High Dimensionality with Customer Prediction

Nathan Weber

***Abstract*—High-dimensional datasets present unique challenges in predictive modeling, requiring thoughtful feature selection and model choice to achieve robust results. This report explores several machine learning approaches to predict customer transactions, focusing on handling high dimensionality effectively. Initial attempts, including KNN and Principal Component Analysis (PCA), highlighted the limitations of traditional methods in managing the complexity of the dataset. Gradient boosting models like XGBoost and LightGBM were then employed, with LightGBM ultimately achieving a 92% accuracy rate. This demonstrates its suitability for dealing with high-dimensional data and its effectiveness compared to other techniques, without resorting to overfitting tactics.**

***Index Terms*—Machine Learning, XGBoost, LightGBM, Santander, Predictive Modeling.**

I. Introduction

Banco Santander, a leading financial institution based in Spain, has initiated a challenge to improve its predictive capabilities for customer transactions. By leveraging data science and machine learning techniques, the goal is to identify which customers are likely to make specific transactions in the future, irrespective of the transaction amount.

I am defining success not by winning the Kaggle competition. There are two reasons for this: first, the competition has already concluded, meaning there is no monetary reward, and attempting to win would be fruitless. Second, I want to focus on the concept of high dimensionality and explore possible models for that application. Success will be defined by achieving a score higher than 90%. Given that the highest score achieved in the competition was 92.5%, I consider this goal to be ambitious enough.

II. Dataset Overview

In this project, I was provided with three datasets: train.csv, test.csv, and sample_submission.csv. The dataset is highly dimensional, containing 200 features, which presents additional complexity and necessitates careful consideration of feature selection and dimensionality reduction techniques.

The Santander Customer Transaction Prediction dataset is an anonymized dataset provided by Santander Bank, with the goal of predicting whether a customer will make a specific transaction. The features, named as *var_0* to *var_199*, are all numerical, and their actual meanings have been deliberately withheld to maintain confidentiality. This makes feature engineering particularly challenging, as there is no domain-specific context to guide the modeling process.

Additionally, the dataset is highly imbalanced, with a majority of the target values being 0 (indicating no transaction) and only a small fraction being 1 (indicating a transaction). This imbalance adds another layer of complexity, as standard machine learning models may struggle to effectively learn from such data without appropriate techniques to handle class imbalance, such as sampling methods or specialized loss functions.

Detail Compact Column 10 of 201 columns

ID_code	var_0	var_1	var_2	var_3
test_0	11.0656	7.7798	12.9536	9.4292
test_1	8.5304	1.2543	11.3047	5.1858
test_2	5.4827	-10.3581	10.1407	7.0479

Fig. 1. Dataset Overview

III. KNN Classifier

My first attempt was to run it through sklearn's KNN Classifier. I wanted to train it with 1, 3, and 5 neighbors to see if I was getting closer to the truth. To be honest, not much research was put behind this approach; I was hoping for an easy early win. The result was abysmal—the ROC score was so bad I didn't even submit a submission file to Kaggle. Below is the ROC score. I either needed to use a different model or learn to mine the data.

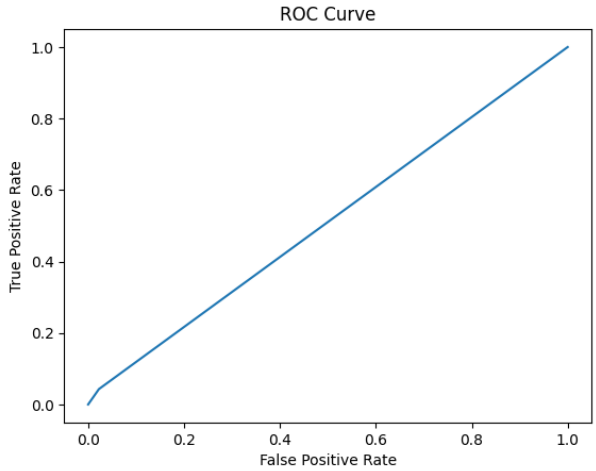

Fig. 2. KNN Classifier ROC Score

IV. Dimensionality Reduction

My first attempt in managing the dataset was to reduce the dimensionality. In the beginning, I truly assumed there was a limit to how many variables could be important. So I used Principal Component Analysis (PCA) from sklearn on each variable to get a reading on how important each one is to the whole picture.

Below is a graph of how much information we learn on the y-axis and how many variables are used on the x-axis.

This graph being a straight line is a bad thing. What I was hoping for was an ”elbow” somewhere around the 25-variable mark that shows we really only need 25 variables and the others could be discarded. We would solve the curse of dimensionality without having to learn anything.

However, this graph tells me that every variable is required, and the more I cut out, the worse my model will be. So I need a model that can handle all 200 variables.

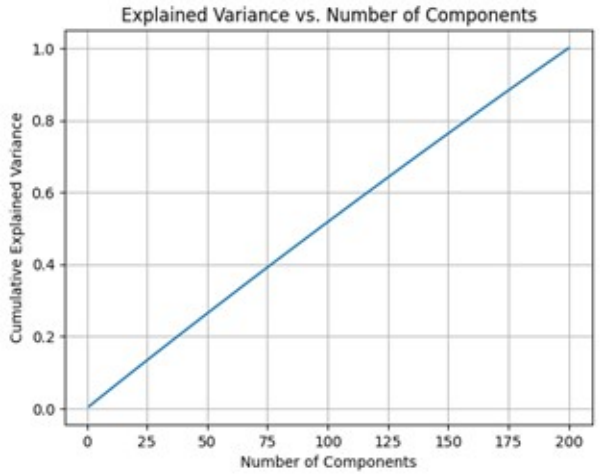

Fig. 3. PCA Explained Variance Graph

V. XGBoost (Extreme Gradient Boosting)

I chose XGBoost because it is known to be highly effective for high-dimensional datasets. I had used it before, but never on this scale of dimensionality. Given its reputation for handling such problems, I decided to give it a try. My initial attempt resulted in a 70% accuracy, which was significantly better than what I achieved with KNN. This gave me confidence that I was on the right track.

