Sarah Yurick BS

Kameron MacKenzie BS

Data Science & Analytics and Computer Science

Binary Customer Transaction Predictions for Santander Bank

with

Professor Pan Li

Department of Electrical Engineering & Computer Science

Case Western Reserve University

Project Timeline

Week of February 25:

- Initial exploration of potential DSCI 133 project interests, focusing on accessibility of datasets
- Kameron and Sarah agree to investigate Kaggle competitions, identify Satander competition

Week of March 4:

- Santander Competition Project approved for DSCI 133 course project
- Kameron and Sarah further individual data science techniques and programming

Weeks of March 11 & March 18:

- Customer Transaction Prediction competition formally entered on Kaggle
- Kameron and Sarah individually begin basic explorations of the dataset provided

Week of March 25:

- Kameron begins researching possible approaches based on our finds
- After research, Kameron narrows down on a Naive Bayes model, Random Forest Algorithm, or Light Gradient Boosting
- Kameron and Sarah run independent attempts which perform poorly

Week of April 1:

- Sarah and Kameron investigate public kernels for successful approaches to implement
- After further research, Kameron and Sarah decide to pursue LightGBM algorithm
- Sarah tests and runs basic LightGBM models
- Kameron begins work on presentation and report, focusing on introduction and exploration

Week of April 8:

- After studying the results of past models, Sarah experiments with adding more parameters to protect against overfitting
- Kameron verifies exploratory analysis, runs LightGBM models without Kaggle kernels and compares runtime, outcomes, and performance
- Sarah submits the top 2 performing models to the competition by the final submission deadline

Week of March 15:

- Kameron and Sarah finalize presentation, continue work on report, and reflect on development as data scientists

BINARY CUSTOMER TRANSACTION PREDICTIONS FOR SANTANDER BANK

Kameron MacKenzie, Sarah Yurick, *Case Western Reserve University*

Abstract--This paper describes Kameron MacKenzie and Sarah Yurick's research and implementation of various machine learning techniques and their application to the Santander Customer Transaction Challenge hosted by Kaggle.

***Index Terms*—Kaggle, LightGBM, Naive Bayes, Random Forest**

I. Introduction

Binary classification is a common, applicable family of prediction problems in the corporate world that can be solved with appropriate machine learning models and techniques. In partnership with the Kaggle online data science community, Santander Bank provided data which was said to have the same structure as the real data available to the company, with the goal to identify which customers will make a specific transaction in the future. We were given a dataset containing 199 columns of variables and a labelled binary "target" column. In order to compete, we had to analyze public approaches to the task as discussed in the Kaggle community, learn new machine learning and data analysis methods, and develop our own unique and competitive solutions.

II. Methods and Procedures

A. Exploratory Analysis

The data provided by the Santander Bank contained 200 variables per customer, with 200,000 customers total, and each variable labelled as "var_0" through "var_199" and the binary column labelled as "target." Because the training data provided a complete "target" column, this was a supervised machine learning task.

After some basic inquiries into the dataset, we found that there were no missing values, no unreasonable outliers, and all of the data had matching types. Thus, no data cleaning was required.

Next, we attempted to find insights into the data by using the describe() function to provide descriptive statistics, but because the variables covered a broad distribution of means and deviations, it was difficult to draw any conclusions. The anonymity of the variables meant that generally useful metrics of mean, standard deviation, ranges, and other descriptive statistics provided little analytic, actionable information.

Therefore, more in-depth techniques were used for further exploratory analysis. We noted that the training data was unevenly distributed between the target outcomes, with significantly more "no transaction" rows (target = [0]) than "transaction" rows (target = [1]). Therefore, we could stratify the rows based on this difference and observe the distributions of each variable based on its labeled outcome. This would allow for the graphical, qualitative analysis of the difference in distributions of each variable and how that distribution changes when the variable is connected to a transaction versus not. Then, by looking at these differences in distributions, the most important and salient variables were identified as the ones with the greatest change when distributed based on target outcome. For example, "var_01" in Figure 1 below likely has much more impact on the outcome of the target variable than say "var_03", since the latter has a much more similar distribution between target variables.

