National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Simitha Sitaram Shetty
Student ID: X18134980

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Simitha Sitaram Shetty
Student ID:	X18134980
Programme:	Data Analytics
Year:	2019-2020
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	12/12/2019
Project Title:	Configuration Manual
Word Count:	903
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	11th December 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Simitha Sitaram Shetty
X18134980

The configuration manual gives an overview of the hardware and software requirements used to run the code for Pre-processing, Transformation and Implementation of models in the research study : "Breast Cancer Analysis and Prognosis Using Machine Learning".

1 Hardware/Software Requirements :

1.1 Hardware

- Processor : Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80GHz
- Installed Memory : 8:00 RAM
- Storage : 1 TB 5400 rpm SATA SSHD
- Operating System : Windows 10, 64-bit

1.2 Software

- RStudio : R programing softwrare with IDE Rstudio is used for data cleaning, processing, tranformations and implementation of all the models.
- Microsoft Excel : Used for saving data and creating graphs to analyze the performance of the techniques.
- Power BI : for creating bar and line graphs
- Draw.io : Creation of methodology diagram and implementation framework.

2 Data Preparation

Data Preparation includes data cleaning, transforming and selecting necessary features out of all the attributes. Two datasets are used for the study i.e. the 1st dataset is taken from the data.world[1] and the 2nd data is taken from kaggle[2]. These datasets gives information about the breast mass and breast tissue. Firstly the datasets are checked if any null value or unwanted data exists. The 1st dataset did not contain any null values whereas the 2nd dataset had special characters denoted by "?" which was replaced by a

[1]https://data.world/health/breast-cancer-wisconsin
[2]https://www.kaggle.com/roustekbio/breast-cancer-csv

null value first and later was replaced with the mean of that column as shown in Figure 1 and is performed using R programming.

```
#Loading the Dataset
test <- read.csv("C:/Users/Simitha Shetty/Desktop/Research Preoject/Week 2/Coding/BreastCancer.csv")

#Converting the dataframe to matrix and then back to dataframe (gsub replaces all the occurences)
t <- as.data.frame(gsub("[[:punct:]]", "", as.matrix(test)))
write.csv(t, file = "C:/Users/Simitha Shetty/Desktop/Research Preoject/Week 2/Coding/BreastCancerFinal.csv",row.names = FALSE
          , na = "")

Load <- read.csv("C:/Users/Simitha Shetty/Desktop/Research Preoject/Week 2/Coding/BreastCancerFinal.csv")
missmap(Load)
#Number of missing values
sum(is.na(Load))
#Column with missing value
colSums(is.na(Load))
mean = mean(Load$bare_nucleoli, na.rm = TRUE)
#Replacing missing value with the mean of that column i.e. 3.545
Load$bare_nucleoli[is.na(Load$bare_nucleoli)]= '3.545'
sum(is.na(Load))
View(Load)
write.csv(Load, file = "C:/Users/Simitha Shetty/Desktop/Research Preoject/Week 2/Coding/BreastCancerFinalFinal.csv",row.names
          )
```

Figure 1: Cleaning Code - Dataset 2

As the 1st Dataset contains 32 attributes in total including the target variable, RFE feature selection algorithm is applied and the important features are displayed based on the Root Mean Square Error value. Figure 2 shows the code that is applied to select important features. The rfeControl() contains lots of pre-defined functions of which the rfFuncs i.e. random forest function is used.

```
################################################################################################ RFE ##################
#Load the data
dataset <- read.csv("C:/Users/Simitha Shetty/Desktop/Research Preoject/Week 2/Coding/WisconsinDataset.csv")
set.seed(300)

#loading libraries
#install.packages("mlbench")
library(mlbench)
library(caret)

# defining the control using a random forest selection function rfFuncs
control <- rfeControl(functions=rfFuncs, method="cv", number=20)

#running the RFE algorithm
results <- rfe(dataset[,1:30], dataset[,31], sizes=c(1:30), rfeControl=control)

# summarizing the output
print(results)

# listing the chosen features
predictors(results)

# plot the solution
plot(results, type=c("g", "o"))
```

