Imbalanced Data Classification using Sampling Techniques and XGBoost

Priyanka Lahoti
Department of Mathematics
NIT Warangal
Telangana, India

Ajeet Kumar Rai
Department of Mathematics
NIT Warangal
Telangana, India

ABSTRACT
While implementing any machine learning algorithms it is good to have the descriptive knowledge of the dataset. In any dataset, in case having more than 90% of the data in target variable is from class 1 and the remaining data is from class 2. In such type of dataset, error evaluation metric accuracy is not going to help much. Having the unknown dataset with only class 1 itself gives more than 90% accuracy, which shows accuracy as evaluation metric should be ignored. Such a problem with highly skewed target outcome is known as an Imbalanced classification problem. There is a number of techniques to deal with imbalanced dataset. In this paper, we are interested to see how sampling techniques and XGBoost can be used while working with the Imbalanced dataset.

General Terms
Machine Learning, Data Mining, Classification Problem, Sampling techniques, R programming

Keywords
Random Forest, XGBOOST, ROC curve, Anomaly detection, ROSE

1. INTRODUCTION
In manufacturing to detect defective products, failure of some computers in the computer center, fraud detection in the banking system these are the few problems which can be seen as an imbalanced classification problem. Almost every industry faces the similar types of problem. These problems can be solved if there is a method to deal with an imbalanced dataset and one of them is the sampling technique. Here using the Credit card fraud detection dataset from kaggle.com to implement the above idea in R. credit card fraud detection dataset contains 31 variables including target outcome and rows.

2. VARIOUS METHODS TO DEAL WITH IMBALANCED DATASET
Having a good amount of data without noise always helps the model to achieve better accuracy, so in case of an imbalanced dataset always look for additional data if possible. It might be helpful for skewed data. Also instead of accuracy other values like Precision, Recall, and F1 score, AUC etc. can also be used as error evaluation. Analyzing density estimation using normal distribution used in anomaly detection can also be a solution to an imbalanced dataset. K- Fold cross-validation can also be used before sampling technique. Even designing own model can also help. There are lots of other methods which can also be used, but here interested to work with only sampling technique.

3. METHODLOGY
The method are generally known as 'Sampling Techniques'. By and large, these techniques plan to adjust an imbalanced data into adjusted appropriation utilizing some component. The adjustment happens by modifying the span of unique informational collection and give a similar extent of adjust. These method have gained higher significance after numerous investigates have demonstrated that adjusted information brings about enhanced general arrangement execution contrasted with an imbalanced dataset. Thus, it's imperative to learn them.

4. SAMPLING TECHNIQUES
The following are the methods used to treat imbalanced datasets:

- Undersampling
- Oversampling
- Engineered Data Generation
- Cost-Sensitive Learning

Will comprehend only first two

4.1 UNDERSAMPLING
This method works with greater part class. It diminishes the number of perceptions from larger class to make the informational index adjusted. This method is best to utilize when the informational index is enormous and lessening the quantity of preparing tests enhances runtime and capacity inconveniences. Undersampling techniques are of 2 types: Random and Informative. Random undersampling method haphazardly picks perceptions from larger class which are wiped out until the point that the informational index gets adjusted. Informative undersampling takes after a pre-indicated choice foundation to expel the perceptions from greater part class.

4.2 OVERSAMPLING
This strategy works with minority class. It repeats the perceptions from minority class to adjust the information. It is otherwise called upsampling. Like undersampling, this strategy likewise can be separated into two sorts: Random Oversampling and informative Oversampling. Random oversampling adjusts the information by haphazardly oversampling the minority class. Informative oversampling utilizes a pre-determined model and artificially creates minority class perceptions.

5. DECISION TREE
A decision tree is a graphical portrayal of conceivable answers for a choice in light of specific conditions. It is known as a decision tree since it begins with a solitary

variable, which at that point diverges into various arrangements, much the same as a tree. A decision tree has three fundamental segments:

Root node: The best most node is called Root node. It suggests the best indicator (free factor).

Choice/Inward node: The node in which indicators (free factors) are tried and each branch speaks to a result of the test

Leaf/Terminal node: It holds a class mark (classification) - Yes or No (Last Characterization Result)..

6. RANDOM FOREST

Random Forest is Ensemble Supervised Learning Technique which uses divide and conquer to improve performance. A number of week learners (Decision Trees) are used to make a strong learner (Random Forest). It is used in both Classification and Regression problem. Random Forest builds n number of decision trees simultaneously. While building a decision tree, it randomly select some features amongst the total number of features. And while splitting, best feature is used instead of most important feature .For each randomly created decision tree, outcome for the target variable is predicted and stored. Votes for each predicted target are calculated and the one with high votes is final prediction of Random Forest.

One of the important advantage of Random forest is, it avoids over-fitting. Random forest can also be used to get variable importance plot. Random forest works well with high variance dataset.