Tuning the hyperparameters, however, took more time than all previous steps combined. I used a package called BayesSearchCV, specifically designed for this purpose. In simple terms, for each parameter, I provided a range (e.g., from 1 to 10), and BayesSearchCV iteratively ran the full model to determine the best value within that range. If the returned value was at the boundary (e.g., 1 or 10), it indicated that the optimal parameter might be outside the given range. If a parameter had little effect, I simply removed it, which sped up the tuning process.

After extensive tuning, I found a set of hyperparameters that worked well, pushing the model’s accuracy up to 81%. It was amazing to see how much of a difference proper tuning could make. However, 81% was still below the defined success threshold and far from the top competition scores. I realized that XGBoost might not be able to take me further, and I needed a different approach to achieve my goal.

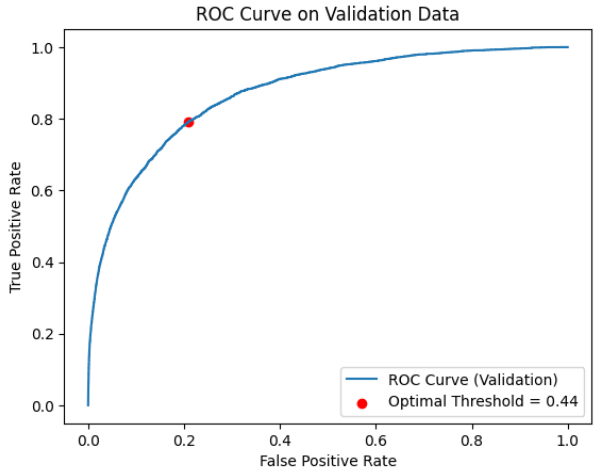

Fig. 4. XGBoost ROC Score

VI. LGBM (Light Gradient-Boosting Machine)

I found LGBM by reviewing the other Kagglers. Of the top 20 participants in this competition, all of them were using this model. It is faster, takes less memory, and, more importantly, it is better at handling large datasets. On my first attempt, I got an 80% accuracy, which completely trivialized all my work with XGBoost.

I used the same method to tune the hyperparameters by narrowing down which parameters were important and using BayesSearchCV. Finally, I achieved a 92% accuracy, which was well above my goal of 90% and was actually among the highest scores. It’s no wonder that LGBM is a popular choice for this challenge.

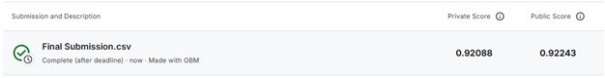

Fig. 5. LightGBM Performance

VII. My Goal vs. Milking the Model

I stopped at 92%. The top submission on Kaggle is 92.5%. Why did I stop? My goal is different from other Kagglers. I have already found a good method of making models with high dimensionality. I am not trying to win any money here, especially since the competition is over.

Many of those who scored higher than me were participating in something I call ”milking the model.” They used tactics like using specific random seeds or even had large tables of preprogrammed answers to get the edge. many kagglers referred to this as ”The Magic” In a contest where a half percent can make the difference, it is worth it to tinker for days to get the best result. I, however, consider it overfitting.

I believe my model is as good as the first-place submission. It does not use any milking techniques, and I believe it would perform just as well on Santander’s next dataset as it did on the competition set. I am proud of my model.

VIII. Conclusion

The goal of this project was not to achieve the highest score on the Kaggle leaderboard but rather to create a robust model that could effectively handle high-dimensional datasets and be useful in practical applications. The journey began with simple methods like KNN and moved on to more complex models such as XGBoost and LightGBM, with a consistent focus on finding a solution capable of managing the high dimensionality of the dataset.

By using LightGBM, I was able to achieve a 92% accuracy, which met the defined success threshold and demonstrated the effectiveness of the model in predicting customer transactions. Unlike many of the top Kaggle submissions, my model avoids overfitting tactics, such as using specific random seeds or preprogrammed answers, making it more generalizable to future datasets.

Overall, this project provided valuable insights into managing high-dimensional data, tuning hyperparameters, and building models that are both effective and generalizable. I am confident that the model's performance would hold up on new data, making it a valuable asset for Santander or similar predictive tasks.

References

[1] W. Jia, M. Sun, J. Lian, and S. Hou, "Feature dimensionality reduction: a review," *Complex & Intelligent Systems*, vol. 8, no. 3, pp. 2663-2693, 2022.

[2] H. Liang, K. Jiang, T. A. Yan, and G. H. Chen, "XGBoost: an optimal machine learning model with just structural features to discover MOF adsorbents of Xe/Kr," *ACS Omega*, vol. 6, no. 13, pp. 9066-9076, 2021.

[3] S. Song, T. Steinke, O. Thakkar, and A. Thakurta, "Evading the curse of dimensionality in unconstrained private GLMs," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 2638-2646, 2021.