Random Forest Models on Imbalanced Data

Haoxuan Fu & Sunho Choi

Introduction

The `ImbalancedRF6520` package provides functions for multiple models of Random Forest that can be applied to Imbalanced data sets. It contains traditional models such as Balanced Random Forest, Weighted Random Forest, as well as derived methods such as Mixed Random Forest and Threshold Random Forest. This package can be used to compare the performance of the above methods.

This vignette includes 4 functions that correspond to each model.

1. `BRF` : Fits a Balanced Random Forest model
2. `WRF` : Fits a Weighted Random Forest model
3. `MixedRF`: Fits a Mixed Random Forest model that combines BRF and WRF model
4. `ThresholdRF`: Fits a Threshold Random Forest model

Installation

To install the package, you can use the following command:

```
# Install devtools if not already installed
install.packages("devtools")

# Install ImbalancedRF from GitHub
devtools::install_github("SunhoChoi-C/ImbalancedRF")
```

Then load the package

```
library(ImbalancedRF)
```

Model Introduction and Examples

1.BRF

This function utilizes the randomForest library to implement a balanced random forest model on the data. This function partitions the data into a training set and test set, then fits a balanced random forest model using the training data. Finally, it returns the fitted model and the confusion matrix of the test data.

Parameters

- `data`: A data frame that contains the data, with the response variable y being a binary variable of 0 and 1 with 1 being the minority class.
- `ntree`: The number of trees used in the balanced random forest model. The default is set to 100.
- `prob`: The ratio of training set to the whole data. The default is set to 0.8

Output The function returns a vector of length 2 with "model" being the Balanced Random Forest model fit on the data and "matrix" being the confusion matrix on the test data set.

```
set.seed(123)

data <- caret::twoClassSim(1000, intercept = -15, linearVars = 15, noiseVars = 5)
#generate sample imbalanced data set

names(data)[names(data)=="Class"]="y"
# rename the response variable as "y"

data$y = as.factor(ifelse(data$y=="Class1",0,1))
# Change the response variable value to 0,1 with 1 being the minority class

result <- ImbalancedRF::BRF(data,ntree=100,prob=0.8)

print(result$matrix)
```

Example

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 163   4
##          1  12  20
##
##                Accuracy : 0.9196
##                  95% CI : (0.8727, 0.9533)
##     No Information Rate : 0.8794
##     P-Value [Acc > NIR] : 0.04569
##
##                   Kappa : 0.6686
##
##  Mcnemar's Test P-Value : 0.08012
##
##             Sensitivity : 0.9314
##             Specificity : 0.8333
##          Pos Pred Value : 0.9760
##          Neg Pred Value : 0.6250
##              Prevalence : 0.8794
##          Detection Rate : 0.8191
##    Detection Prevalence : 0.8392
##       Balanced Accuracy : 0.8824
##
##        'Positive' Class : 0
##
```

2.WRF

This function utilizes the randomForest library to implement a weighted random forest model on the data. This function partitions the data into a training set and test set, then fits a balanced random forest model using the training data. Finally, it returns the fitted model and the confusion matrix of the test data.

Parameters

- `data`: A data frame that contains the data, with the response variable y being a binary variable of 0 and 1 with 1 being the minority class.
- `weights`: A vector that contains the weights that we want to test on. The function uses k-fold CV to find best weight.
- 'k': A parameter that indicates the number of folds to create during cross validation process. The default is set to 5.
- `ntree`: The number of trees used in the balanced random forest model. The default is set to 100.
- `prob`: The ratio of training set to the whole data. The default is set to 0.8

Output The function returns a vector of length 2 with "model" being the Weighted Random Forest model fit on the data and "matrix" being the confusion matrix on the test data set.

```
set.seed(123)

data <- caret::twoClassSim(1000, intercept = -15, linearVars = 15, noiseVars = 5)
#generate sample imbalanced data set

names(data)[names(data)=="Class"]="y"
# rename the response variable as "y"

data$y = as.factor(ifelse(data$y=="Class1",0,1))
# Change the response variable value to 0,1 with 1 being the minority class

result <- ImbalancedRF::WRF(data,weights=c(100,1000,10000,100000),k=5,ntree=100,prob=0.8)

print(result$matrix)
```

Example

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 174  18
##          1   1   6
##
##                Accuracy : 0.9045
##                  95% CI : (0.8549, 0.9415)
##     No Information Rate : 0.8794
##     P-Value [Acc > NIR] : 0.1636942
##
```

```
##                   Kappa : 0.3518
##
##  Mcnemar's Test P-Value : 0.0002419
##
##             Sensitivity : 0.9943
##             Specificity : 0.2500
##          Pos Pred Value : 0.9063
##          Neg Pred Value : 0.8571
##              Prevalence : 0.8794
##          Detection Rate : 0.8744
##    Detection Prevalence : 0.9648
##       Balanced Accuracy : 0.6221
##
##        'Positive' Class : 0
##
```

