# Project 1: SMOTE-NC

## SMOTE

```
library(readr)
library(archive)

# read data from UCI Machine Learning Repository
url_ <- "https://archive.ics.uci.edu/static/public/571/hcv+data.zip"
data <- read_csv(archive_read(url_, 1), show_col_types = FALSE)
```

```
## New names:
## * `` -> `...1`
```

```
data <- subset(data, select = -...1)
```

```
# Implement MICE, using predictive mean matching (PMM)
set.seed(321)
imputed_data <- mice(data, m=10, method = "pmm")
```

```
##
##  iter imp variable
##   1   1  ALB  ALP  ALT  CHOL  PROT
##   1   2  ALB  ALP  ALT  CHOL  PROT
##   1   3  ALB  ALP  ALT  CHOL  PROT
##   1   4  ALB  ALP  ALT  CHOL  PROT
##   1   5  ALB  ALP  ALT  CHOL  PROT
##   1   6  ALB  ALP  ALT  CHOL  PROT
##   1   7  ALB  ALP  ALT  CHOL  PROT
##   1   8  ALB  ALP  ALT  CHOL  PROT
##   1   9  ALB  ALP  ALT  CHOL  PROT
##   1   10  ALB  ALP  ALT  CHOL  PROT
##   2   1  ALB  ALP  ALT  CHOL  PROT
##   2   2  ALB  ALP  ALT  CHOL  PROT
##   2   3  ALB  ALP  ALT  CHOL  PROT
##   2   4  ALB  ALP  ALT  CHOL  PROT
##   2   5  ALB  ALP  ALT  CHOL  PROT
##   2   6  ALB  ALP  ALT  CHOL  PROT
##   2   7  ALB  ALP  ALT  CHOL  PROT
##   2   8  ALB  ALP  ALT  CHOL  PROT
##   2   9  ALB  ALP  ALT  CHOL  PROT
##   2   10  ALB  ALP  ALT  CHOL  PROT
##   3   1  ALB  ALP  ALT  CHOL  PROT
##   3   2  ALB  ALP  ALT  CHOL  PROT
##   3   3  ALB  ALP  ALT  CHOL  PROT
##   3   4  ALB  ALP  ALT  CHOL  PROT
##   3   5  ALB  ALP  ALT  CHOL  PROT
##   3   6  ALB  ALP  ALT  CHOL  PROT
##   3   7  ALB  ALP  ALT  CHOL  PROT
```

```
##   3   8  ALB  ALP  ALT  CHOL  PROT
##   3   9  ALB  ALP  ALT  CHOL  PROT
##   3   10  ALB  ALP  ALT  CHOL  PROT
##   4   1  ALB  ALP  ALT  CHOL  PROT
##   4   2  ALB  ALP  ALT  CHOL  PROT
##   4   3  ALB  ALP  ALT  CHOL  PROT
##   4   4  ALB  ALP  ALT  CHOL  PROT
##   4   5  ALB  ALP  ALT  CHOL  PROT
##   4   6  ALB  ALP  ALT  CHOL  PROT
##   4   7  ALB  ALP  ALT  CHOL  PROT
##   4   8  ALB  ALP  ALT  CHOL  PROT
##   4   9  ALB  ALP  ALT  CHOL  PROT
##   4   10  ALB  ALP  ALT  CHOL  PROT
##   5   1  ALB  ALP  ALT  CHOL  PROT
##   5   2  ALB  ALP  ALT  CHOL  PROT
##   5   3  ALB  ALP  ALT  CHOL  PROT
##   5   4  ALB  ALP  ALT  CHOL  PROT
##   5   5  ALB  ALP  ALT  CHOL  PROT
##   5   6  ALB  ALP  ALT  CHOL  PROT
##   5   7  ALB  ALP  ALT  CHOL  PROT
##   5   8  ALB  ALP  ALT  CHOL  PROT
##   5   9  ALB  ALP  ALT  CHOL  PROT
##   5   10  ALB  ALP  ALT  CHOL  PROT
```

```
## Warning: Number of logged events: 2
```

```r
# Choose which of these imputed datasets
imp.data <- complete(imputed_data,10)
```

```r
# Pre-process data
# Convert male or female data to numeric type
imp.data$Sex <- ifelse(imp.data$Sex=="m", 1, 0)
# Swap columns
imp.data <- imp.data %>% relocate(Category, Sex)

