BNP Paribas Cardif Claims Management

Rong Huang(A53048013) Weiwei Li(A53107958) Xinrui Li(A53106153) Yanxing Zhang(A53100300)

1 Introduction

As a global specialist in personal insurance, BNP Paribas Cardif serves 90 million clients in 36 countries. When facing unexpected events, in order to accelerate claims management process and support clients as soon as possible, digital technology can be applied to help the company deal with all the insurance cases. There are two categories of claims: claims for which approval could be accelerated leading to faster payments and claims for which additional information is required before approval. We aimed to predict the category of a claim based on features available early in the process, then help BNP Paribas Cardif accelerate its claims process.

In this binary classficiation problem, we applied XGBoost to build boosting gradient trees. It is fast and accurate in prediction. After preprocessing data, building models and model ensembing, we get 0.45797 on the leaderboard.

2 Data

The database is from BNP Paribas Cardif, a global specialist in personal insurance. When faced with a claim, claims management requires different levels of check before a claim can be approved and a payment can be made. The goal is to prdict whether a claim process could be accelareated leading to faster payments or additional information is required based on features available early in the process. Hence, the response is a binary variable, and there are 131 predictor variables in the raw data, including 19 categorical variables and 112 numeric variables.

Table 1 Type of predictor variables

character	integer	numeric
19	4	108

The target is a binary variable, so we checked whether the data is balanced in the training set. The pie chart shows that in this case, only 24% data are classified as “1”, while 76% data are classified as “0” (Figure 1). Therefore, this dataset is imbalanced.

3 Analysis and Results

XGBoost (short for eXtreme Gradient Boosting package) is a library for boosting trees algorithms , and it is an extension of the classic gradient boosting trees model (Friedman (2002)). We chose this model because it has a much faster training speed and gets more accurate prediction.

3.1 Data proprecessing

Usually there are several issues in a real-world database. So before we analyzed the data, we fixed some potential problems and did some diagnostics.

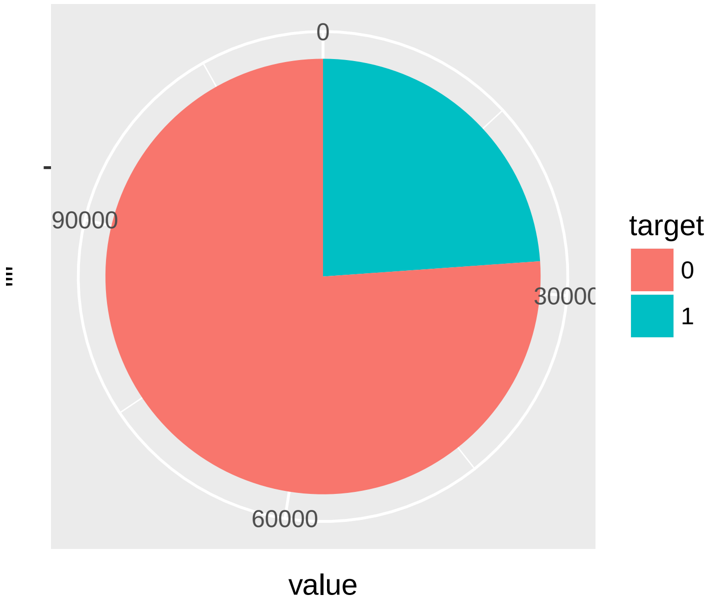

Figure 1: Target distribution

3.1.1 Missing data

We calculated the percentage of missing data. About 34% of the training data are missing, and 34% of the test data are missing. About 85% of the observations contain missing values in the training set. Therefore, it is impossible to ignore all observations with missing data.

To get a clear impression on missing data, we generated graphs to visualize the missing data (Figure 2, Figure 3). Missng data were indicated by red. Missing data are widespread across both databases.

XGBoost could handle missing values automatically. When splitting by using a feature with missing values, xgboost will assign a direction to the missing values instead of a numerical value. In other words, XGBoost guides all the data points with missing values to the left and right respectively, then choose the direction with a higher gain with regard to the objective. Hence we can just set the parameter ‘missing’ to mark the missing value label to enable this feature.

Another way is to make up missing values by imputation. We tried to replace missing values by mean. However, this increased the logloss score. So we decided to keep missing values and let XGBoost to deal with them.

3.1.2 Categorical variables

There are 19 categorical variables, and some of these include too many levels to regress properly (Figure 4). More importantly, XGBoost only deal with numerical variables. One option is to convert categorical variables to integers and treat them as numerical variables. By comparing the results, we found that after this convertion the predictive result is improved significantly.

A more appropriate method to deal with categorical variables is to introduce dummy variables, and each categorical variable with l levels needs $l-1$ dummy variables. So we need to expand the predictor matrix. Since v22 has 23419 levels, which makes it meanless after expansion, we removed this variable. Finally, we got 471 variables. With a rough model, the test logloss score decreased a bit (Figure 5). Therefore, we decided to utilize this approach. The comparision between different methods also tells us that categorical variables contain important information.

