# Does it worth to devote time on this customer?

Santander Customer Value Prediction Challenge

Recep Yusuf Bekci
*McGill University*
Montreal, Canada
recep.bekci@mail.mcgill.ca

***Abstract*—To stay competitive in the banking sector, the banks have to provide more special services from customer to customer. The data is taken from Kaggle challenge of Santander Group who aims using these predicted results in order to give an improved service consisting of personalization and customization. In order to conduct prediction of the value of customer transactions; neural networks, support vector regression and Gaussian processes are utilized. Following the several trials, runs and parameter tuning, the performances of the best models of each method are validated and compared as a final step. The future research directions are suggested as follow up procedures.**

***Index Terms*—gaussian processes, neural networks, prediction, support vector regression**

## I. Introduction

With the increase in choices for financial services, Santander Group wants to enhance their chances to be preferred by customers in order to maintain its competitiveness in the banking sector. At this point, the problem that we deal with tackling the provided data is the identification of the value of transactions for each potential customer. Thus, Santander Group aims using these predicted results in order to give an improved service consisting of personalization and customization. To conduct the predictions, firstly, data is analyzed in the second section and problems related to data is detected to be handled during preprocessing. In Section 3, data is scaled using the appropriate strategy and the dimension is reduced accordingly favoring the better predictions. Forth section is the main project part demonstrating the applied methods on the preprocessed data. After finalizing the runs and trials, at Section 5, the results of each method are compared and validated on differently sized data sets. The error rates of best models uploaded to Kaggle competition are revealed by explaining the possible reasons behind the results. The report is concluded with future research suggestions.

## II. Explanatory Analysis of Data

The dataset belongs to a Kaggle competition which was ended on 20 August 2018. The dataset is divided into train and test sets for the competition. The sizes are (4459, 4993) for the training set and (49342, 4992) for the test set. Obviously, we have fewer samples in our training set. Moreover, the number of columns is more than the number of rows in the training set. Our target variable is the values of transactions made by customers. So, in each row, we have some information related to each transaction. However, the column names are masked so that we do not have any idea about their nature. Density plot of our target variable can be observed from Figure 1.

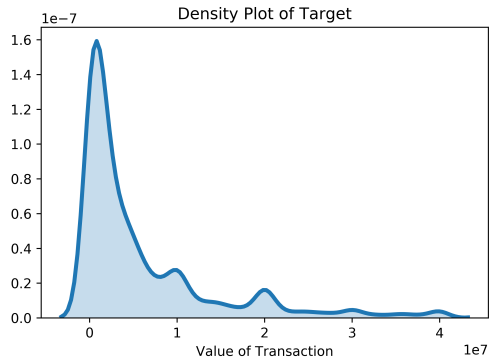


Fig. 1. Density Plot of Full Dataset Target Value

As a result, our target is negatively skewed with a reasonably long tail. In order to strengthen our analysis, we decided to use the target variable in log scale. The density of log transformation is more close to normal distribution. Figure 2 show the density of log-transformed target.

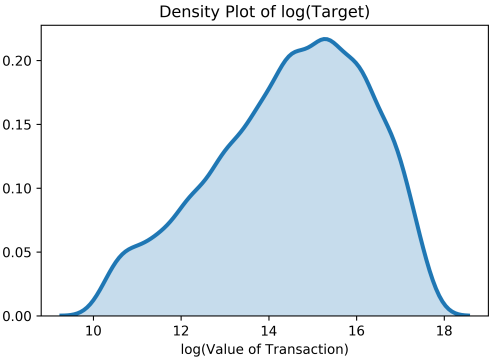


Fig. 2. Density Plot of Log Scaled Target Value

Additionally, we look for correlations between predictors. It is hard to examine due to the high number of predictors. We reduced the number with a threshold value considering the correlation value between predictors and target. So, we have the heatmap for 13 predictors on Figure 3 .

