AdvanceML_Discussion_3

Ali Alghaithi

11/30/2020

Reading the data

```
library(readr)
library(dplyr)
creditcard <- read_csv("/Users/alialghaithi/Box/Advance NL Class/Discussion3/creditcard.csv")
head(creditcard)
```

```
## # A tibble: 6 x 31
##    Time     V1      V2    V3     V4      V5      V6      V7      V8     V9
##   <dbl>  <dbl>   <dbl> <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl>
## 1     0 -1.36  -0.0728 2.54   1.38  -0.338   0.462   0.240   0.0987  0.364
## 2     0  1.19   0.266  0.166  0.448  0.0600 -0.0824 -0.0788  0.0851 -0.255
## 3     1 -1.36  -1.34   1.77   0.380 -0.503   1.80    0.791   0.248  -1.51
## 4     1 -0.966 -0.185  1.79  -0.863 -0.0103  1.25    0.238   0.377  -1.39
## 5     2 -1.16   0.878  1.55   0.403 -0.407   0.0959  0.593  -0.271   0.818
## 6     2 -0.426  0.961  1.14  -0.168  0.421  -0.0297  0.476   0.260  -0.569
## # ... with 21 more variables: V10 <dbl>, V11 <dbl>, V12 <dbl>, V13 <dbl>,
## #   V14 <dbl>, V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>,
## #   V20 <dbl>, V21 <dbl>, V22 <dbl>, V23 <dbl>, V24 <dbl>, V25 <dbl>,
## #   V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl>, Class <dbl>
```

```
table(creditcard$Class)
```

```
##
##      0      1
## 284315    492
```

```
reduce_major <- creditcard %>% filter(creditcard$Class==0)
minority <- creditcard %>% filter(creditcard$Class==1)

for30 <- reduce_major[sample(nrow(reduce_major), ((nrow(minority)*100)/30)-nrow(minority)), ]
Data30 <- rbind(for30,minority)
Data30$Class <- as.factor(Data30$Class)
prop.table(table(Data30$Class))
```

```
##
##   0   1
## 0.7 0.3
```

```
for10 <- reduce_major[sample(nrow(reduce_major), ((nrow(minority)*100)/10)-nrow(minority)), ]
Data10 <- rbind(for10,minority)
Data10$Class <- as.factor(Data10$Class)
prop.table(table(Data10$Class))
```

```
##
##   0   1
## 0.9 0.1
```

```
for1 <- reduce_major[sample(nrow(reduce_major), ((nrow(minority)*100)/1)-nrow(minority)), ]
Data1 <- rbind(for1,minority)
Data1$Class <- as.factor(Data1$Class)
prop.table(table(Data1$Class))
```

```
##
##    0    1
## 0.99 0.01
```

```
# write.csv(Data1,"Data1.csv", row.names = FALSE)
# write.csv(Data10,"Data10.csv", row.names = FALSE)
# write.csv(Data30,"Data30.csv", row.names = FALSE)
```

function for measurements

```
measurements <- function(real,pred,Name) {
Confusion_Matrix<- table(real,pred,dnn = c("Real", "y_pred"))
TN = Confusion_Matrix[1,1]
FN= Confusion_Matrix[2,1]
FP= Confusion_Matrix[1,2]
TP= Confusion_Matrix[2,2]

# True Negative Rate
Acc_Negative = TN/(TN+FP)

# True Positive Rate
Acc_Positive = TP/(TP+FN)
Recall = Acc_Positive

# G-mean
G_mean = (Acc_Negative * Acc_Positive)^(1/2)

# Precision
Precision = TP/(TP+FP)

# Weighted Accuracy
Beta= 0.5 # Here we use equal weights for both true positive rate and true negative rate; i.e.,  equals
Weighted_Accuracy=  (Beta * Acc_Positive) + ((1-Beta)*Acc_Negative)

# F-measure
F_measure = (2 * Precision * Recall) /(Precision + Recall)

performance_measures <- data.frame("Method"= Name,"Acc_Positive(Recall)" =Acc_Positive, "Acc_Negative" =

return(performance_measures)

}
```

==

30% Data

Regular random forest

```
set.seed(2020)
library(randomForest)
RegularRF30 <- randomForest(Class~., data=Data30)
table(Data30$Class,RegularRF30$predicted,dnn = c("Real", "y_pred"))
```

```
##     y_pred
## Real    0    1
##    0 1139    9
##    1   56  436
```

```
measurements(Data30$Class ,RegularRF30$predicted,"RegularRF30")
```

```
##        Method Acc_Positive.Recall. Acc_Negative Precision F_measure    G_mean
## 1 RegularRF30            0.8861789    0.9921603 0.9797753 0.9306297 0.9376734
##   Weighted_Accuracy
## 1         0.9391696
```

