CSCI 567 Spring 2016 Mini-Project

Tampong Yooyen
Department of Computer Science
University of Southern California
yooyen@usc.edu

Kai-Chieh Ma
Department of Computer Science
University of Southern California
kaichiem@usc.edu

Abstract

For the mini-project, we are participating the "Santander Customer Satisfaction competition" in the Kaggle platform. We implemented a satisfaction prediction system using the anonymized features provided by the Santander Bank. We achieved a score of 0.826826 in the final test set and got the $3rd$ place among all groups in our class and $566th$ out of 5236 groups participating the competition. In this paper, we present our prediction system in details as well as the machine learning techniques we've explored.

1 Introduction

In this section, we briefly describe the data set we are processing and what models we used to predict the satisfaction level of customers. Further details of our prediction system are elaborated in the next section.

1.1 Santander Customer Satisfaction

In this competition, we are working with hundreds of anonymized features to predict if a customer is satisfied or dissatisfied with their banking experience. There are 369 features where each value measures a certain property of a customer and there are 76020 labelled training samples and 75818 unlabelled testing samples for us to predict. This is essentially a Supervised Binary Classification problem where 0 corresponds to satisfied and 1 corresponds to dissatisfied. In this particular domain, the problem can be challenging because of the credibility of questionnaire itself and the subjective level of satisfaction for different individuals.

1.2 Evaluation Metric

Each submission of the prediction result is evaluated using the metric of area under receiver operating characteristic curve (AUROC) or area under curve (AUC) in short. In statistics, a receiver operating characteristic (ROC) is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The accuracy can be expressed in terms of AUC. It's a continuous scalar value between 0.5 and 1. The higher the value the better the prediction is.

There are two different scores in this competition. One is the public score which is evaluated on one half of the testing set and is available to every one. The other is the private score which is evaluated on the other half of the testing set and is available only after the deadline of submission. The problem of overfitting may arise if we bias on the getting higher public score.

1.3 System Overview

In our system, we use decision tree ensembles model as the core for prediction and we employ gradient boosting approach to train our trees. The workflow of the system can be divided into multiple stages in the following order.

1. Feature preprocessing.
2. Feature Selection based on pearson correlation coefficient.
3. Training models with cross-validation to pick out the best hyperparameters.
4. Training the final model with best hyperparameters.
5. Make predictions from the testing data.

Further explanation for each stage is described in Sec. 3.

2 Problem Formulation

Let $\boldsymbol{x}_i$ be a feature vector of size d for the ith sample and $y_i \in \{0, 1\}$ be the label for the ith sample. We use n to denote the total number of training samples. We are interested in training an estimator $f(\boldsymbol{x}_i)$ which outputs an estimated label, $\hat{y}_i$, given $\boldsymbol{x}_i$.

Here, $\boldsymbol{x}_i$ corresponds to the feature values of a customer and $y_i \in \{0, 1\}$ corresponds to whether the customer is satisfied (0) or not (1). Based on this formulation, we describe our approach of getting the estimator $f(\boldsymbol{x}_i)$ in greater detail in the following section.

3 Methodology

3.1 Model Representation: Decision Tree Ensembles

The decision tree ensemble model is a set of classification and regression trees (CART). Mathematically, we can write our estimator model in the form

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F} \tag{1}$$

where K is the number of trees, f is a function in the functional space $\mathcal{F}$, and $\mathcal{F}$ is the set of all possible CARTs. Our objective to optimize can be further written as

$$obj(\Theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2}$$

where $l(y_i, \hat{y}_i)$ is the loss function defined as

$$l(y_i, \hat{y}_i) = \sum_{i=1}^{n} [y_i \log(1 + e^{-\hat{y}_i}) + (1 - y_i) \log(1 + e^{\hat{y}_i})] \tag{3}$$

for the case of binary classification and $\Omega(f_k)$ is the regularization term.