Figure 2: Feature Selection Code - Dataset 1

3 Implementation Codes for Machine Learning Algorithms

In this study, seven machine learning models are applied i.e. Support Vector Machine(SVM), Naive Bayes(NB), K-Nearest Neighbour(KNN), Logistic Regression (LR), Classification and Regression Tree(CART), Artificial Neural Network(ANN), Extreme

Gradient Boosting(Xgboost). 10-fold cross validation is used in order to evaluate the performance of the models. All the coding is done using the R programming language. Before applying the algorithms, the datasets are balanced to achieve accurate results.For the same the Synthetic Minority Over-Sampling Method(SMOTE)[3] which was introduced by Nitesh V. Chawla (2006) is used, code of which is presented in Figure 3.

```
#Applying SMOTE to balance the dataset
SMOTEData = SMOTE(diagnosis ~ ., trainFinal, perc.over = 100, k = 5, perc.under = 200)
table(SMOTEData$diagnosis)
```

Figure 3: SMOTE to balance the datasets

Figure 4 shows the code for SVM and CART algorithm. Both the algorithms are implemented using the caret package's train() function. First the dataset is balanced using the SMOTE() and then the models are applied on the balanced data. The perc.over is the over-sampling of minority class whereas perc.under is the under-sampling of majority class.

```
#Setting train and test data
library(caret)
set.seed(300)
SampleFinal <- createDataPartition(DataFinal$diagnosis, p= 0.80, list = FALSE)
trainFinal <- DataFinal[SampleFinal, ]
testFinal <- DataFinal[-SampleFinal, ]

#Balancing the Dataset
#install.packages("DMwR")
library(DMwR)
#install.packages("e1071")
library(e1071)

set.seed(300)
#Applying SMOTE to balance the dataset
SMOTEData = SMOTE(diagnosis ~ ., trainFinal, perc.over = 100, k = 5, perc.under = 200)
table(SMOTEData$diagnosis)

#10-fold Cross Validation
traincrt <- trainControl(method = "cv", number = 10)

##################################### SVM ########################
SVM_SMOTE <- train(diagnosis ~symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                   + smoothness_worst + smoothness_se + concavity_worst +
                     texture_se + concavity_se + concave.points_worst, data = SMOTEData, method = "svmLinear",
                   trControl = traincrt, preProcess = c("center", "scale"),
                   tuneLength = 10) #for tuning the algorithm

SVM_SMOTE
test_predictSMOTE <- predict(SVM_SMOTE, newdata = testFinal)
test_predictSMOTE
#To check accuracy
confusionMatrix(table(test_predictSMOTE, testFinal$diagnosis))

##################################### CART ########################
set.seed(200)
modelfitSMOTE <- train(diagnosis~symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                       + smoothness_worst + smoothness_se + concavity_worst +
                         texture_se + concavity_se + concave.points_worst, data = SMOTEData,
                       trControl = trainControl(method = "cv", number = 10),
                       method = "rpart")

print(modelfitSMOTE)

#prediction
predCARTSMOTE <- predict(modelfitSMOTE, testFinal)
confusionMatrix(table(predCARTSMOTE, testFinal$diagnosis))
```