Balanced random forest

```
set.seed(2020)
# BRF cutoff=.2
sampsize = Data30  %>%filter(Class=="1") %>% nrow()
library(randomForest)
BRF30 <- randomForest(Class~., data=Data30, importance=TRUE,sampsize=sampsize,cutoff = c(0.3, 0.7),
                           replace = T)
table(Data30$Class , BRF30$predicted,dnn = c("Real", "y_pred"))
```

```
##     y_pred
## Real    0    1
##    0 1147    1
##    1   73  419
```

```
measurements(Data30$Class ,BRF30$predicted,"BRF30")
```

```
##   Method Acc_Positive.Recall. Acc_Negative Precision F_measure    G_mean
## 1  BRF30             0.851626    0.9991289  0.997619 0.9188596 0.9224338
##   Weighted_Accuracy
## 1         0.9253775
```

==

10% Data

Regular random forest

```
set.seed(2020)
library(randomForest)
RegularRF10 <- randomForest(Class~., data=Data10)
table(Data10$Class,RegularRF10$predicted,dnn = c("Real", "y_pred"))
```

```
##     y_pred
## Real    0    1
##    0 4425    3
##    1   68  424
```

```
measurements(Data10$Class ,RegularRF10$predicted,"RegularRF10")
```

```
##        Method Acc_Positive.Recall. Acc_Negative Precision F_measure    G_mean
## 1 RegularRF10            0.8617886    0.9993225 0.9929742 0.9227421 0.9280112
##   Weighted_Accuracy
## 1         0.9305556
```

Balanced random forest

```
set.seed(2020)
# BRF cutoff=.2
sampsize = Data10  %>%filter(Class=="1") %>% nrow()
library(randomForest)
BRF10 <- randomForest(Class~., data=Data10, importance=TRUE,sampsize=sampsize,cutoff = c(0.3, 0.7),
                           replace = T)
table(Data10$Class , BRF10$predicted,dnn = c("Real", "y_pred"))
```

```
##     y_pred
## Real    0    1
##    0 4427    1
##    1   93  399
```

```
measurements(Data10$Class ,BRF10$predicted,"BRF10")
```

```
##   Method Acc_Positive.Recall. Acc_Negative Precision F_measure    G_mean
## 1  BRF10            0.8109756    0.9997742    0.9975 0.8946188 0.9004401
##   Weighted_Accuracy
## 1         0.9053749
```

Weighted random forest

==

1% Data

Regular random forest

```
set.seed(2020)
library(randomForest)
RegularRF1 <- randomForest(Class~., data=Data1)
table(Data1$Class,RegularRF1$predicted,dnn = c("Real", "y_pred"))
```

```
##      y_pred
## Real     0     1
##    0 48695    13
##    1    85   407
```

```
measurements(Data1$Class ,RegularRF1$predicted,"RegularRF1")
```

```
##       Method Acc_Positive.Recall. Acc_Negative Precision F_measure    G_mean
## 1 RegularRF1            0.8272358    0.9997331 0.9690476 0.8925439 0.9094036
##   Weighted_Accuracy
## 1         0.9134844
```

Balanced random forest

```
set.seed(2020)
# BRF cutoff=.2
sampsize = Data1  %>%filter(Class=="1") %>% nrow()
library(randomForest)
BRF1 <- randomForest(Class~., data=Data1, importance=TRUE,sampsize=sampsize,cutoff = c(0.3, 0.7),
                     replace = T)
table(Data1$Class , BRF1$predicted,dnn = c("Real", "y_pred"))
```

```
##      y_pred
## Real     0     1
##    0 48702     6
##    1   252   240
```

```
measurements(Data1$Class ,BRF1$predicted,"BRF1")
```

```
##   Method Acc_Positive.Recall. Acc_Negative Precision F_measure    G_mean
## 1   BRF1            0.4878049    0.9998768 0.9756098 0.6504065 0.6983873
##   Weighted_Accuracy
## 1         0.7438408
```

Comments

- After comparing the three methods, it shows that wghited Randomforest perforemd better than the other methods. And the balanced randomforest prfrme also better than th basic Randomforest.

this approach of tuning randomforest to help it perfor better when we have unbalance data sets can be only useful when cosdiering yousing the randomforest alograthem. however, this opns a great way of learnign how to learn to have an alograthem that performs bad wiht unblanced data sets and make it work better.

In adiiton, the more the positve class gets smaller, the more difiecult to fins the right values for tuning the cutoff and wighted class parameters.