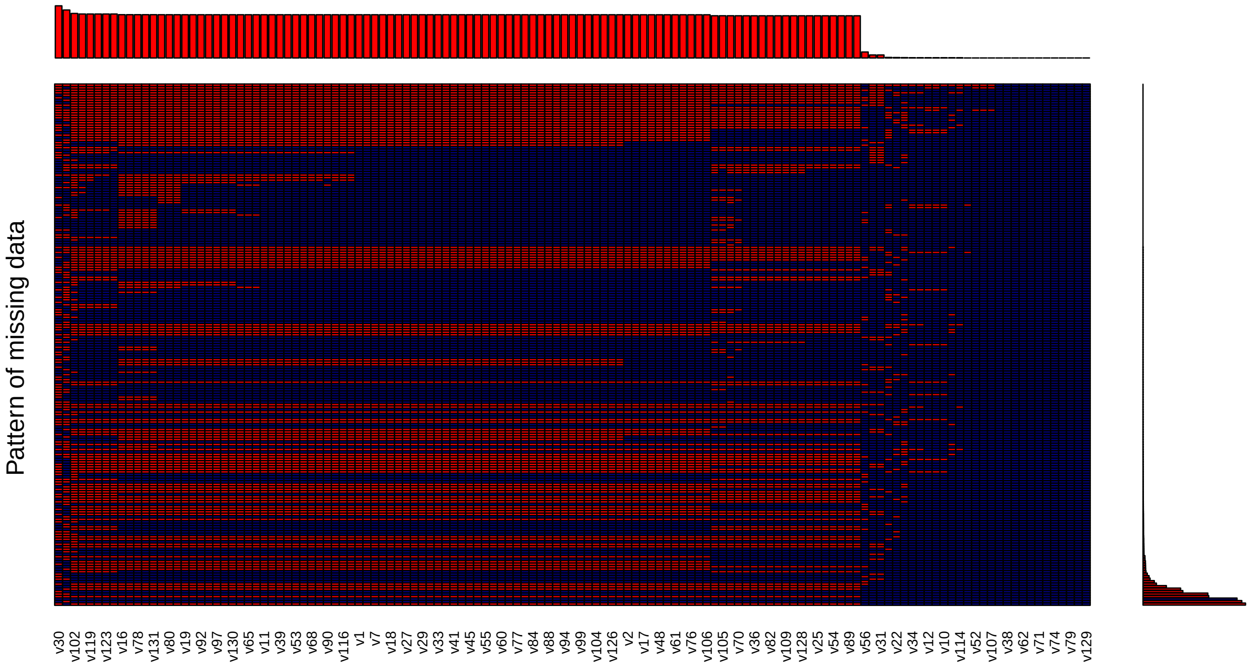

Figure 2: Pattern of missing data in the training set

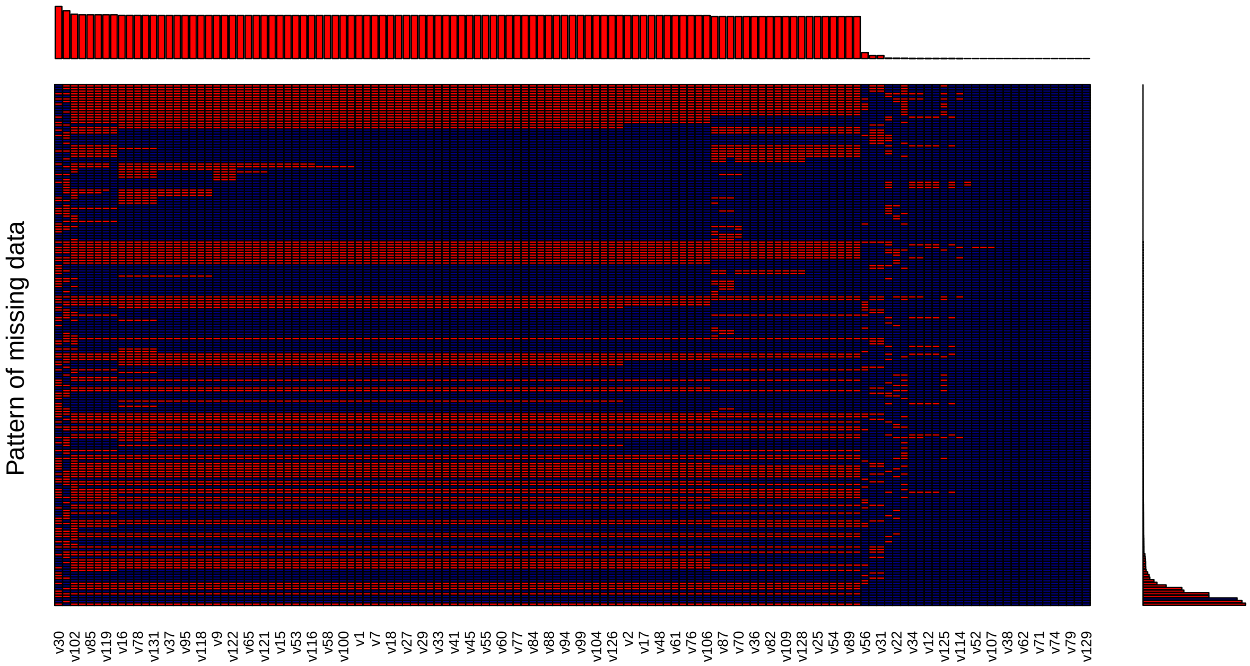

Figure 3: Pattern of missing data in the test set

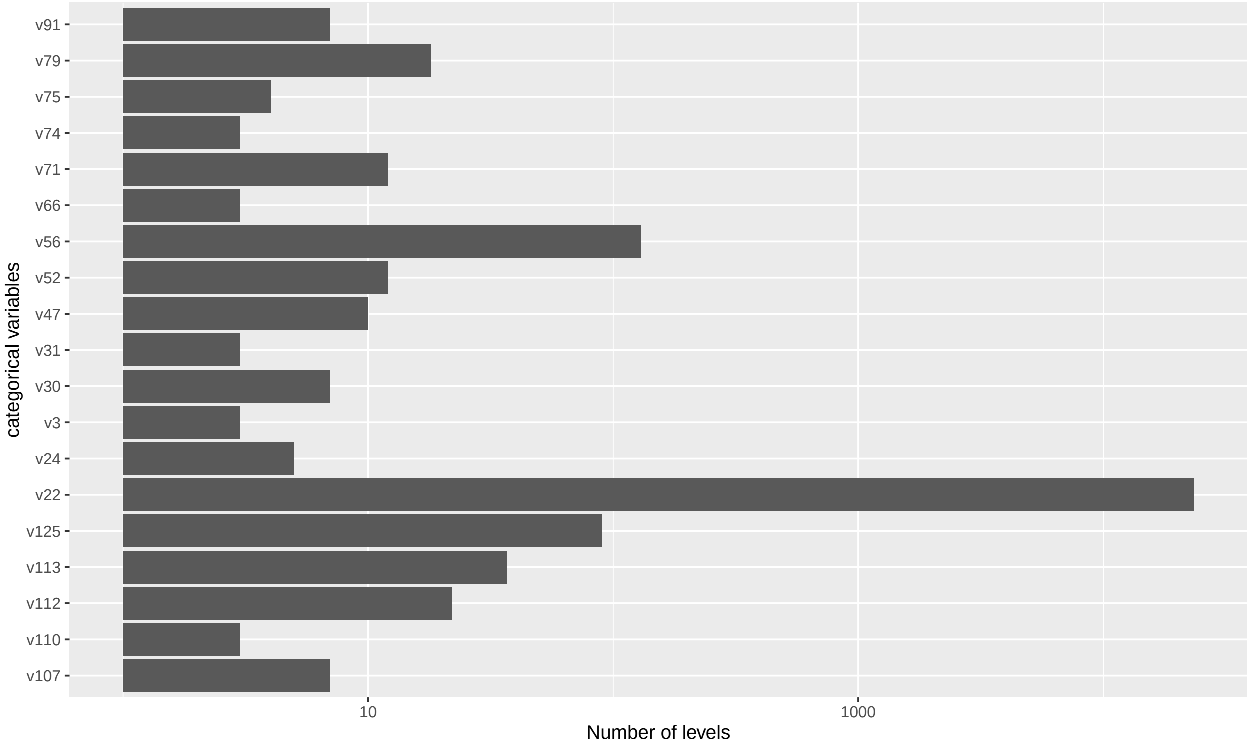

Figure 4: Number of levels in categirical variables

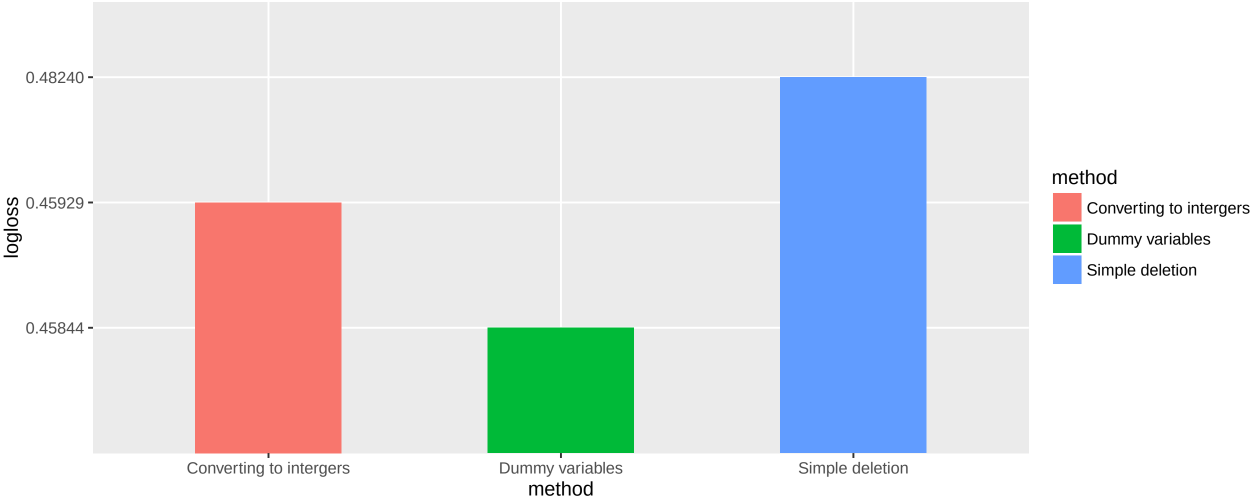

Figure 5: Methods of dealing with categotical variables