# Next, we create a 'training dataset,' on which we train or classifiers, and a
# 'testing dataset,' on which we shall test out classifiers.

library(caret)
trainIndex <- createDataPartition(imp.data$Category, p = 0.7,
                                  list = FALSE,
                                  times = 1)
# Sub-setting into training data
train <- imp.data[ trainIndex,]

# Sub-setting into testing data
test <- imp.data[-trainIndex,]

# Let us inspect these new dataframes using frequency tables
as.data.frame(table(train$Category))
```

```
##                     Var1 Freq
## 1          0=Blood Donor  374
## 2 0s=suspect Blood Donor    5
## 3            1=Hepatitis   17
## 4             2=Fibrosis   15
```

```
## 5            3=Cirrhosis   21
```

```
as.data.frame(table(test$Category))
```

```
##                     Var1 Freq
## 1          0=Blood Donor  159
## 2 0s=suspect Blood Donor    2
## 3            1=Hepatitis    7
## 4             2=Fibrosis    6
## 5            3=Cirrhosis    9
```

### Mean centre and normalise the test and train data

```
train1 <- subset(train, select = -c(Category, Sex))
test1 <- subset(test, select = -c(Category, Sex))

# Mean-centre and normalise the 11 features, doing nothing to the target
# category, of course
preProcValues <- preProcess(train1, method = c("center", "scale"))

# Now, we transform the test data set using the same transformation parameters
# that we used on the training set -- this is important.
train.transformed <- predict(preProcValues, train1)
test.transformed <- predict(preProcValues, test1)

# Recombine into full scaled datasets
train.scaled <- cbind(subset(train, select = c(Category, Sex)), train.transformed)
test.scaled <- cbind(subset(test, select = c(Category, Sex)), test.transformed)
```

```
library(tidyverse)

new.train <- train.scaled %>%
  mutate(Category = as.factor(str_replace_all(Category,
                                              "0s=suspect Blood Donor",
                                              "0=Blood Donor")))

# Let us inspect these new dataframes using frequency tables
as.data.frame(table(new.train$Category))
```

```
##           Var1 Freq
## 1 0=Blood Donor  379
## 2   1=Hepatitis   17
## 3    2=Fibrosis   15
## 4   3=Cirrhosis   21
```

```
# The categories are clearly highly imbalanced. This can lead to issues.
# To attempt to solve this, we shall use SMOTE-NC. Unfortunately, the packages
# required to do this in R have been removed from CRAN; However, we can
# simply perform this process using Python.

# Write our data into a csv file -- please adjust the file path to your desired
# location
write.csv(new.train, "/Users/thomaspagulatos/Documents/R_stuff/train.csv",
          row.names=FALSE)
```

## NOTE:

At this point, please run the file smote.py. This will create a 'zip' file of the over-sampled synthetic data called 'sm_train.zip'. Please open this file, and save the data in your desired location so that you can run the next chunk, and read the file in R.

```
# NOTE! Please adjust the file path for your use
train.smote <- read.csv("/Users/thomaspagulatos/Documents/R_stuff/sm_train.csv",
                    header=TRUE)

# We have scaled the data before over-sampling.
# Let us look at the new ratios
as.data.frame(table(train.smote$Category))
```

```
##            Var1 Freq
## 1 0=Blood Donor  432
## 2   1=Hepatitis  108
## 3    2=Fibrosis  108
## 4   3=Cirrhosis  108
```

```
# At this stage, it is useful in the future to change the datatype in Category
# from character to factor

# Furthermore, we are interested in predicting if the patient is diseased or not
# and so we split between 'Donor,' and 'Diseased,' by the latter we mean that
# the patient has a presence of liver disease.

train <- train.smote %>%
  mutate(Category = if_else(str_detect(Category, "Donor"), "Donor",
                            "Liver Disease")) %>%
  mutate(Category = factor(Category, levels = c("Liver Disease", "Donor"))) %>%
  relocate(Category, .before = Category)

test <- test.scaled %>%
  mutate(Category = if_else(str_detect(Category, "Donor"), "Donor",
                            "Liver Disease")) %>%
  mutate(Category = factor(Category, levels = c("Liver Disease", "Donor"))) %>%
  relocate(Category, .before = Category)

