Google Analytics Costumer Revenue Prediction

Ana Sofia Medeiros Oliveira and Fábio Henrique da Silva Pereira

I. Introduction

In this project we'll use data retrieved from Kaggle, that was in turn taken from Google Analytics [1]. The data refers to users' sessions from the Google Merchandise Store, between August 2016 and May 2018. Our objective is to predict the revenue of a user's session given the information regarding that same session.

The business utility of such a project is one that is not entirely achieved in this paper alone, as that would be, for instance, to track an online user in real-time and predict the likelihood and, furthermore, the amount of a transaction. This could create the chance of customizing the user's experience in real-time considering what we know about the session's outcome.

II. State of the Art

As it is the first time that there is such a descriptive data set of an online store's sessions' information, there is no scientific paper that we can refer to related to this project.

There is, however, the Kaggle community, which worked in this data set too, as it comes from a Kaggle competition. From our research we can state that what's most effective in this problem is regression using Gradient Boosted Decision Trees (GBDT) [2]. In short, GBDT are a sequence of decision trees where each of them tries to correct the error made by its predecessor. There were some who used Artificial Neural Networks, but GBDT methods produced better results.

III. Data Preparation

Our data set is composed of over 1.7 million observations corresponding to different sessions in the Google Merchandise Store and 11 features of which 4 are in JSON format. After extracting the attributes from these JSON columns, we had a total of 58 features.

This data set had no prior preprocessing, since it came straight from Google Analytics, so it needed a lot of cleansing. It had a lot of missing values in different forms – besides NAs, there were entries with ”not available in demo dataset” and ”not set”, for example. We converted all these into NAs. The resulting frequencies of missing values are presented in Figure 7.

There were some attributes that didn't provide any new information to our model, so they were removed from the dataset.

As some categorical variables had a lot of different values that represented only a small percentage of the total rows, we decided to convert all values that amounted to less than 0.01% of the corresponding feature to ”other”, as that will make our model generalize better for values never seen before.

We then derived some variables such as the day of the week, the month, and how many days have passed since the user's last visit as they will probably have some useful information to our model.

In the end of data preparation, we finished with 33 features. We decided not to drop columns based on their NA frequencies because the assigned values may still have valuable information.

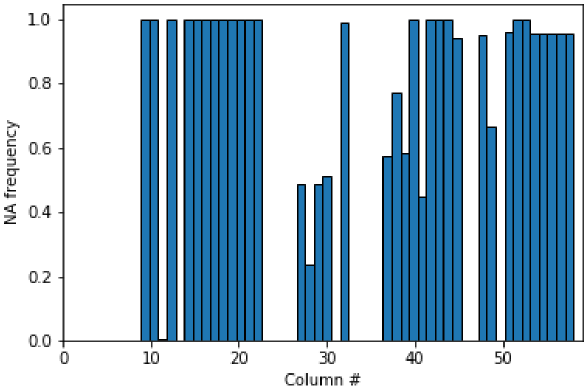

Fig. 1. Proportion of NAs per column number.

IV. Data Visualization

The target variable in our problem is the transaction revenue, which has 98.9% of zeros, leaving 18514 non-zero values. A zoomed in non-zero values distribution is in figure 2. The variable summary with and without zeros is presented in tables I and II.

Finally, we present some of the variables' distributions that seem most important to our model, in Figures 3 to 6.

Note that in order to have a good insight about the importance of each of the variables, we must be aware of both the total transaction revenue and the frequency of each of them. This is because we can, for instance, have a great transaction revenue at a certain value, but that can be caused by said value being predominant in our data set. This wouldn't represent the actual likelihood of a transaction, since it would be biased by the imbalance in the data set.

So, we choose to present a combination of both fraction of sessions and fraction of revenues in function of a given variable.