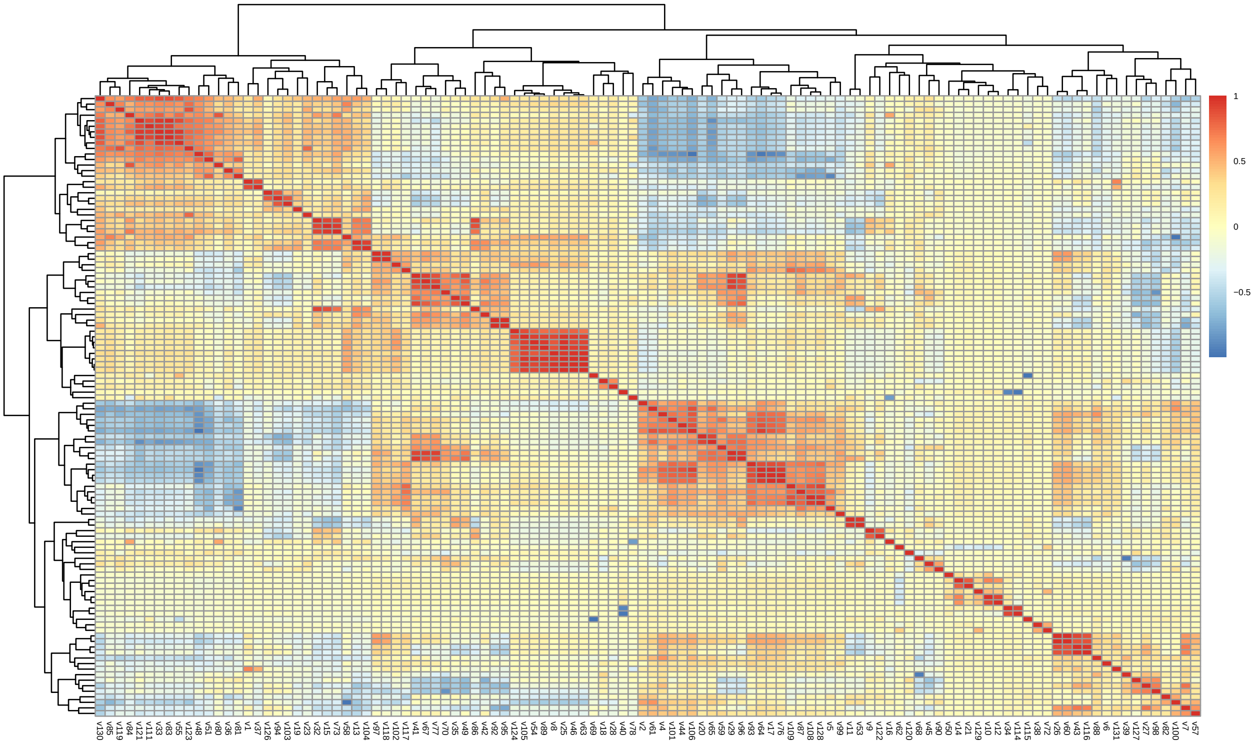

Figure 6: Correlation matrix

3.1.3 Multicollinearity analysis

Considering if the predictors are nearly linearly dependent, there will be several issues, such as difficult interpretation, increased variances of the estimates, numerically unstable fit, etc. So we checked pairwise pearson correlation and tried to remove some highly correlated variables. The heatmap shows that there are several clusters (Figure 6). For instance, v105, v54, v89, v8, v25, v46 and v63 contain almost same information. Hence we set height=0.05 as a cut-off in the hclust (Figure 7), and removed some highly correlated variables. Finally, we kept 107 variables.