# Split our data into useful subsets. This is a common syntax.
x.train <- subset(train, select = -Category)
y.train <- subset(train, select = Category)
x.test <- subset(test, select = -Category)
y.test <- subset(test, select = Category)
```

```
library(caret)

# Create a random forest classifier, that uses cross-validation
train.data <- train
control <- trainControl(method="repeatedcv", number=10, repeats=3)
tunegrid <- expand.grid(.mtry=sqrt(ncol(train.data)))
model_rf <- train(Category~., data=train.data, method="rf", metric="Accuracy",
                  tuneGrid=tunegrid, trControl=control)

# View the model
print(model_rf)
```

```
## Random Forest
##
## 756 samples
##  12 predictor
##   2 classes: 'Liver Disease', 'Donor'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 681, 681, 680, 680, 680, 680, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9902981  0.9802479
##
## Tuning parameter 'mtry' was held constant at a value of 3.605551
```

```
# Test the model on unseen (scaled) test data
pred_test <- predict(model_rf, x.test)
confusionMatrix(pred_test, y.test$Category)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Liver Disease Donor
##   Liver Disease            19     2
##   Donor                     3   159
##
##                Accuracy : 0.9727
##                  95% CI : (0.9374, 0.9911)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 6.336e-06
##
##                   Kappa : 0.8683
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8636
##             Specificity : 0.9876
##          Pos Pred Value : 0.9048
##          Neg Pred Value : 0.9815
##              Prevalence : 0.1202
##          Detection Rate : 0.1038
##    Detection Prevalence : 0.1148
##       Balanced Accuracy : 0.9256
##
##        'Positive' Class : Liver Disease
##
```

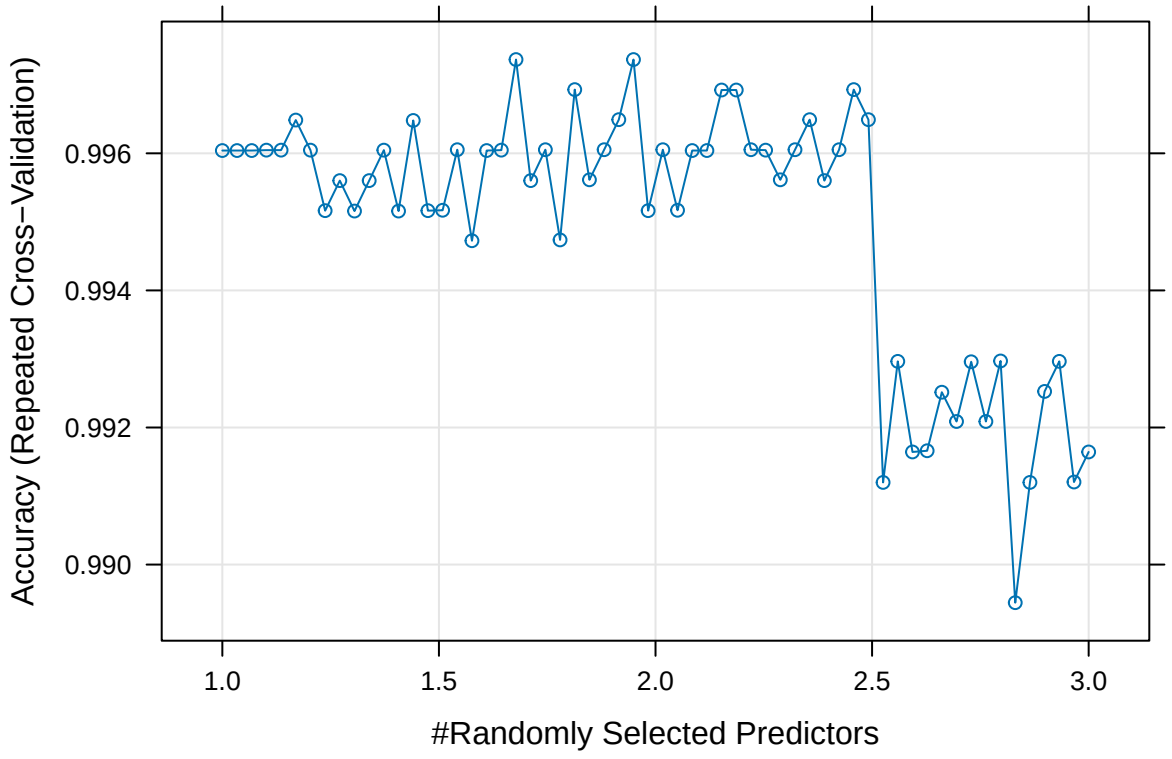
```
# Hyper-parameter optimisation
# This may take a bit of time to run

set.seed(1)
tunegrid <- expand.grid(.mtry=seq(1, 3, length = 60))
model_rf2 <- train(Category~., data=train.data, method="rf", metric="Accuracy",
```

```
                  tuneGrid=tunegrid, trControl=control)
model_rf2
```

```
## Random Forest
##
## 756 samples
##  12 predictor
##   2 classes: 'Liver Disease', 'Donor'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 681, 680, 681, 680, 680, 681, ...