3.MixedRF

This function utilizes the randomForest library to implement a mixed random forest. The mixed random forest takes a parameter alpha and does downsampling on the original data so that the ratio is 1:alpha. Then it uses weighted random forest for the downsampled data to fit a model. Finally, it returns the fitted model and the confusion matrix of the test data.

Parameters

- `data`: A data frame that contains the data, with the response variable y being a binary variable of 0 and 1 with 1 being the minority class.
- `weights`: A vector that contains the weights that we want to test on. The function uses k-fold CV to find best weight.
- `alphas`: A vector that contains the alphas that we want to test on. The function uses k-fold CV to find the best alpha.
- 'k': A parameter that indicates the number of folds to create during cross validation process. The default is set to 5.
- `ntree`: The number of trees used in the balanced random forest model. The default is set to 100.
- `prob`: The ratio of training set to the whole data. The default is set to 0.8

Output The function returns a vector of length 2 with "model" being the Mixed Random Forest model fit on the data and "matrix" being the confusion matrix on the test data set.

```r
set.seed(123)

data <- caret::twoClassSim(1000, intercept = -15, linearVars = 15, noiseVars = 5)
#generate sample imbalanced data set

names(data)[names(data)=="Class"]="y"
# rename the response variable as "y"

data$y = as.factor(ifelse(data$y=="Class1",0,1))
# Change the response variable value to 0,1 with 1 being the minority class
```

```
result <- ImbalancedRF::MixedRF(data,weights=c(100,1000,10000,100000),alphas=c(2,3,4,5),k=5,ntree=100,p

print(result$matrix)
```

Example

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 165   8
##          1  10  16
##
##                Accuracy : 0.9095
##                  95% CI : (0.8608, 0.9455)
##     No Information Rate : 0.8794
##     P-Value [Acc > NIR] : 0.1128
##
##                   Kappa : 0.5884
##
##  Mcnemar's Test P-Value : 0.8137
##
##             Sensitivity : 0.9429
##             Specificity : 0.6667
##          Pos Pred Value : 0.9538
##          Neg Pred Value : 0.6154
##              Prevalence : 0.8794
##          Detection Rate : 0.8291
##    Detection Prevalence : 0.8693
##       Balanced Accuracy : 0.8048
##
##        'Positive' Class : 0
##
```

4.ThresholdRF

This function fits a threshold Random Forest model using the existing RandomForestSRC library. It returns the fitted model and the confusion matrix of the test data.

Parameters

- `data`: A data frame that contains the data, with the response variable y being a binary variable of 0 and 1 with 1 being the minority class.
- `ntree`: The number of trees used in the balanced random forest model. The default is set to 100.
- `prob`: The ratio of training set to the whole data. The default is set to 0.8

Output The function returns a vector of length 2 with "model" being the Weighted Random Forest model fit on the data and "matrix" being the confusion matrix on the test data set.

```r
set.seed(123)

data <- caret::twoClassSim(1000, intercept = -15, linearVars = 15, noiseVars = 5)
#generate sample imbalanced data set

names(data)[names(data)=="Class"]="y"
# rename the response variable as "y"

data$y = as.factor(ifelse(data$y=="Class1",0,1))
# Change the response variable value to 0,1 with 1 being the minority class

result <- ImbalancedRF::ThresholdRF(data,ntree=100,prob=0.8)

print(result$matrix)
```

Example

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 133   0
##          1  42  24
##
##                Accuracy : 0.7889
##                  95% CI : (0.7256, 0.8435)
##     No Information Rate : 0.8794
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : 0.433
##
##  Mcnemar's Test P-Value : 2.509e-10
##
##             Sensitivity : 0.7600
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.3636
##              Prevalence : 0.8794
##          Detection Rate : 0.6683
##    Detection Prevalence : 0.6683
##       Balanced Accuracy : 0.8800
##
##        'Positive' Class : 0
##
```

Conclusion

Generally, BRF and ThresholdRF shows good performance. The performance of WRF and MixedRF rely heavily on how precisely we can fine tune the weight and alpha parameters. However, this process takes much computing time when the data set is large. Choosing the appropriate method depending on the data set is key to high performance.