7. XGBOOST

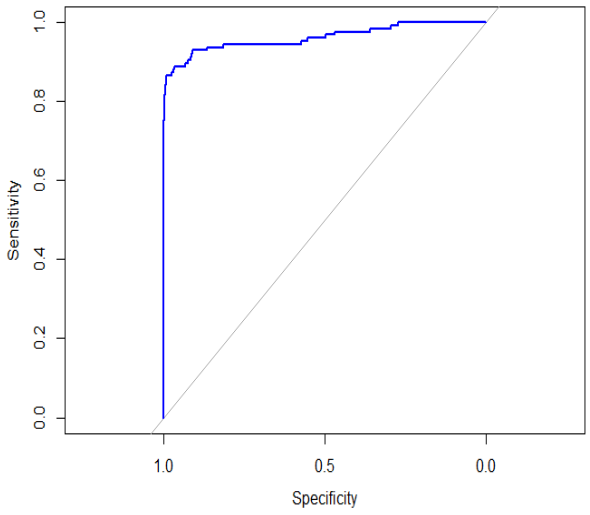
XGBoost uses Gradient tree boosting algorithm. It 1st creates one decision tree and train it. Then it records the samples for which tree makes wrong prediction. When the second tree is created, it is trained so as to predict those samples which were wrongly predicted by preceding tree and so on. Thus every tree created has better accuracy than previous one.

What makes XGBoost better than Gradient Boosting is, along with Training loss function in objective function, XGBoost have one more term called Regularization term. This regularization term prevents over-fitting of the model.

8. IMPLEMENTATION IN R

```
library(dplyr)

library(irr)

library(caret)

library(rpart)

library(rpart.plot)

library(ROCR)

library(randomForest)


#data prep
>creditcard<-read.csv("D:/creditcard.csv")
>summary(creditcard)

#There are no missing values and all variables are numeric
> table(creditcard$Class)

    0     1
284315   492

#This is highly imbalanced dataset, only 492 out of 284807 observations are fraud

#data partitioning
>set.seed(100)
>index <- sample(1:nrow(creditcard), nrow(creditcard)*0.7)
>training<- creditcard[index,]
>validation<- creditcard[-index,]
> table(training$Class)

    0     1
198998   366


# Decision Tree
# Apply decision tree model on training dataset
mod<-rpart(Class~.,data=training, method="class")

#Evaluation of model on validation dataset using area under ROC curve
predicted <- predict(mod ,validation, type="prob")
area_under_curve<-auc(validation$Class, predicted[,2])
area_under_curve
Area under the curve: 0.89


#Random Forest
#Implementing Random forest on training dataset
>n <- names(training)
>rf.form <- as.formula(paste("Class ~", paste(n[!n %in% "Class"],          collapse   = " + ")))
> trainset.rf <- randomForest(rf.form, training ,ntree=100,importance=T)

#Evaluation of model on validation dataset using area under ROC curve

> predicted0<- predict(trainset.rf ,validation, type="prob")
> area_under_curve0<-auc(validation$Class, predicted0[,2])
> area_under_curve0
Area under the curve: 0.9181

# This is imbalanced classification problem, so use function ROSE from library ROSE for data balancing

> library(ROSE)
> training$Class<-as.factor(training$Class)
> data.rose <- ROSE(Class ~ ., data = training, seed = 100)$data
> table(data.rose$Class)
   0    1
99849 99515

# Applying Decision Tree model on This balanced dataset

> mod1<-rpart(Class~.,data=data.rose, method="class")

#Accuracy on validation dataset
> predicted1 <- predict(mod1 ,validation, type="prob")
> area_under_curve1<-auc(validation$Class, predicted1[,2])
> area_under_curve1
Area under the curve: 0.9011
# AUC value for decision tree is increased from 0.89 to 0.9011

# Applying Random Forest model on This balanced data
```

```
> n <- names(data.rose)
> rf.form1 <- as.formula(paste("Class ~", paste(n[!n %in% "Class"],        collapse = " + ")))
> trainset.rf1 <- randomForest(rf.form1,data.rose,ntree=100,importance=T)

#Accuracy on validation dataset
>predicted2<- predict(trainset.rf1 ,validation, type="prob")
> area_under_curve1<-auc(validation$Class, predicted2[,2])
> area_under_curve1
Area under the curve: 0.9224
# AUC value for Random forest is increased from 0.9181 to 0.9224

#sampling
#We use function ovun.sample from library ROSE on imbalanced training dataset and take sample of 10000 observations.

> library(ROSE)
> training$Class<-as.factor(training$Class)
> data_balanced_under <- ovun.sample(Class ~ ., data = training, method = "both" ,N=10000 , seed = 300)$data
> table(data_balanced_under$Class)
   0    1
4894 5106


# Applying Decision Tree model on sampled data

> mod2<-rpart(Class~.,data=data_balanced_under, method="class")

#Accuracy on validation dataset
> predicted3 <- predict(mod2 ,validation, type="prob")
> area_under_curve3<-auc(validation$Class, predicted3[,2])
> area_under_curve3
Area under the curve: 0.9336
#AUC value on original dataset was 0.89 and is increased to 0.9336



# Applying Random Forest on This sampled data

> n <- names(data_balanced_under)
> rf.form2<- as.formula(paste("Class ~", paste(n[!n %in% "Class"], collapse = " + ")))
> trainset.rf2 <- randomForest(rf.form2, data_balanced_under,ntree=500,importance=T)

#Accuracy on validation dataset
> predicted4 <- predict(trainset.rf2 ,validation, type="prob")
> area_under_curve4<-auc(validation$Class, predicted4[,2])
> area_under_curve4
Area under the curve: 0.9647
#AUC value on original dataset was 0.9181 and is increased to 0.9647

#XGBoost

#Loading required libraries

>library(xgboost)
>library(magrittr)
>library(Matrix)

#we need to make data such that it is used in xgboost model
#training and test dataset should be in xgb.DMatrix (xgboost's own datatype)

>train_label<-training[,"Class"]
>data.train<- xgb.DMatrix(as.matrix(training[, colnames(training) != "Class"]), label = train_label)
>test_label<-validation[,"Class"]
>data.test<- xgb.DMatrix(as.matrix(validation[, colnames(validation) != "Class"]), label = test_label)

#parameters
#Here problem is classification type so objective function will be "binary:logistic", #we set evaluation metric as auc. Then train the model using xgb.train function

>n1<-length(unique(train_label))
>parameters<-list("objective"="binary:logistic","eval_metric"="auc","numclass"=n1)
>watchlist<-list(train= data.train, test= data.test)


>XGB_model<-xgb.train(params=parameters, data=data.train,   nrounds=500,
+                     watchlist=watchlist)


##training and test error plot

> error<-data.frame(XGB_model$evaluation_log)
> plot(error$iter, error$train_auc ,col='blue')


##validation of model

>predicted5 = predict(bst_model, newdata = as.matrix(validation[, colnames(validation) != "Class"], ntreelimit = bst_model$bestInd)
>library(pROC)
>area_under_curve5 = roc(validation$Class, predicted5, plot = TRUE, col = "blue")
```




```
>print(area_under_curve5)

Call:

roc.default(response = validation$Class, predictor = predicted5,     plot = TRUE, col = "blue")
```