## Resampling results across tuning parameters:
##
##   mtry      Accuracy   Kappa
##   1.000000  0.9960408  0.9919227
##   1.033898  0.9960408  0.9919227
##   1.067797  0.9960408  0.9919227
##   1.101695  0.9960466  0.9919390
##   1.135593  0.9960466  0.9919467
##   1.169492  0.9964852  0.9928425
##   1.203390  0.9960466  0.9919390
##   1.237288  0.9951636  0.9901311
##   1.271186  0.9956022  0.9910269
##   1.305085  0.9951577  0.9901149
##   1.338983  0.9956022  0.9910269
##   1.372881  0.9960466  0.9919592
##   1.406780  0.9951577  0.9901072
##   1.440678  0.9964794  0.9928262
##   1.474576  0.9951636  0.9901235
##   1.508475  0.9951694  0.9901824
##   1.542373  0.9960525  0.9919755
##   1.576271  0.9947250  0.9892651
##   1.610169  0.9960408  0.9919502
##   1.644068  0.9960466  0.9919678
##   1.677966  0.9973683  0.9946503
##   1.711864  0.9956022  0.9910544
##   1.745763  0.9960525  0.9919830
##   1.779661  0.9947367  0.9893082
##   1.813559  0.9969297  0.9937621
##   1.847458  0.9956139  0.9910810
##   1.881356  0.9960525  0.9919830
##   1.915254  0.9964911  0.9928650
##   1.949153  0.9973683  0.9946503
##   1.983051  0.9951636  0.9901724
##   2.016949  0.9960525  0.9919893
##   2.050847  0.9951694  0.9901825
##   2.084746  0.9960408  0.9919449
##   2.118644  0.9960408  0.9919439
##   2.152542  0.9969238  0.9937455
##   2.186441  0.9969238  0.9937455
##   2.220339  0.9960525  0.9919830
##   2.254237  0.9960466  0.9919602
##   2.288136  0.9956139  0.9910810
```

```
##    2.322034  0.9960525  0.9919818
##    2.355932  0.9964911  0.9928650
##    2.389831  0.9956022  0.9910567
##    2.423729  0.9960525  0.9919830
##    2.457627  0.9969297  0.9937621
##    2.491525  0.9964911  0.9928663
##    2.525424  0.9911987  0.9821010
##    2.559322  0.9929648  0.9857057
##    2.593220  0.9916431  0.9830156
##    2.627119  0.9916605  0.9830603
##    2.661017  0.9925145  0.9847633
##    2.694915  0.9920876  0.9839116
##    2.728814  0.9929589  0.9856671
##    2.762712  0.9920876  0.9839179
##    2.796610  0.9929706  0.9857345
##    2.830508  0.9894441  0.9785385
##    2.864407  0.9911987  0.9821035
##    2.898305  0.9925262  0.9848037
##    2.932203  0.9929648  0.9857183
##    2.966102  0.9912045  0.9821260
##    3.000000  0.9916431  0.9830005
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 1.677966.
```

```r
# Plot optimised model
plot(model_rf2)
```




```r
# Check the performance of the model
pred_test0 <- predict(model_rf2, x.test)
confusionMatrix(pred_test0, y.test$Category)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Liver Disease Donor
##   Liver Disease            19     1
##   Donor                     3   160
##
##                Accuracy : 0.9781
##                  95% CI : (0.945, 0.994)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1.235e-06
##
##                   Kappa : 0.8924
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.8636