3.2 Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models. For the case of training decision tree ensembles, it can be expressed as a combination of additive terms. Let the prediction value at step t be $\hat{y}_i^{(t)}$, the estimator can be expressed as:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(\boldsymbol{x}_i) = \hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i) \tag{4}$$

From Eq. (2), Eq. (3) and Eq. (4), the objective at step t becomes:

$$obj^{(t)} = \sum_{i=1}^{n} [g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i)] + \Omega(f_t) \tag{5}$$

where g_i and h_i are defined as:

$$\begin{aligned} g_i &= \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{t-1}) \\ h_i &= \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{t-1}) \end{aligned} \tag{6}$$

We can see that the term only depends on g_i and h_i. Therefore we can optimize every loss function, including logistic regression, weighted logistic regression, using exactly the same solver that takes g_i and h_i as input.

3.3 Hyperparameter Optimization using Cross Validation

For hyperparameter optimization, we do an exhaustive grid search through a manually specified subset of the hyperparameter space of our system. The hyperparameter includes

- The number of features to be selected for training.
- The number of boosting iterations.
- Maximal depth of a tree.
- etc.

Cross-validation is a common model validation technique for assessing how the results of a statistical analysis will generalize to another independent data set. We use the non-exhaustive 3-fold cross-validation to pick out the best hyperparameters.

3.4 Data Exploration

In order to roughly get some insightful information from the data, the first thing we did after downloading the datasets is data exploration. We believe it helps leading us to the right direction in performing analysis. Here is what we found.

In the 76020 training samples (rows), 5900 are duplicates, and what even worst is, 109 of which have different labels (i.e. two samples that have exactly the same features can belong to different classes).

After removing those duplicates, 68365 samples are negative and 2739 are positive. Only 3.85% are positive, so we clearly have a class imbalance issue here.

Furthermore, 63 out of 369 features (columns) are also duplicates. After removing those duplicated features, 43 are floats and 263 are integers. Some have values -999999 or 9999999999, those are likely to be missing values that have been replaced with an arbitrary predefined ones.

From what we found above, the data set clearly contains a lot of noise that needs to be cleaned and preprocessed, which we will talk about in the following section.

3.5 Feature Preprocessing

Removing duplicate samples: To make the overall process simpler, we made an assumption that exact duplicates have no useful contribution to our prediction model. Therefore, for each duplication, we keep only one of them and remove the rest. Except for the case that they have different TARGET, we remove all of them to get rid of the ambiguity.

Removing redundant features: Having two identical features does not add any meaningful information to our model but some algorithms can suffer from it, so it is better to remove them in this step.

Removing constant features: A lot of features have only zeros and some features have the same value for all the samples. It is obvious that they cannot be used to distinguish the samples into two different classes.

Normalization: normalization is a common technique to eliminate the effects of certain gross influences. We tried to normalize the data but it turns out not to be so useful because we ended up using the decision tree for prediction model.

Rescaling: Some features like $var38$ have a very large range, so we tried applying log scale to them and it turns out to produce a better result.

Missing values: Values like -999999 and 9999999999 are not likely to be real values. Our assumption is they are missing values that have been replaced by an arbitrary predefined ones. For some features, the most common values are equals to the mean of remaining values, so they might also be missing values that has been replaced by the mean as well. We tried a couple of things such as treating them as NaN, or introduce another binary features to indicate that those values are missing values. However, it doesn't help much.

3.6 Feature Generation

Feature Generation is the process of taking raw, unstructured data and defining new features (i.e. variables) for potential use in our statistical analysis.

One important characteristic of the data set is that, it is sparse (i.e. having a lot of zeros). We can interpret that, for each sample, the more features that are zeros, the less information given by the sample (more unknowns).

Therefore, we introduce a new feature, "ZeroSum", that counts the number of zeros in each sample, to capture the above fact that we believe to be important.

3.7 Feature Selection

In machine learning and statistics, feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for three reasons:

- Simplification of models to make them easier to interpret by researchers/users,
- Shorter training times,
- Enhanced generalization by reducing overfitting (formally, reduction of variance)

We have tried 3 approaches as follows:

1. Calculating the PCC (Pearson Correlation Coefficient) between labels and each individual features, then pick the first N features with largest absolute PCC values. We then use cross validation to choose the best number of N. A significant improvement has been made by this method, because it helps to generalize to model.
2. Using PCA (Principal Component Analysis) to reduce feature dimensions in the hope that it might also reduce noise. However, this method doesn't help much from our experiment.
3. Calculating individual feature performance, by using only one feature at a time to train our model then perform cross validation. This method give us some insight such as "var15" is a very important feature, and we ended up using it in manual labeling step mentioned later.