Data: predicted5 in 85317 controls (validation$Class 0) < 126 cases (validation$Class 1).

Area under the curve: 0.9617

Table 1: Algorithms and Percentage of Accuracy

Algorithms	Training 70% Validation30%	Data balancing using ROSE function	Data balancing using ovun.sample function
Decision Tree	0.89	0.9011	0.9336
Random Forest	0.9181	0.9224	0.9647

9. CONCLUSIONS

After doing preprocessing and modelling, results for decision tree and random forest algorithm are presented in Table 1.

It can be inferred that on imbalanced dataset, before applying algorithms such as decision tree or random forest, data balancing should be done using sampling techniques, either using ROSE function or ovun.sample function from ROSE package to improve the accuracy and better performance of model.

One can use XGBoost method on such imbalanced dataset for better prediction and improved performance. Advantage of XGBoost is we're able to process ~12 million rows in a matter of minutes

10. ACKNOWLEDGMENTS

My special thanks to my parents, Prof. J. V. Ramana Murthy Sir and my friends who encouraged me to write this paper.

11. REFERENCES

[1] Cochran, W.G. (1977). Sampling Techniques. New York: Wiley.

[2] Richard G. Lyons, How Fast Must You Sample? , Test and Measurement World, November, 1988, pp. 47-57.

[3] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi, "Automatically countering imbalance and its empirical relationship to cost," Data Mining and Knowledge Discovery, vol. 17, no. 2, pp. 225–252, 2008.

[4] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in Advances in Intelligent Computing, (Hefei, China), vol. 3644, pp. 878–887, Springer-Verlag, 2005.

[5] N. Japkowicz "Learning from imbalanced data sets: A comparison of various strategies," in AAAI Workshop on Learning from Imbalanced Data Sets, (Austin, Texas), vol. 68, AAAI Press, 2000.

[6] Almuallim H., An Efficient Algorithm for Optimal Pruning of Decision Trees. Artificial Intelligence 83(2): 347-362, 1996.

[7] Breiman L., Friedman J., Olshen R., and Stone C.. Classification and Regression Trees. Wadsworth Int. Group, 1984

[8] Trevor Hastie, Rob Tibshirani, Jerome Friedman (2009) "Statistical Learning" (Springer).

[9] Leo Breiman (2001) "Random Forests" Machine Learning, 45, 5-32.

[10] Bradley, A.P., 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recogn. 30 (7), 1145–1159

[11] Saitta, L., Neri, F., 1998. Learning in the ''real world''. Mach. Learning 30, 133–163.

[12] Egan, J.P., 1975. Signal detection theory and ROC analysis, Series in Cognition and Perception. Academic Press, New York.

[13] Zhou, Z.-H. (2012). Ensemble Methods: Foundations and Algorithms. Chapman & Hall/CRC, 1st edition.

[14] Schapire, R. E., and Freund, Y. (2012). Boosting: Foundations and Algorithms. The MIT Press.