V. Modelling

As stated before, our goal is to predict the transaction value of a user's session. As we want to predict the full range of values, especially big transactions which represents a tiny

TABLE I
TARGET VALUE SUMMARY

Min	1st Qu	Median	Mean	3rd Qu	Max
0.000	0.000	0.000	1.55	0.000	47082.06

TABLE II
NON-ZERO TARGET VALUE SUMMARY

Min	1st Qu	Median	Mean	3rd Qu	Max
1.20	28.96	52.79	142.82	108.97	47082.06

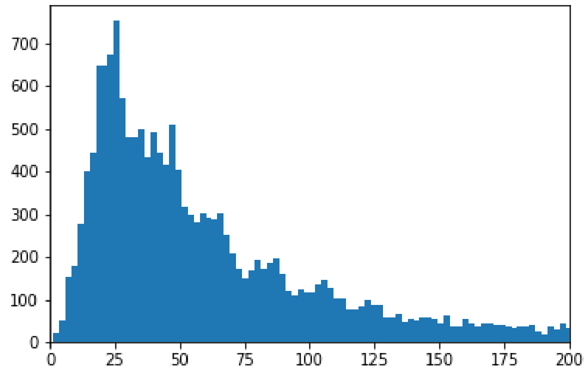

Fig. 2. Histogram of the positive instances of the original target value. Note that we cut the x axis at 200, because the values reach the order of dozens of thousands.

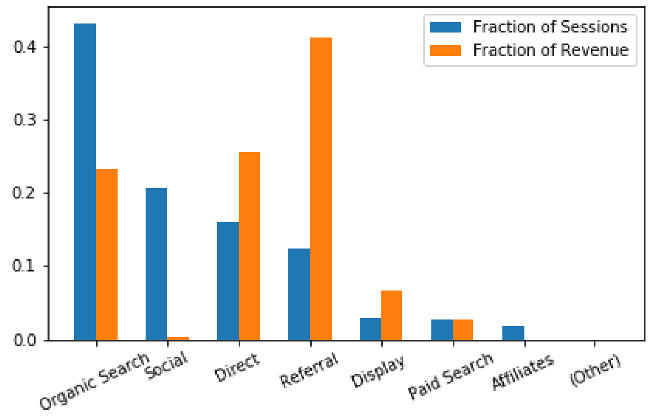

Fig. 3. Channel Grouping distributions. This represents where the user came from.

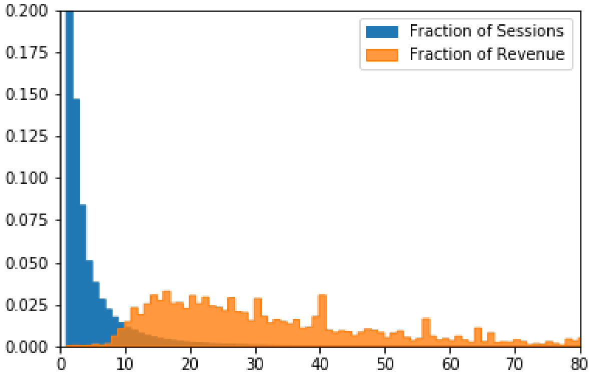

Fig. 4. Distributions of pages viewed in a session. Note that we cut the y axis at 0.200.[1] The sessions where only 1 page was seen account for 0.5 of the total sessions. [2]

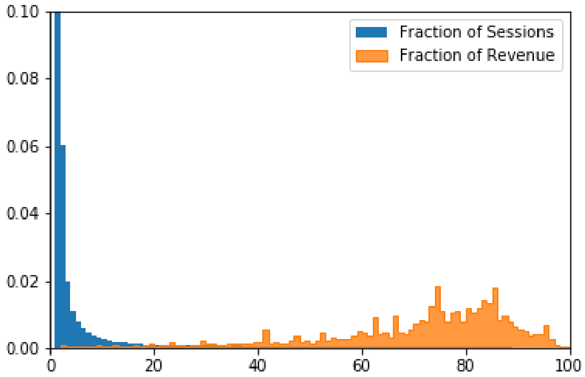

Fig. 5. Distributions of the sessions' quality dimension. This is an estimate of how close a particular session was to transacting, ranging from 1 (low quality) to 100 (high quality), calculated for each session. A value of 0 indicates that Session Quality is not calculated. Note that 0 quality represents 0.8 of the total sessions.

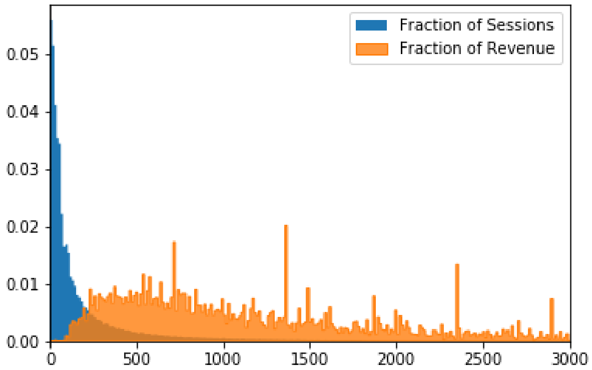

Fig. 6. Distributions of time spent on site, in seconds.

amount of our data set, we will use Root Mean Squared Error (RMSE), as this metric gives a great importance to outliers.

