Class Imbalance classification refers to a classification predictive modeling problem where the number of observations in the training dataset for each class is not balanced.

In other words, the class distribution is not equal or close and it is skewed into one particular class. So, the prediction model will be accurate for skewed classes and we want to predict another class then the existing model won’t be appropriate.

The imbalance problems may be due to biased sampling methods or may be due to some measurement errors or unavailability of the classes.

Let’s look at one of the datasets and how to handle the same in R.

**Load Library**

library(ROSE)

library(randomForest)

library(caret)

library(e1071)

**Getting Data**

data <- read.csv("D:/RStudio/ClassImBalance/binary.csv", header = TRUE)

str(data)

Total 400 observations and 4 variables contains in the dataset.

'data.frame': 400 obs. of 4 variables:

$ admit: int 0 1 1 1 0 1 1 0 1 0 …

$ gre : int 380 660 800 640 520 760 560 400 540 700 …

$ gpa : num 3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 …

$ rank : int 3 3 1 4 4 2 1 2 3 2 …

Let’s convert the admit variable into factor variable for further analysis.

data$admit <- as.factor(data$admit)

summary(data)

admit gre gpa rank

0:273 Min. :220.0 Min. :2.260 Min. :1.000

1:127 1st Qu.:520.0 1st Qu.:3.130 1st Qu.:2.000

Median :580.0 Median :3.395 Median :2.000

Mean :587.7 Mean :3.390 Mean :2.485

3rd Qu.:660.0 3rd Qu.:3.670 3rd Qu.:3.000

Max. :800.0 Max. :4.000 Max. :4.000

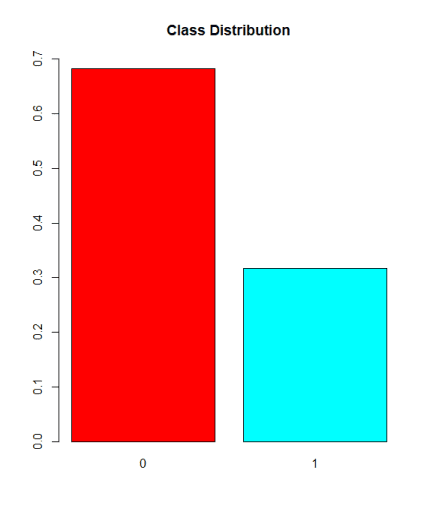
Based on summary data 273 observations pertaining to not admitted and 127 observations pertained to students admitted in the program.

**Class Imbalance**

barplot(prop.table(table(data$admit)),

        col = rainbow(2),

        ylim = c(0, 0.7),

        main = "Class Distribution")

Based on the plot it clearly evident that 70% of the data in one class and the remaining 30% in another class.

So big difference observed in the amount of data available. If we are making a model based on these a dataset accuracy predicting students not admitted will be higher compared to students who are admitted.

**Data Partition**

Lets partition the dataset into train dataset and test dataset based on set.seed.

set.seed(123)

ind <- sample(2, nrow(data), replace = TRUE, prob = c(0.7, 0.3))

train <- data[ind==1,]

test <- data[ind==2,]

**Predictive Model Data**

Let create a model based on the training dataset and look at the classification in the training dataset.

table(train$admit)

0 1

188 97

You can see that 188 observations in class 0 and 97 observations in class 1.

prop.table(table(train$admit))

Based on proportion table 65% in one class and 34% in another class.

summary(train)

admit gre gpa rank

0:188 Min. :220.0 Min. :2.260 Min. :1.000

1: 97 1st Qu.:500.0 1st Qu.:3.120 1st Qu.:2.000

Median :580.0 Median :3.400 Median :2.000

Mean :582.4 Mean :3.383 Mean :2.502

3rd Qu.:660.0 3rd Qu.:3.640 3rd Qu.:3.000

Max. :800.0 Max. :4.000 Max. :4.000

**Predictive Model**

The Random Forest model is using for prediction purposes.

rftrain <- randomForest(admit~., data = train)

Predictive Model Evaluation with test data

Let’s cross-validate based on test data. In this case, we are mentioning positive=1.

confusionMatrix(predict(rftrain, test), test$admit, positive = '1')

Confusion Matrix and Statistics

          Reference

Prediction  0  1

         0 69 22

         1 16  8

               Accuracy : 0.6696

                 95% CI : (0.5757, 0.7544)

    No Information Rate : 0.7391

    P-Value [Acc > NIR] : 0.9619

                  Kappa : 0.0839

Mcnemar's Test P-Value : 0.4173

            Sensitivity : 0.26667

            Specificity : 0.81176

         Pos Pred Value : 0.33333

         Neg Pred Value : 0.75824

            Prevalence : 0.26087

         Detection Rate : 0.06957

   Detection Prevalence : 0.20870

      Balanced Accuracy : 0.53922

       'Positive' Class : 1

Now you can see that model is around 66% and based on 95% confidence interval accuracy is lies between 57 & to 75%.

Sensitivity also is around only 30%, we can clearly mention that one of the classes is dominated over another class. Suppose if you’re interested in zero class then this model is quite a good one and if you want to predict class 1 then need to improvise the model.