However, in this case, prediction is the main goal, and the number of observation is much more than the number of variables. Hence this step didn't improve our prediction result. In addition, considering properties of boosting tree algorithm, we decided to keep all variables.

3.2 Analysis

3.2.1 Parameter tuning

XGBoost is powerful to build a model, but we need to imporve our model by parameter tuning. The main parameters we used are:

- nrounds: the number of decision trees in the final model. We used cross validation to decide this parameter.
- early.stop.round = 10: xgboost will terminate the training process if the performance is not getting better in the iteration.
- objective = "binary:logistic": the training objective.
- eval_metric = "logloss: considering the final evaluation score is logloss function.

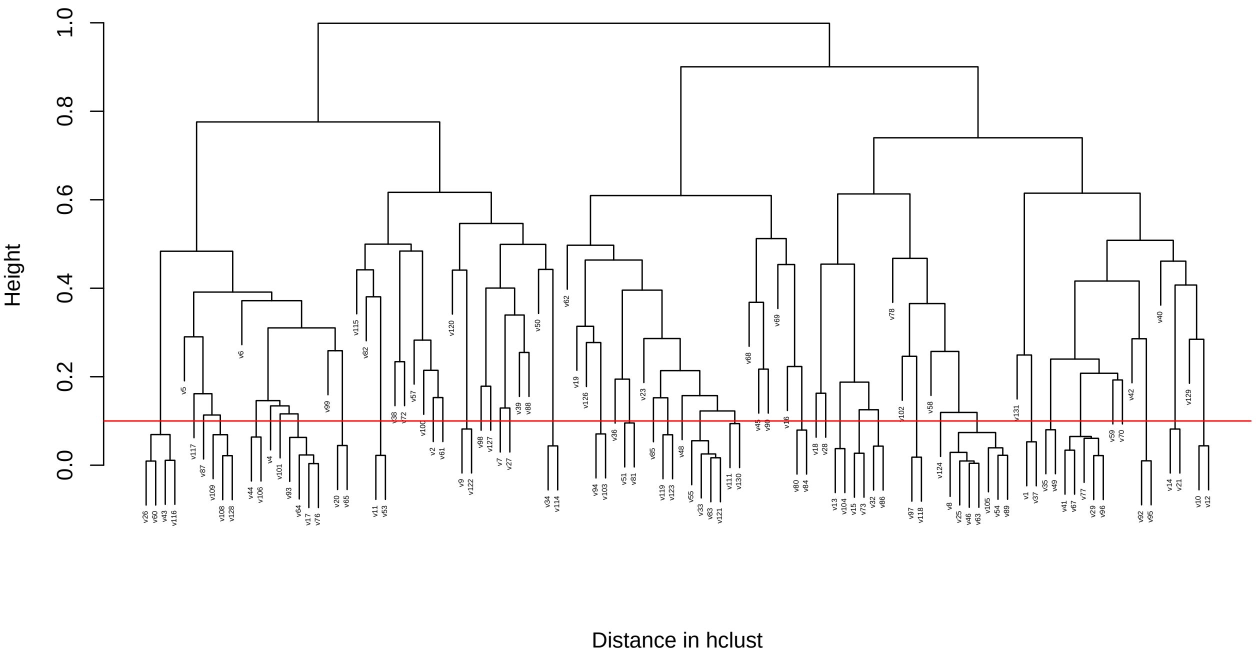

Figure 7: Cluster dendrogram

- subsample = 0.8: the fraction of observations to be randomly samples for each tree, and we set it to 0.8 to avoid overfitting.
- colsample_bytree = 0.8: the fraction of columns to be randomly samples for each tree. To get an optimal parameter for nround, we utilized cross validation to exam our model in order to avoid overfitting. The trends of logloss mean showed that after 75 rounds, even though the train-logloss-mean kept decreasing, the test-logloss mean stopped decreasing (Figure 8).

3.2.2 Model Ensembling

Empirically, ensembles tend to reduce problems related to voerfitting of the training data then yield better reuslts than a single algorithm. Therefore, we applied model ensembling for our final step. We got 10 sets of predicted probability by setting different seeds, and took average of them to get final results. After this, our prediction score decreased to 0.45797, ranking at 803 on the leaderboard.

3.3 Model inspection

3.3.1 Feature importance

Thereorically, we could plot tree models directly to visualize final models. However, in this case, the number of feartures is too large to find useful patterns by plotting. So we checked feature importance instead (Figure 9). The graph of feature importance shows that feature v50 is the most important one. Features are grouped by a K-means clustering. Feature importance gives us feature weight information but not interaction between features. But from this graph, we can tell which features are more related to the target variable.

