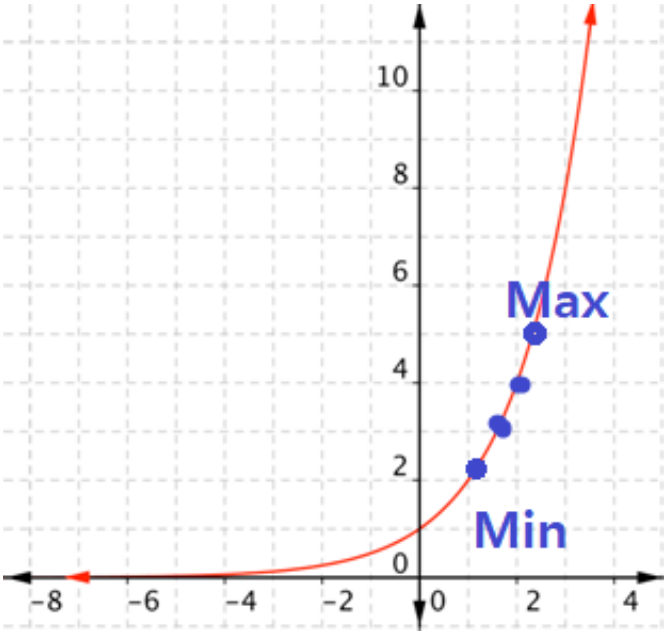

Figure 2. Non-linear ensemble

4. Discussion

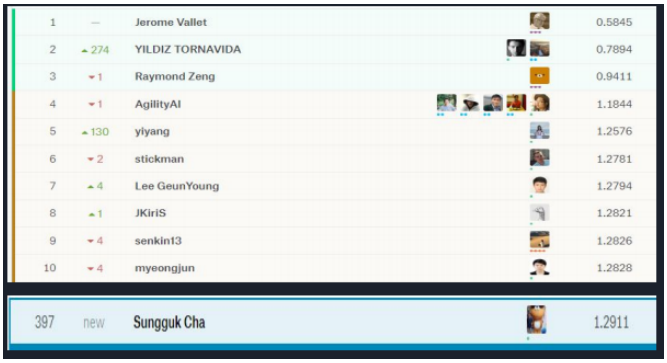

Figure 3. Competition result, Nov 7, 2018

In Google Analytics Customer Revenue Prediction, I tried regression challenge. Through feature processing, I added, removed and revised some features based on the data analysis. With several encodings, I could know encoding matters quite a lot in regression challenge. Lastly, by ensemble of boosting models and tuning the ensemble ratio, I could significantly improve performance.

4.1. Weekday and date

For weekday and date, I thought a lot. In an intuition that weekday and date are important factors for sales(our target revenuetransaction), those features are must be in-

cluded. For weekday, by counting all transaction per a weekday, I mean encoded them and gaussian-normalized them. This approach is more intuitive than label encoding for weekday, like basic output from well-known function pandas.DataFrame's get_weekday. For the dates, first get the number of transaction per dates, removing years like (181121 into 1121). Then mean encode each date according to the number of the transaction (366 dates including Feb.29). Due to the fact that test set's date (e.g. Nov. to Feb. only) does not cover all dates, I should be careful.

4.1.1 One-hot encoding

Because one-hot encoding does encode scale-independently, it is a really fair encoding. However, like label encoding, the characteristic that it is irrelevant to the target can be a flaw if an encoding is one of the keys for the regression.

References

[1] Google Analytics Customer Revenue Prediction, https://www.kaggle.com/c/ga-customer-revenue-prediction

[2] Sergey Ivanov (MSU MMP), by Kaggle user 'karkun'

[3] Which-Encoding-Is-Good-for-Time-Validation, Lee Youhan

[4] baseline-with-lightgbm-xgb-catboost, Wang Jianfei