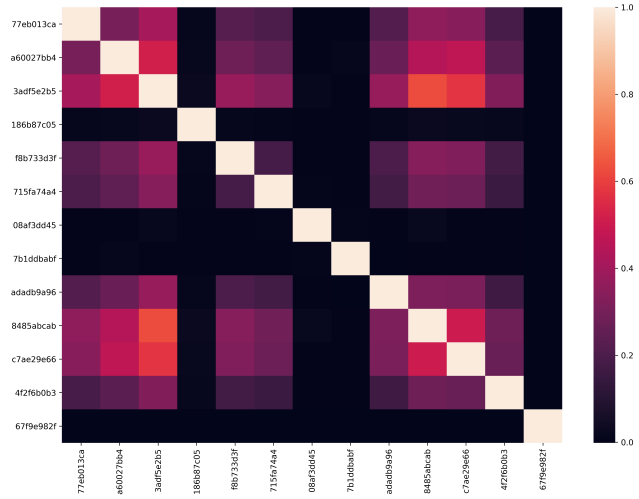


Fig. 3. Heatmap of 13 Predictors

## III. Preprocessing

One can realize from our exploratory analysis that our target variable can be best interpreted in log scale. Hence, we transformed our target variable into log scale. The performance metric used by kaggle Santander Bank competition is root mean squared logarithmic error (RMSLE) metric with the formula:

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1) - \log(a_i+1))^2} \qquad (1)$$

With
$\epsilon$ is the RMSLE value (score)
$n$ is the total number of observations in the data set
$p_i$ is prediction of target
$a_i$ is the actual target for $i$
$log(x)$ is the natural logarithm of $x$.

Thus, the performance measurement metric gets along well with the log scaled target value, and this could be a hint given by the challenge itself. Also, having a zero-intensive dataset is very problematic when considering the range of each column. We have an average range 108502230.84 which creates difficulty in running our algorithms. In order to deal with this issue, we scaled our data. One of our trials was log scale. Although log scaling is a good idea for output, it was problematic in term of optimization when we implement our algorithms. We received negative loss in our autoencoder. This was the case when we tried to apply autoencoder with our raw data. The reason behind this issue is the sparsity of the data. Log transformation does not solve the sparsity problem, hence, it was not a good choice. Secondly, we tried the most common scaling, namely normalizing with standard deviation and mean. With this scaling, we did encounter some algorithmic problems. When we examined Kaggle challenge website, for other studies in this dataset, we realized that another scaling approach is used. So, we decided to insert this scaling approach in comparison with our current one: scaling between 0 and 1. Consequently; [0,1] scaling has resulted best on the independent variables dataset. We utilized following formula:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (2)$$

Another problem in the provided data is the excessive number of features. An autoencoder is exploited to reduce the size of columns by running autoencoder for different sizes and we have tested each of the sizes using ridge regression with 0.5 regularization coefficient. The results are in the Figure 4 . According to these results, we decided to decrease feature size to 512 columns, since we acquired minimum error at this point before highly overfitting.

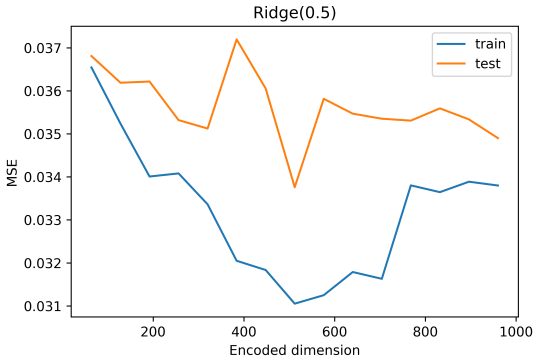


Fig. 4. Train and Test Error vs Encoded Reduced Dimensions

Figure 5 shows the zero proportion of the dataset before preprocessing. It is very difficult to get proper inference from this dataset. The dataset after preprocesing is in Figure 6 . Obviously, it becomes more convenient for our statistical learning methods. Using autoencoder, the sparsity in the real data set handled thanks to the right dimension reduction.