To achieve this goal, we will first use the state of the art to create a regression model that is better than always predicting the average of the target value. Thereafter, to try to achieve better results, we will do a pipeline of models that will first separate the zero from the positive revenues, and then regress only on the instances that the classifier identified as

[1] This was done because otherwise most of the distribution wouldn't be visible.

[2] The total area of the histogram may not be 1 due to the variable's NA frequency.

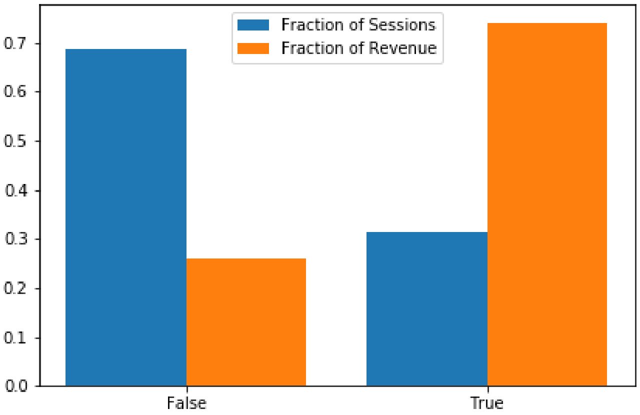

Fig. 7. TrueDirect values distribution. This field is true if the user typed the URL into the browser, came to the site via a bookmark or if 2 successive but distinct sessions have exactly the same campaign details.

positives. This idea comes from the assumption that it is easier to separate zero from non-zero revenues first, rather than to immediately regress the actual values, leading to better results in the identification of zeros. Moreover, the regression step of the pipeline will do just as good a job as the standalone regression. Those 2 models combined will, if our assumption is correct, reduce the overall error of our predictions.

Instead of using the target value, we will use by default the natural logarithm of it. This was done because we saw that with the logarithm, we achieved a greater score in all our models (in relation to predicting always the mean) and that there was a greater homoscedasticity in the residuals of our models as a function of the respective fitted values.

We will train all the models using the library LightGBM [3]. Other Gradient Boosting Machines, such as CatBoost and XGBoost, might be tried for this objective as they might provide better scores.

For reference, we always divide our (unbalanced) dataset in 90/5/5 (train/validation/test set).

A. Regression using the state of the art

To reproduce the state of the art, we just trained a model with a procedure from LightGBM that uses GBDT. With a bit of parameter tuning, we easily got a better score than predicting always the average of the target value, as will be shown in the next section.

B. Pipeline

As stated before, this pipeline will feature two different models: one classifier and one regression, both using GBDT.

1) Data flow: Each new instance that we want to predict will flow the pipeline in the following manner: first, it will go through the classifier. If classified as zero, there is nothing more to be done; otherwise, the instance will proceed to the regression which will assign it a non-negative value.

2) Classifier: The classifier must be able to separate the zero from the non-zero cases. Due to the nature of our goal, we will prefer a classifier that can better identify the non-zeros, since our regression can still approximate the misclassified zeros. On the other hand, any misclassified positive instance will result in a failure to predict revenue and in an error that can't be reduced by the regression.

To achieve this, we tried a few separate models:

- A classifier trained on the unbalanced training set.
- A classifier trained on a balanced subset of the training set, with a reduced number of instances.
- An ensemble of classifiers, that uses the full training set, but each of the classifiers uses a balanced subset of the full training set. This means that we are repeating all the positive instances between the classifiers and not repeating any zero instance.

3) Regression Model: The regression must be able to determine the most likely revenue value of a given instance. As stated before, the only instances that will reach this step of the pipeline are the ones labeled as positive by the classifier. Unfortunately, our classifier can and most certainly will mislabel some zero revenues as being positive, which means that we will have zeros among our positive instances, and that therefore we must train our regression model to predict the full range of revenues.

This was already done before in the simple regression using the state of the art, so we'll use that same model in this step of the pipeline.

VI. Results and Evaluation

We now evaluated all the models done:

1) GBDT Regression using all the training data
2) Ensemble of GBDT Classifications + Model 1
3) GBDT classification using all the training data + Model 1
4) GBDT classification using balanced set + Model 1

In Table III, we present the RMSE of each of the 4 models mentioned above, as well as of the model that always predicts the mean, in order to have a perspective of the models' performance.