To validate the importance of v50, we applied side-by-side boxplot. The difference in the distribution of v50 is clear by boxplot (Figure 10). Also, wilcoxon rank sum test shows that under null hypothesis, p value is less than 2.2e-16, which means there is a true location shift between two groups.

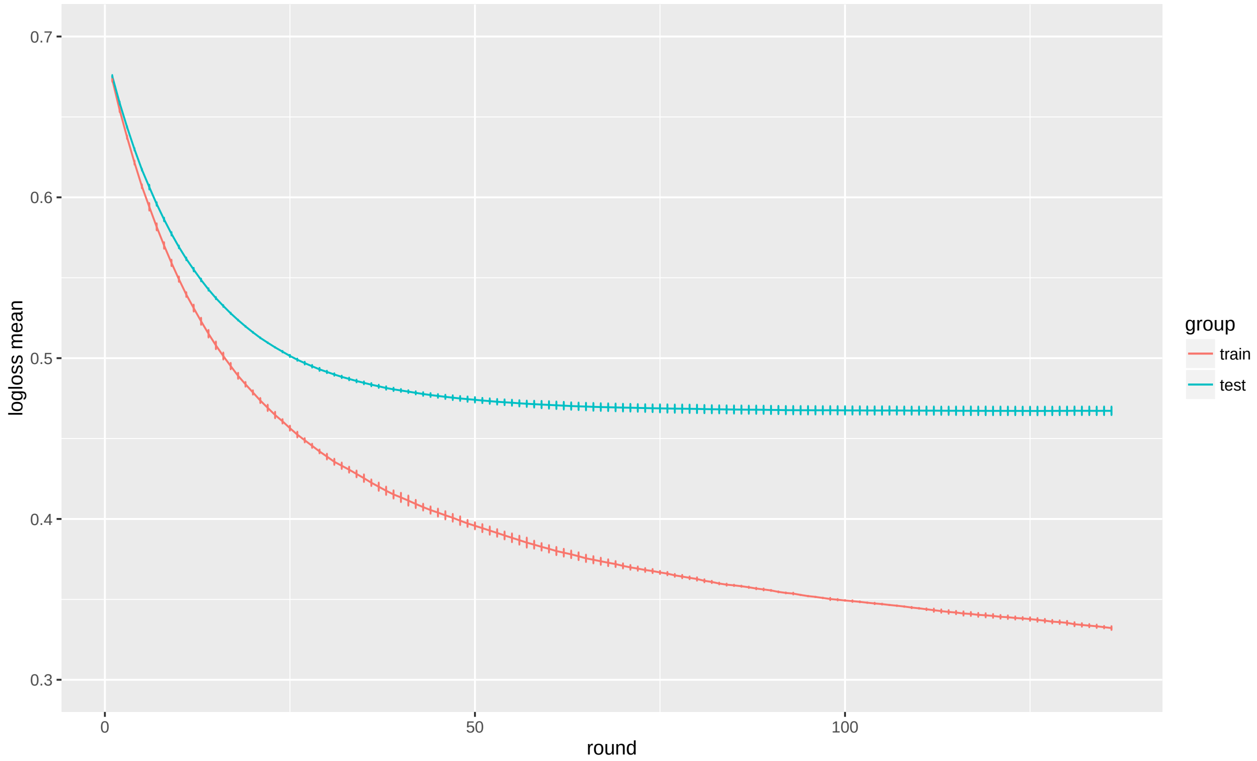

Figure 8: Logloss in cross-validation with dummy variables

```
    Wilcoxon rank sum test with continuity correction

data:  train_dat[, "v50"] and train_target
W = 8958600000, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

3.3.2 ROC

To investigate the performance of this binary classifier system, we used the receiver operating characteristic (ROC) curve, which plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings (Bradley (1997)). Since we didn't know true target values in the test data, we split the training data to get a rough ROC curve (Figure 11).

4 Code online

All codes can be found on [Github] (https://github.com/Rong0707/Math289_Final_Project).

5 Conclusions and Discussion

In order to predict the category of a claim in BNP Paribas Cardif, we applied XGBoost to build boosting tree models to predict. In this real-world case, there are lots of potential issues, such as missing data, categorical

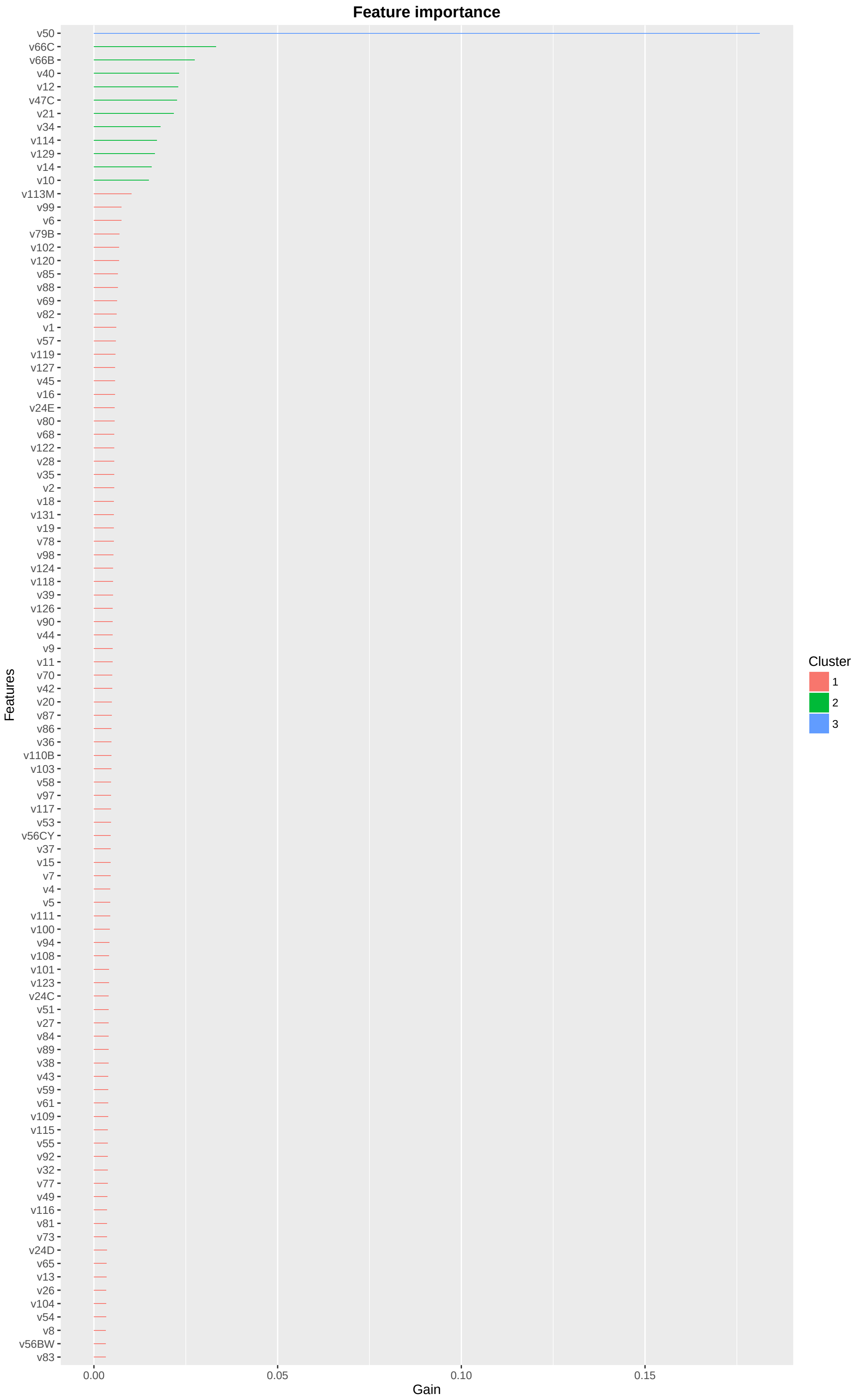

Figure 9: Feature importance

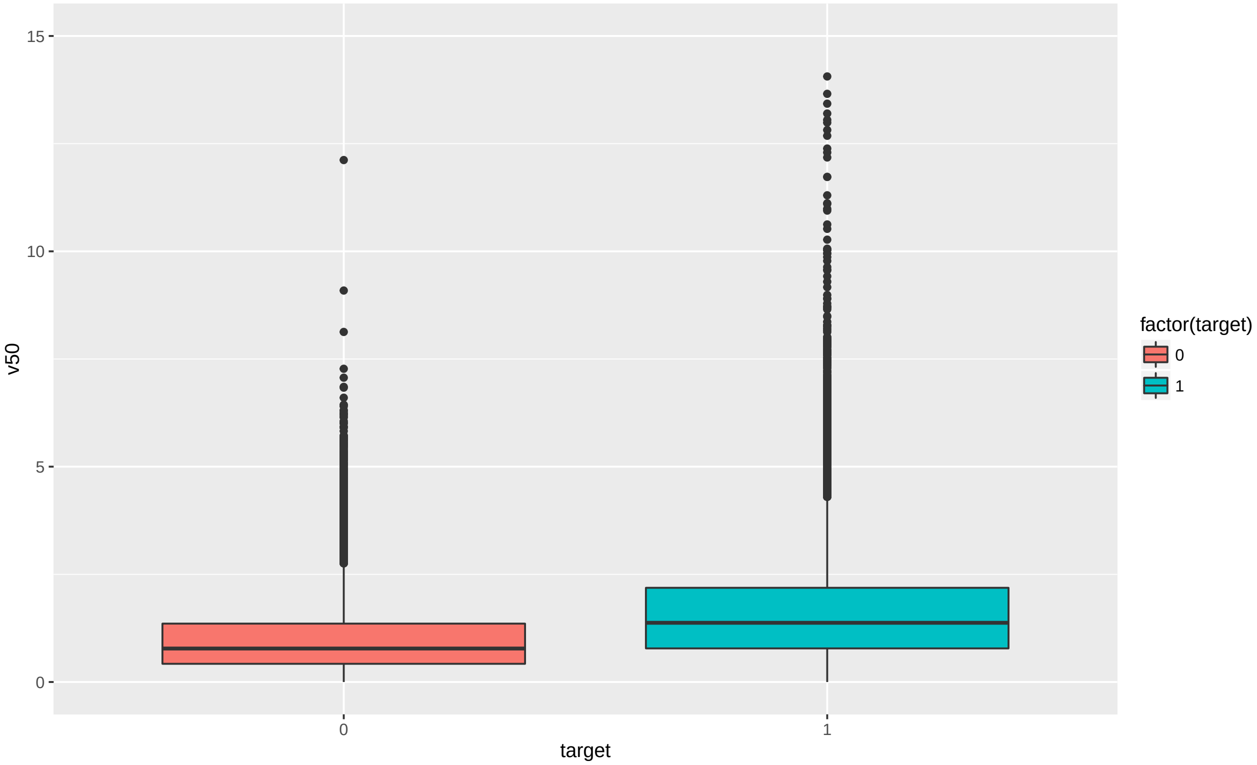

Figure 10: Distribution of v50 in two groups