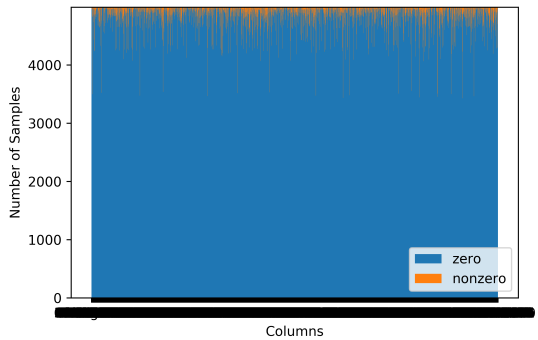


Fig. 5. Zero proportion of raw dataset

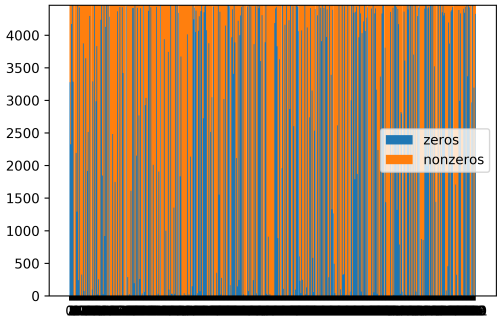


Fig. 6. Zero proportion of dataset after scaled and encoded

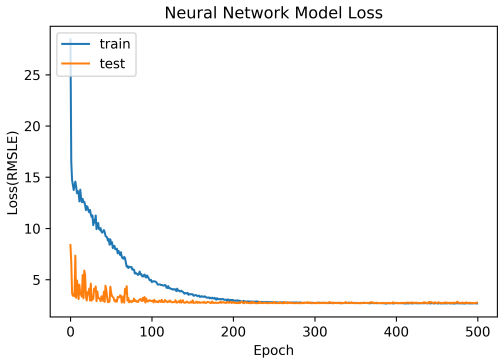


Fig. 7. Neural Network Loss Model

## IV. Prediction Methods

In this section, main part of the project which is making the accurate predictions and learning from the data is mainly conducted and clearly explained via three best models out of the space of possible models. The quality of the fit to training data, the validation data and the test data (via Kaggle challenge website) is measured using previously explained RMSLE metric.

### A. Deep Learning

First predictions acquired using neural networks. We exploited various network architectures whose training and validation performance can be seen at Table I and we decided on two hidden layers in our network using rectified linear unit (ReLU) as the activation function (defined as the positive part of its input arguments) on these two hidden layers. 400 nodes in the first layer and 41 nodes in the second layer.

We initialized weights using Glorot and Bengio (2010) also known as Xavier initialization [3] which is formulated as:

$$W \sim U[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}] \tag{3}$$

We inserted ADAM (Adaptive Moment Estimation) algorithm for optimization [5]. Which is one of the most common used Stochastic Gradient Descent algorithms in Neural Networks. To prevent overfitting and stabilize our training phase, we utilized dropout [7]. We determined the dropout probabilities using Bayesian Optimization using Python package GPyOpt. Bayesian Optimization delivered the dropout probabilities 0.586 and 0.103 for first and second hidden layer respectively.

When we use default learning rate 0.001, we observed fluctuations of the loss of test set. Then, we tried a lower learning rate 0.0001. Even the setting with lower learning rate needs more epochs to converge, it reduces the fluctuations of the loss function. However, it resulted in the same loss value.

The graph at Figure 7 shows the loss function over epochs.

### B. Support Vector Regression

Secondly, for prediction, we carried out Support Vector Regression with three different kernels: Linear, polynomial of various degrees and radial basis function. The version of Support Vector Regression we used in our study is $\epsilon$-Support Vector Regression as mentioned in Chang, Chih-Chung, and Chih-Jen Lin [2]. In a nutshell, this algorithm solves a constrained optimization problem over maximal margins. Note that, the input of this algorithm that the margins are subjected to is manipulated by kernels in terms of Sklearn package jargon. The kernel term is used for basis functions.

When we observed that RBF kernel outperforms others, we focused on tuning the model with RBF kernel. We conducted tuning processes on 5-fold cross-validation setting on a grid of hyperparameters. The results which are obtained are summarized at Figure 8 .

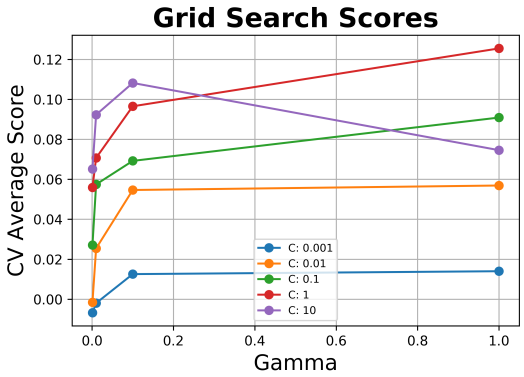


Fig. 8. Grid Search Scores

### C. Gaussian Processes

Finally, Gaussian Processes (GPs) is conducted on the data as a promising method, since it can contain different kernels together (multiplying and adding kernels) and their hyperparameters can be specified [1] [6]. Additionally, Python has a powerful tool: GaussianProcessRegressor together with gp.optimizer maximizing log-marginal-likelihood. Hence, this