3.8 Oversampling and Weighting

To mitigate the class imbalance issue (i.e. only 3.85% of samples are positive), we tried two things:

1. Oversampling the positive class to make the data set more balance.
2. Assigning more weight on the positive samples during model training.

Unfortunately, they did not have any significant impact on the performance.

3.9 Manual Labeling

From individual feature performance, we got some potential useful features. After looking closely at their characteristic, we found some interesting information that we can use to guess the interpretation of the features.

For example, looking at the distribution of the feature var15 in Figure 1.

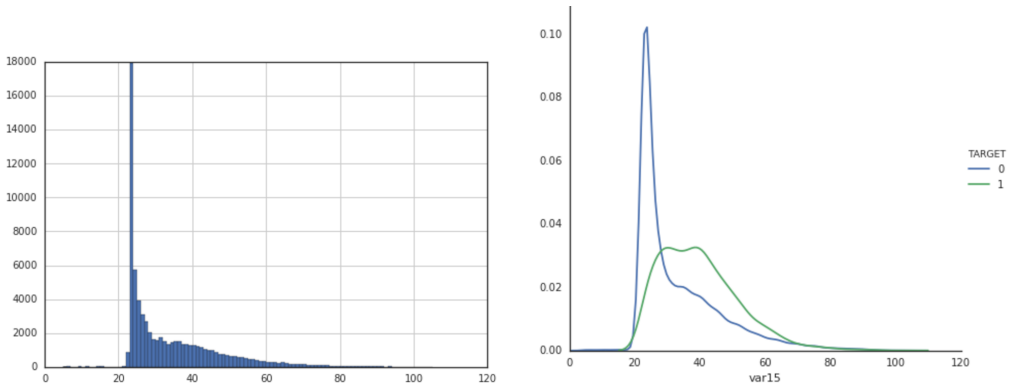

Figure 1: Left: var15 overall distribution, Right: "var15" density by label

We can see that the density of "var15" escalates high from value around 20 then declines to 0 around 80. It is possible that this "var15" variable might be the age of customers. From the age assumption, by inspecting the density of satisfaction, we found that unhappy customers tend to be older.

From this finding, we made a strong assumption that people under age 23 are always happy. Because there is a peak at value around $20-23$ for happy customers, this assumption might make sense. And this indeed improve the cross validation performance.

There are more features like this that we can use as a heuristic to predict the labels. However, this method can easily lead to overfitting and it's also risky because of misinterpretation. Therefore we don't use much of this in order to make our prediction model more general.

4 Conclusion and Possible Improvement

In the end of the competition, we achieved a score of 0.826826 in the final test set and we got the $3rd$ place among all groups in our class and $566th$ out of 5236 groups participating the competition. However, there still exists some techniques we can try to improve the performance.

- More feature generation using some nonlinear transformation $\phi(\boldsymbol{x}_i)$. This helps us to create a more complex model to fit the data, however it might also cause the problem of overfitting. Also it increases the training time exponentially if we do an exhaustive generation to the order of Q. Some techniques such as feature selection might be used in combination with this approach.

- Ensemble between multiple type of models/algorithms. Here we only use decision trees as the unit of ensemble. It might be possible that we use different models such as SVM and logistic regression to predict different values and then combine them together to generate the final prediction. The potential problem of this approach is how we decide the weights between different models.

Acknowledgments

Thanks to the course instructor, Sanjay Purushotham and the teaching assistants, Dong Guo and Omid Davtalab for teaching this course. Thanks to Chi-Hao Wu and Kuan-Wen Huang for discussing the project with us.

References

[1] Santander Customer Satisfaction Competition https://www.kaggle.com/c/santander-customer-satisfaction

[2] XGBoost Library http://xgboost.readthedocs.io/en/latest/index.html

[3] Ensembling Guide http://mlwave.com/kaggle-ensembling-guide/

[4] Complete Guide to Parameter Tuning in XGBoost http://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/