##             Specificity : 0.9938
##          Pos Pred Value : 0.9500
##          Neg Pred Value : 0.9816
##              Prevalence : 0.1202
##          Detection Rate : 0.1038
##    Detection Prevalence : 0.1093
##       Balanced Accuracy : 0.9287
##
##        'Positive' Class : Liver Disease
##
```

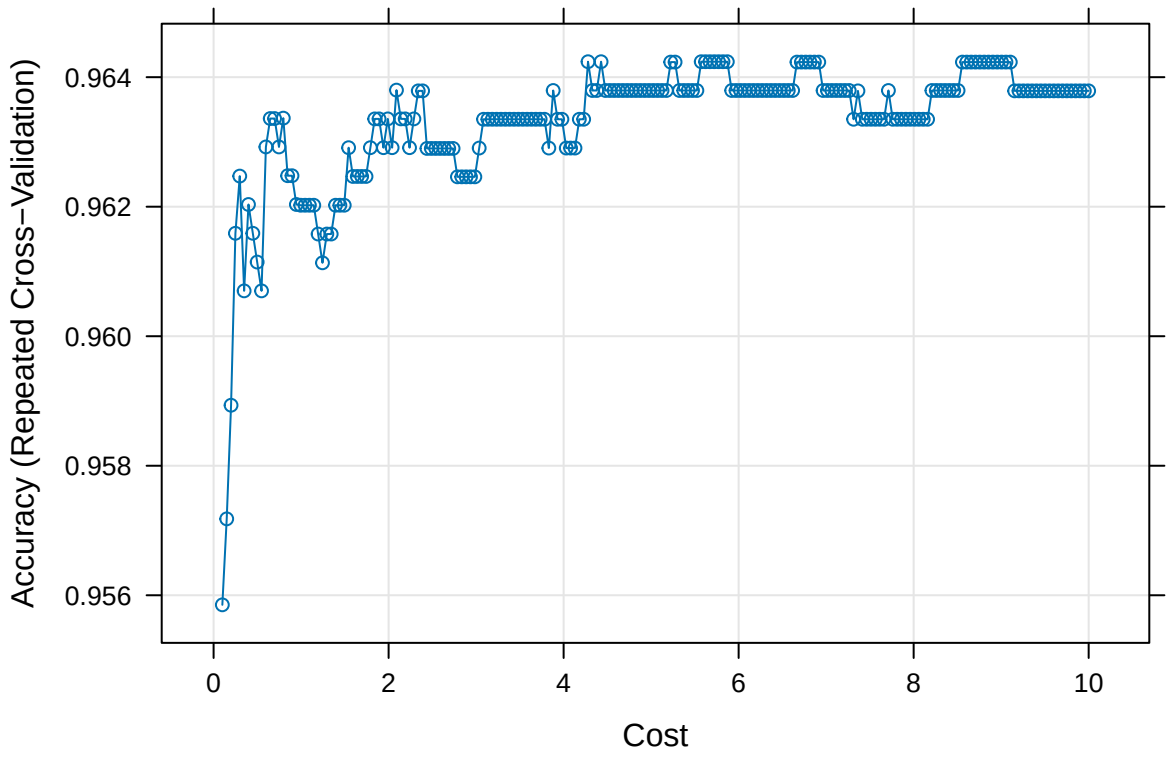
```
train_control <- trainControl(method="repeatedcv", number=10, repeats=3)

# Fit the model x
svm1 <- train(Category ~., data = train.data, method = "svmLinear", trControl = train_control)
#View the model
svm1
```

```
## Support Vector Machines with Linear Kernel
##
## 756 samples
##  12 predictor
##   2 classes: 'Liver Disease', 'Donor'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 681, 680, 680, 681, 680, 680, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9612216  0.9209226
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
# Check the performance of the model
pred_test1 <- predict(svm1, x.test)
confusionMatrix(pred_test1, y.test$Category)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Liver Disease Donor
##   Liver Disease            17     3
##   Donor                     5   158
##
##                Accuracy : 0.9563
##                  95% CI : (0.9157, 0.9809)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 0.0003133
##
##                   Kappa : 0.7849
##
##  Mcnemar's Test P-Value : 0.7236736
##
##             Sensitivity : 0.7727
##             Specificity : 0.9814
##          Pos Pred Value : 0.8500
##          Neg Pred Value : 0.9693
##              Prevalence : 0.1202
##          Detection Rate : 0.0929
##    Detection Prevalence : 0.1093
##       Balanced Accuracy : 0.8770
##
##        'Positive' Class : Liver Disease
##
```

```r
# Hyperparameter optimisation
svm2 <- train(Category ~., data = train.data, method = "svmLinear",
              trControl = train_control,
              tuneGrid = expand.grid(C = seq(0.1, 10, length = 200)))
#View the model
#svm2

plot(svm2)
```




```
svm2$bestTune
```

```
##           C
## 85 4.278894
```

```
# Check the performance of the model
pred_test1 <- predict(svm2, x.test)
confusionMatrix(pred_test1, y.test$Category)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Liver Disease Donor
##   Liver Disease            15     4
##   Donor                     7   157
##
##                Accuracy : 0.9399
##                  95% CI : (0.895, 0.9696)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 0.005158
##
##                   Kappa : 0.6981
##
##  Mcnemar's Test P-Value : 0.546494
##
##             Sensitivity : 0.68182
##             Specificity : 0.97516
##          Pos Pred Value : 0.78947
##          Neg Pred Value : 0.95732
##              Prevalence : 0.12022
##          Detection Rate : 0.08197
```

```
##    Detection Prevalence : 0.10383
##       Balanced Accuracy : 0.82849
##
##        'Positive' Class : Liver Disease
##
```