TABLE III
RMSE in the Validation Set

Mean Pred.	**Model 1**	**Model 2**	**Model 3**	**Model 4**
0.44045	0.36050	0.36040	0.36003	0.36040

All of our classification models are able to identify most of the zeros, but still misclassify some as non-zeros. Despite our models having a good recall, because our data set is unbalanced, the false positives outnumber the true positives by a proportion of 3:1. This causes our regression to approximate the distribution to zero in order to reduce the error. This can be seen in Figure 9 and Figure 8, where the distributions of the predicted values are completely different from the real values distribution.

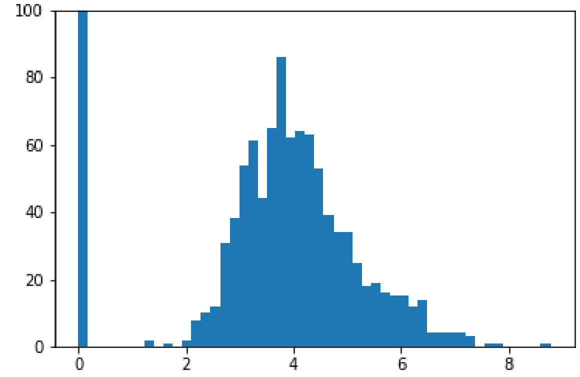

Fig. 8. Distribution of the target values in the validation set. Note that the value 0 corresponds to close to 85000 sessions.

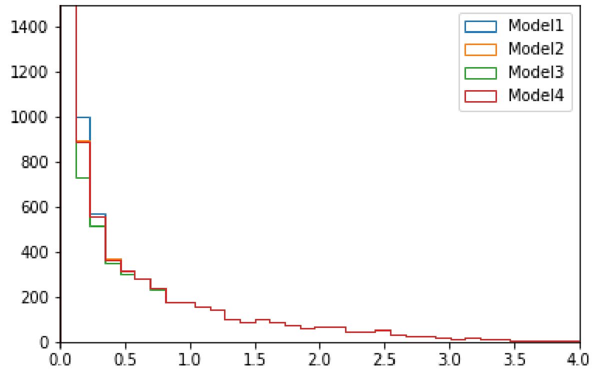

Fig. 9. Distributions of each of the 4 models for the predicted values in the validation set.

All of our models have very similar distributions, so we can only look at the scores on the validation set to reach a conclusion about the best model – Model 3.

Finally, in Figure 10, we see how the distribution of the target values in the test set compares to what our best model predicts those values to be. Our Model 3, in the test set, achieves a reduction of 20.5% in RMSE when compared to always predicting the mean.

VII. Conclusions

We managed to achieve a better score than the state of the art, but by such a small margin that we're not sure if such an improvement is statistically significant.[3]

The score could be massively improved if the classification had a better precision, maintaining at least the achieved recall, so that it wouldn't mislabel as many zeros as it does now. This would allow us to do a regression using as training set only the positive instances, and in theory, achieve a much better RMSE.

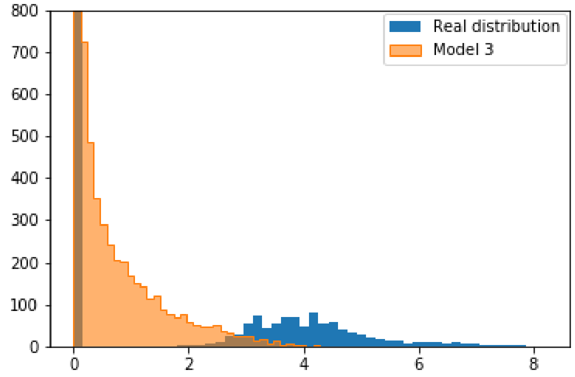

Fig. 10. Distribution of the target value and model 3's predictions for the test set. Our model predicts that more than 80000 cases are zeros.

References

[1] https://analytics.google.com/

[2] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

[3] Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." Advances in Neural Information Processing Systems. 2017.

[3] Given such a big data set, we supposedly don't need a method like CV or bootstrap to check if one model is better than another, but in this case, as the improvement is so small, we're not sure if we can infer that our model is actually better. Nonetheless, every time we trained the models, ours achieved a better score than the state of the art.