TABLE I
TRAIN AND VALIDATION ERRORS FOR DIFFERENT NETWORKS

| Input | Hidden 1 | Hidden 2 | Hidden 3 | Output | Train RMSLE | Validation RMSLE |
|---|---|---|---|---|---|---|
| 512 | 41 | | | 1 | 1.61 | 1.90 |
| 512 | 400 | | | 1 | 1.49 | 1.92 |
| 512 | 4000 | | | 1 | 1.39 | 2.15 |
| 512 | 41 | 250 | | 1 | 1.42 | 1.94 |
| 512 | 41 | 400 | | 1 | 1.37 | 1.79 |
| 512 | 41 | 4000 | | 1 | 1.63 | 2.13 |
| 512 | 41 | 400 | 4000 | 1 | 1.51 | 1.91 |

allows us to estimate and repeat the processes until we reach a better result. The kernel called WhiteKernel deals with the noise in the data which is an important step during estimations.

The data obtained from the challenge was very sparse with full of zeros and huge positive numbers. As mentioned earlier, we have scaled the data and reduced the size in a manageable size. This reduction first prevented the overfitting to the dataset and handled the sparsity, since we kept the features with more meaningful content using autoencoder. In the literature, sparse gaussian processes are studied suggesting several methods in order to deal sparsity and make useful predictions. To illustrate; subset of data, pseudo-input approximations and variational approximations are the main techniques introduced throughout years. The dimension reduction resulted as a solution to sparsity and we used direct Gaussian Processes to predict target values, value of customer transactions.

GPs are parameterized by a mean function, $\mu(x)$, and a covariance function, or kernel, $K(x, x')$. They can learn the kernel parameters automatically from data, no matter how flexible we wish to make the kernel. It can also learn the regularization parameter C without cross-validation. The distinguished feature of GPs can incorporate interpretable noise models and priors over functions and can sample from prior to get intuitions about the model assumptions. We can combine automatic feature selection with learning using Automatic Relevance Determination (ARD) which finds the relevance of features by optimizing the model marginal likelihood, also known as the evidence [1].

Standard Kernels are Radial Basis Function (RBF) kernel (squared exponential kernel), Rational Quadratic kernel, Periodic kernel, and Linear Kernel. The most useful property of GPs is ability to integrate different kernels by multiplying (AND operation) and adding (OR operation) them. Constructing an additive function, we can decompose the posterior over functions into additive parts. To give an example, the popular kernel which highly used in support vector machine and GPs is the Radial Basis Function kernel, the Gaussian kernel. It catches the patterns inside the dataset thanks to its ability to conform with parameters tuning. It has the form:

$$k_{SE}(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \tag{4}$$

RBF is universal, and can be integrated into most functions that are needed. Every function in its prior has infinitely many derivatives. It also has only two parameters: The length scale $l$ determines the length of the fluctuations/extrapolations in the function. The output variance $\sigma^2$ determines the average distance of your function away from its mean. Every kernel has this parameter out as a scale factor.

In order to reach best model which gives least error among others, we have tried different combinations of kernels which deal with different patterns and signals throughout the data set. Some of the trials are can be observed from the Figure 9 and Figure 10 . After many trials and parameter tuning, the best kernel combination is selected to perform on the real test data highlighted with green colour. To illustrate, best kernel combination and its parameters on 512 dim dataset are:
Kernel= $9.44 * *2 * RationalQuadratic(alpha = 0.000994, lengthscale = 1e-05) + 0.637 * *2 * RBF(lengthscale = 1.19) + 11.3 * *2 * RBF(lengthscale = 17) + WhiteKernel(noiselevel = 0.278)$
The necessity of 2 different RBF function in order to obtain less error can be explained by taking account the content of the data. The additive kernel must be dealing with several patterns inside the data with two different length scales and output variances.

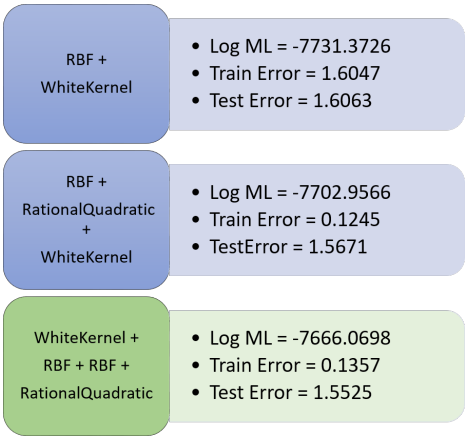


Fig. 9. A. Gaussian Processes Trials

The predicted target values plotted against validation target values together with residuals can be observed from the Figure 11 . The predicted values are accumulated around the mean of real target values (log scaled target values). The residuals of the best model is plotted in the Figure 12 .