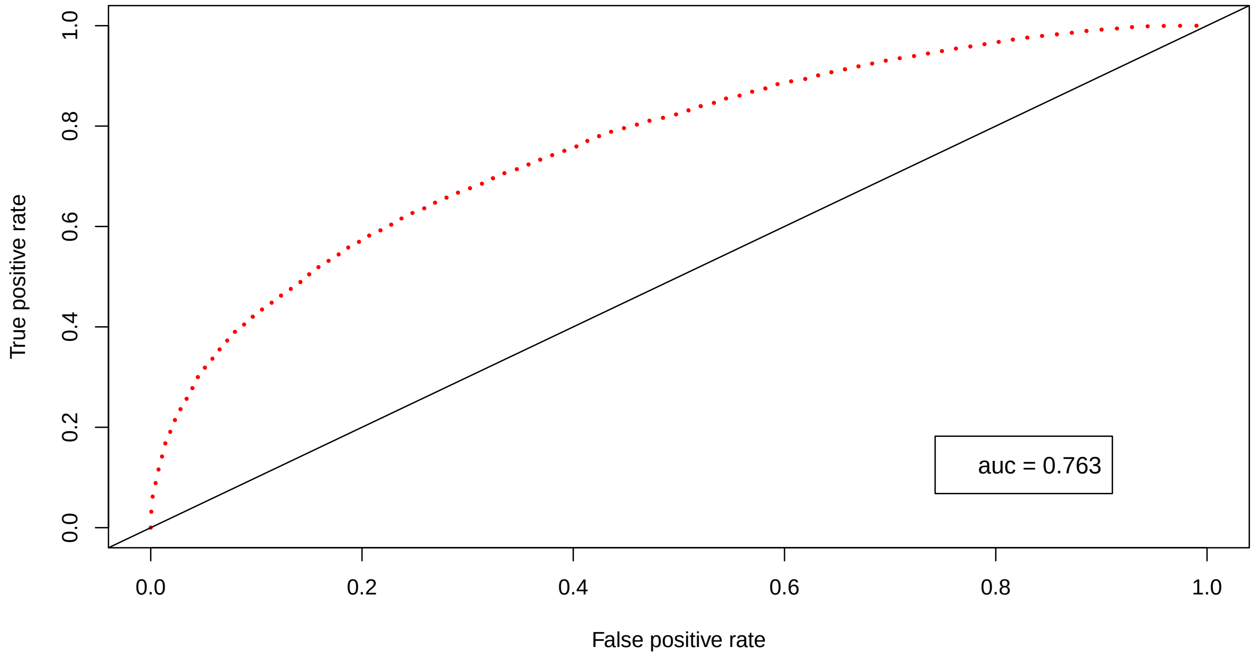

Figure 11: ROC

variables, highly correlated variables, etc. We found that if these issues were not dealt with properly, the prediction results are disasters. Therefore, it is indispensable to clean the data firstly.

There are lots of algorithms to build a binary classfier, however their speed and accuracy are quite different. We chose XGBoost to build boosting gradient trees. And it is much better than random forest model in this case. Finally, we got 0.45797 on the learderboard.

Reference

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159. doi:10.1016/s0031-3203(96)00142-2

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378. doi:10.1016/s0167-9473(01)00065-2