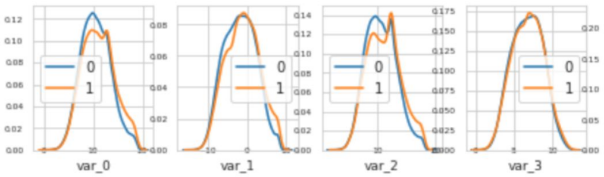

Figure 1 - Example Variable Distributions

The production of this report was motivated by the Introduction to Data Science (DSCI 133) course project, assisted by Dr. Pan Li and Tianxi Ji. Additionally, Santander Bank provided funding and dataset for the Kaggle competition in which the results of this report were entered and ranked. Finally, the Kaggle data science community at large provided significant assistance in the development of our algorithms and models.

Kameron MacKenzie and Sarah Yurick are both candidates for a Bachelors of Science in Computer Science and Data Science at Case Western Reserve University, in the department of Electrical Engineering and Computer Science.

Finally, we also constructed a dataframe of Pearsons pairwise correlation coefficients for all variables from var_0 to var_199. This was accomplished using the popular numerical manipulation package *pandas* in Python and its correlation function (.corr()). Then, the resulting list was sorted so that any significant correlations could be observed. However, the highest correlation was found to be between variables 139 and 26 at 0.0098. This is hardly notable, and lends to the theory that many of the variables are at least linearly independent. However, it does provide insight into possible later feature engineering by illustrating the low linear dependence of each variable on another.

B. Researching an Approach

After our exploratory data analysis, we began researching appropriate models to begin implementing. From our analysis, we found low values for all pearson pairwise correlations, which meant that the variables were highly independent of one another. From this, we considered a Naive Bayes model, which relies on Bayes theorem, conditional probabilities, and approximate distributions. Naive Bayes classifiers inherently assumes an independence of the features. However, because Naive Bayes is not reliant on set parameters, meaning that less can be tweaked and modified, and because we were using the normal distribution curve, our calculated probabilities were skewed.

Throughout our analysis, we had also found that the variables ranged from identical to vastly different distributions, leading us to believe that the features were of varying importance and influence in determining the target. Because of this, we considered a Random Forest Algorithm to be a possible approach because of the decision tree structure and simpler implementation. The Random Forest Algorithm relies on building and merging decision trees using random subsets of features, with a heavy reliance on sklearn and RandomForestClassifier object functions significantly reducing programming struggles. There were a high number of parameters that could be modified as well as room for improvement by dropping features, which could easily be done by using built in methods to determine feature importance. However, after playing around with the Random Forest Algorithm, we simply were not getting as high of results as we wanted, so we decided to abandon this idea.

Finally, because we were given such a large dataset with anonymized data which made manual exploration and insight difficult, we considered gradient boosting because it works well with high quantities of data. Upon looking into Kaggle's community forums, we found that many competitors were using what is called Light Gradient Boosting, or LightGBM. Light Gradient Boosting uses leaf-wise tree growth for a faster gradient boosting algorithm, which was ideal for such a large dataset. However, a fast learning time means a lower opportunity cost for each attempted kernel. But despite its disadvantages, we decided that fast training, low memory usage, and compatibility with large datasets made LightGBM the best way to go.

C. Implementing Light Gradient Boosting

From the start, using LightGBM had its limitations. Despite there being a high number of parameters to tweak, the documentation and resources for LightGBM were limited and vague. Because of this, we had a lot of trial and error attempts in order to try to get close to appropriate values which could be used for each parameter.

As the deadline for the competition came closer, more and more competitors began discussing the "magic" of their models. The "magic" was creating 200 new variables which store the frequency (i.e. the number of counts) of each of the corresponding original variables. Although this concept seemed vague to us, it made sense given the wide distribution of the data. However, during the time of the competition, nobody explicitly sharing their code, and coding this on our own seemed to be beyond our skill set, so we decided to explore other options and try to find our own unique solution.

We constructed our first LightGBM model with guidance from the code that others had posted on the community forums. While running in the Kaggle cloud, the outputs seemed to suggest that the model was training well and that we would achieve a score in the mid-90th percentile range. However, once it was done running, our score dropped to around 88%, which was good, but not as great as our training numbers. We realized that this was likely due to overfitting issues, so we began researching ways to fix this.