Fig. 10. B. Gaussian Processes Trials

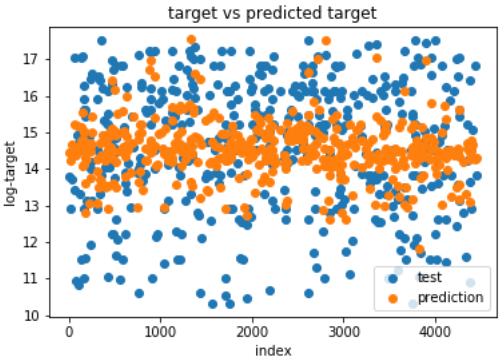


Fig. 11. Real validation data and Predictions

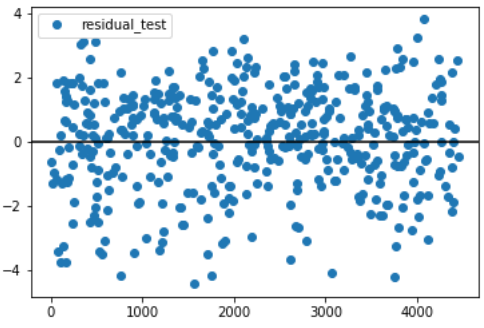


Fig. 12. Residuals of Best GPs on 512 features dataset

## V. Validation and Conclusion

To conclude, the Figure 13 summarizes the test errors (negative RMSLE) obtained by best models of each three prediction methods on the three differently reduced datasets: 256 dim, 512 dim and 1024 dim. By repeating to tune the parameters of functions, best (min RMSLE) prediction of customer transaction values are revealed by Gaussian Process with previously selected kernels on the 512 features reduced data.

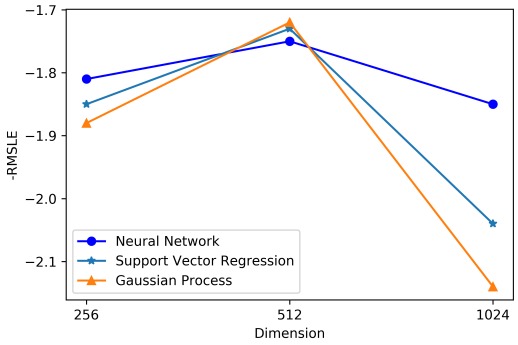


Fig. 13. Kaggle test set results

Furthermore, the important result to compare these three methods is the compilation time. Shortest execution time is acquired by SVR being 25.583148956298828 seconds. The longest execution time belongs to GPs being 307.4525640010834 seconds. The average time is obtained by NN being 169.32000708580017 seconds. Here, the trade-off is execution time versus better prediction performance.

The reasoning behind the best results carried out by GPs is due to the size of the dataset, the success of the dimension reduction strategy, model selection and excessive amount of repetition for tuning. NN will be more successful when we have a larger dataset because it is very useful for large and complex datasets. SVR can be seen as a subset of GPs, it is quicker but insufficient to predict at some point. The additive property of GPs allows the models to become better fit models and succeed the transaction values of the customers in this setting using the reduced dataset of 512 dimensions.

## VI. Future Work

As following procedures to be applied to this dataset, we suggest exploring better and complex feature selection algorithms which can be applied further in order to reduce the data size to obtain meaningful portion of columns. This resulted data may allow us to improve predictions while using the same methods. Since we obtained better results with GP, we can come up with neural networks with non-parametric activation functions corresponding to a n-layer deep Gaussian Process. At this point, the trade-off is the compilation time versus gain in terms of error minimization [1]. We suggest also to implement a combination of deep learning and Gaussian Process as suggested in the paper Huang et al. [4].

## REFERENCES


[1] Duvenaud, D. (2014). Automatic model construction with Gaussian processes (Doctoral dissertation, University of Cambridge).

[2] Chang, Chih-Chung, and Chih-Jen Lin. LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST) 2.3 (2011): 27.

[3] Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256).

[4] Huang, Gao, et al. Densely connected convolutional networks. CVPR. Vol. 1. No. 2. 2017.

[5] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[6] Rasmussen, C. E., & Williams, C. K. (2006). Gaussian process for machine learning. MIT press.

[7] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.