**Over Sampling**

over <- ovun.sample(admit~., data = train, method = "over", N = 376)$data

table(over$admit)

0   1

188 188

This is based on resampling and now both the classes are equal.

summary(over)

admit        gre             gpa             rank

 0:188   Min.   :220.0   Min.   :2.260   Min.   :1.000

 1:188   1st Qu.:520.0   1st Qu.:3.167   1st Qu.:2.000

         Median :580.0   Median :3.450   Median :2.000

         Mean   :589.8   Mean   :3.417   Mean   :2.441

         3rd Qu.:665.0   3rd Qu.:3.650   3rd Qu.:3.000

         Max.   :800.0   Max.   :4.000   Max.   :4.000

**Random Forest Model**

rfover <- randomForest(admit~., data = over)

confusionMatrix(predict(rfover, test), test$admit, positive = '1')

Confusion Matrix and Statistics

          Reference

Prediction  0  1

         0 54 14

         1 31 16

               Accuracy : 0.6087

                 95% CI : (0.5133, 0.6984)

    No Information Rate : 0.7391

    P-Value [Acc > NIR] : 0.99922

                  Kappa : 0.1425

 Mcnemar's Test P-Value : 0.01707

            Sensitivity : 0.5333

            Specificity : 0.6353

         Pos Pred Value : 0.3404

         Neg Pred Value : 0.7941

             Prevalence : 0.2609

         Detection Rate : 0.1391

   Detection Prevalence : 0.4087

      Balanced Accuracy : 0.5843

'Positive' Class : 1

Now the accuracy is 60% and sensitivity is increased to 50%. Suppose our interest is predicting class 1 this model is much better than the previous one.

**Under Sampling**

under <- ovun.sample(admit~., data=train, method = "under", N = 194)$data

table(under$admit)

0  1

97 97

Instead of using all observations will take relevant observations from class zero respected to class1.

Suppose class 1 contain97 observations we need to take only 97 observations from class 0.

rfunder <- randomForest(admit~., data=under)

confusionMatrix(predict(rfunder, test), test$admit, positive = '1')

Confusion Matrix and Statistics

          Reference

Prediction  0  1

         0 48 11

         1 37 19

               Accuracy : 0.5826

                 95% CI : (0.487, 0.6739)

    No Information Rate : 0.7391

    P-Value [Acc > NIR] : 0.999911

                  Kappa : 0.1547

 Mcnemar's Test P-Value : 0.000308

            Sensitivity : 0.6333

            Specificity : 0.5647

         Pos Pred Value : 0.3393

         Neg Pred Value : 0.8136

             Prevalence : 0.2609

         Detection Rate : 0.1652

   Detection Prevalence : 0.4870

      Balanced Accuracy : 0.5990

       'Positive' Class : 1

Now you can see that accuracy reduced by 58% and sensitivity increased to 63%.

Under-sampling is not suggested because the number of data points less in our model and reduces the overall accuracy.

**Both (Over & Under)**

both <- ovun.sample(admit~., data=train, method = "both",

                    p = 0.5,

                    seed = 222,

                    N = 285)$data

table(both$admit)

0   1

134 151

This table is not exactly equal but similar to original situation class1 is higher than class 0.

rfboth <-randomForest(admit~., data=both)

confusionMatrix(predict(rfboth, test), test$admit, positive = '1')

Confusion Matrix and Statistics

          Reference

Prediction  0  1

         0 40  9

         1 45 21

               Accuracy : 0.5304

                 95% CI : (0.4351, 0.6241)

    No Information Rate : 0.7391

    P-Value [Acc > NIR] : 1

                  Kappa : 0.1229

 Mcnemar's Test P-Value : 1.908e-06

            Sensitivity : 0.7000

            Specificity : 0.4706

         Pos Pred Value : 0.3182

         Neg Pred Value : 0.8163

             Prevalence : 0.2609

         Detection Rate : 0.1826

   Detection Prevalence : 0.5739

      Balanced Accuracy : 0.5853

       'Positive' Class : 1

Model accuracy is 53% and sensitivity is 70%.

Now sensitivity is increased into 70% compared to previous model.

**ROSE Function**

rose <- ROSE(admit~., data = train, N = 500, seed=111)$data

table(rose$admit)

0   1

234 266

summary(rose)

When we do rose function closely watch the minimum and maximum values of each variable.

admit        gre             gpa

 0:234   Min.   :130.0   Min.   :2.186

 1:266   1st Qu.:502.9   1st Qu.:3.127

         Median :587.0   Median :3.401

         Mean   :589.7   Mean   :3.389

         3rd Qu.:684.2   3rd Qu.:3.673

         Max.   :887.2   Max.   :4.595

      rank

 Min.   :-0.6079

 1st Qu.: 1.5553

 Median : 2.3204

 Mean   : 2.3655

 3rd Qu.: 3.1457

 Max.   : 4.9871

Recollect when we created summary based on original data gre maximum value is 800 based on the rose function it’s increased into 887.

This won’t make any changes in the model but we need-aware about these changes and we can make use of this function for further analysis.

rfrose <- randomForest(admit~., data=rose)

confusionMatrix(predict(rfrose, test), test$admit, positive = '1')

Confusion Matrix and Statistics

          Reference

Prediction  0  1

         0 36 12

         1 49 18

               Accuracy : 0.4696

                 95% CI : (0.3759, 0.5649)

    No Information Rate : 0.7391

   P-Value [Acc > NIR] : 1

                  Kappa : 0.0168

 Mcnemar's Test P-Value : 4.04e-06

            Sensitivity : 0.6000

            Specificity : 0.4235

         Pos Pred Value : 0.2687

         Neg Pred Value : 0.7500

             Prevalence : 0.2609

         Detection Rate : 0.1565

   Detection Prevalence : 0.5826

      Balanced Accuracy : 0.5118

       'Positive' Class : 1

Based on this model accuracy is come down and sensitivity also reduced in this model.

But some other data set models can perform better. For getting repetitive results every time you can make use of seed function everywhere.

**Conclusions.**

Sensitivity is always closer to 100 is better, this is the way we can handle class imbalance problems efficiently & smartly.