After some research, we found that LightGBM had built in parameters "lambda_l1," "lambda_l2," and "min_gain_to_split" to deal with overfitting. Once again, the documentation behind this was pretty vague, but after some trial and error we realized that all three values are meant to be a decimal percentage between 0 and 1. Lambda_l1 handles L1 regularization, which adds the "absolute value of magnitude" of the coefficient as a penalty term to the loss function and generates more sparse outputs by shrinking the coefficients of less important features. Meanwhile lambda_l2 handles L2 regularization, which adds the "squared magnitude" of the coefficient as a penalty term to the loss function. Finally, "min_gain_to_split" works by fine-tuning the minimal gain required to perform a split. We attempted mixing and

matching these three parameters in order to walk the line between overfitting and underfitting, resulting in a slightly higher score of 90.1%.

With this, we hit what was dubbed by the Kaggle community as "the 901 barrier." In other words, other competitors were getting around the same score as we were, and those that were able to get scores about 90.1% were not sharing their strategies with the larger community. In addition to this, we were also hindered by the fact that we were limited to 3 submissions per day, and our submissions could take around 2-4 hours to run in the Kaggle cloud. By this time, the competition was almost to a close, so we fine-tuned our model one last time and submitted our final results.

III. Results

The Santander Customer Transaction Prediction Challenge officially ended on April 10, 2019. Because Kaggle creates a tentative score based on how well your predictions fit the first half of the data, our submission initially achieved a score of 90.115%. However, after the competition was over and our model was tested against the remaining half of the data, our score dropped down to 89.981%. Once again, this was likely due to overfitting because we ended up submitted the model that had the highest score instead of one of our models that was more protected against overfitting.

Despite the drop in our score, we ended up scoring in the top 21% of the competition. After the competition was officially over, the top competitors released their strategies and code for the community to analyze. As predicted, all of the top performers converged to using LightGBM, with their own added twists boosting them to the top of the leaderboard.

IV. Conclusion

This project began with a thorough exploratory analysis, showing the importance of initial investigations into a dataset before attempting to fit complicated models. It was through these initial insights that the best algorithms for this classification problem were discovered. The resulting experimentation allowed for the discovery of LightGBM and understanding of its behavior with this specific dataset and its quirks.

LightGBM was discovered with assistance from the Kaggle community, and its implementation was found to be fairly simple at first. This algorithm is fairly new, however, and left the authors to experiment and dig through documentation to make further progress. Additionally, LightGBM is very sensitive to overfitting, leading to a significant amount of effort to tweak the model parameters to deal with this issue. In the end, the file we used for our final submission was not as protected against overfitting as some of our other models were, resulting in a lower score than what was predicted. Regardless, the model's accuracy was still above 90%, able to consistently predict the testing data's purchasing outcome. This project demonstrates the applicability and power of the new LightGBM algorithm, and illustrates its convenience for both industrial and academic machine learning problems.

Resources

[1] P. Mandot, "What is LightGBM, How to implement it? How to fine tune the parameters?" *A Medium Corporation*. Medium, 17 Aug. 2017, https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc.

[2] P. Khandelwal, "Which algorithm takes the crown: Light GBM vs XGBOOST?" *Analytics Vidhya*. Analytics Vidhya, 12 June 2017, https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/.

[3] N. Donges, "The Random Forest Algorithm." *Towards Data Science*. Medium, 22 Feb. 2018, https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd.

[4] MJ Bahmani, "Santander ML Explainability." *Kaggle*, Google, 10 Mar. 2019, https://www.kaggle.com/mjbahmani/santander-ml-explainability/notebook.

[5] Kaggle user Dromosys, "SCTP-working(LGB)." *Kaggle*, Google, 21 Feb. 2019, https://www.kaggle.com/dromosys/sctp-working-lgb.

[6] G. Preda, "Santander EDA and Prediction." *Kaggle*, Google, 17 Feb. 2019, https://www.kaggle.com/gpreda/santander-eda-and-prediction/notebook.

[7] "3.2.4.3.1 sklearn.ensemble.RandomForestClassifier." *scikit-learn 0.20.3 documentation*, scikit-learn developers, 2018, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[8] "lightgbm.sklearn." *LightGBM 2.2.4 documentation*, Microsoft Corporation, 2019, https://lightgbm.readthedocs.io/en/latest/_modules/lightgbm/sklearn.html.

[9] C. Deotte, "200 Magical Models - Santander - [0.920]." *Kaggle*, Google, 11 Apr. 2019, https://www.kaggle.com/cdeotte/200-magical-models-santander-0-920.

[10] A. Nagpal, "L1 and L2 Regularization Methods." *Towards Data Science*. Medium, 13 Oct. 2017, https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c.