Figure 4: SVM and CART using 10-fold Cross Validation

[3]https://medium.com/towards-artificial-intelligence/application-of-synthetic-minority-over-sampling-technique-smote-for-imbalanced-data-sets-509ab55cfdaf

```
##################################### KNN ##########################
k=10
for(k in 1:k){
library(caret)
set.seed(300)
SampleFinal <- createDataPartition(DataFinal$diagnosis, p= 0.80, list = FALSE)
trainFinal <- DataFinal[SampleFinal, ]
testFinal <- DataFinal[-SampleFinal, ]

#Balancing the Dataset
#install.packages("DMwR")
library(DMwR)
set.seed(300)
#Applying SMOTE to balance the dataset
SMOTEData = SMOTE(diagnosis ~ ., trainFinal, perc.over = 100, k = 5, perc.under = 200)
table(SMOTEData$diagnosis)


knnSMOTE <- kNN(diagnosis ~  symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                + smoothness_worst + smoothness_se + concavity_worst +
                  texture_se + concavity_se + concave.points_worst,SMOTEData,testFinal,norm=TRUE,k=25)

table(testFinal[,'diagnosis'],knnSMOTE)
}
confusionMatrix(table(testFinal[,'diagnosis'],knnSMOTE))
```

Figure 5: KNN Implementation Code

As shown in Figure 5, the kNN() is used to implement the KNN algorithm[4]. The k value taken is 20 for dataset 1. However, the model performance is analyzed at different k values including 10,15,20,25,30 of which k-value 20 and 25 produced good result for dataset 1 and 2 respectively and hence and used for training the algorithm.

In Figure 6, the nsiveBayes() function of e1071 package is used to train the NB algorithm[5].

```
########################################## NB #####################################
#10-fold Cross Validation
k=10
for (k in 1:k) {

  #Setting train and test data
  library(caret)
  set.seed(300)
  SampleFinal <- createDataPartition(DataFinal$diagnosis, p= 0.80, list = FALSE)
  trainFinal <- DataFinal[SampleFinal, ]
  testFinal <- DataFinal[-SampleFinal, ]

  set.seed(300)
  SMOTEData = SMOTE(diagnosis ~ ., trainFinal, perc.over = 100, k = 5, perc.under = 200)
  table(SMOTEData$diagnosis)

  #Using e1071 package
  nbnew <- naiveBayes(diagnosis~symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                      + smoothness_worst + smoothness_se + concavity_worst +
                        texture_se + concavity_se + concave.points_worst, data = SMOTEData )
  prednb <- predict(nbnew, newdata = testFinal)
  tabnbnew <- table(prednb, testFinal$diagnosis)
}
confusionMatrix(tabnbnew)
```

Figure 6: NB Implementation Code

Figure 7 shows the use of glm() which stands for "generalized linear models" for LR implementation. The function is passed with family as binomial(link="logit") parameter

[4]https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/kNN
[5]https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4930525/

which means logistic regression. The model is trained and tested with 0.5 probability i.e. if greater than 0.5 than malignant else it is benign.

```
################################################### LR ###################################
k=10
for (k in 1:10) {

  #Setting train and test data
  library(caret)
  set.seed(300)
  Sample4 <- createDataPartition(Dataset3$diagnosis, p= 0.80, list = FALSE)
  train4 <- Dataset3[Sample4, ]
  test4 <- Dataset3[-Sample4, ]

  set.seed(400)
  SMOTEtrain3 = SMOTE(diagnosis ~ ., train4, perc.over = 100, k = 5, perc.under = 200)
  table(SMOTEtrain3$diagnosis)


  LRmodelSMOTE <- glm(diagnosis ~ symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                      + smoothness_worst + smoothness_se + concavity_worst +
                        texture_se + concavity_se + concave.points_worst, family = binomial(link = "logit"),
                      data = SMOTEtrain3)
  summary(LRmodelSMOTE)

  Pred_trainSMOTE <- predict(LRmodelSMOTE, newdata = SMOTEtrain3, type = "response")    #response in case of prediction probability

  PredTSMOTE <- ifelse(Pred_trainSMOTE > 0.5, 1, 0)      # if Pred_train is greater than 0.5 then malignant else it is benign
  tableLRTrainSMOTE <- table(Predicted = PredTSMOTE, Actual = SMOTEtrain3$diagnosis)
  tableLRTrainSMOTE

  #on test data
  Pred_testSMOTE <- predict(LRmodelSMOTE, test4, type = "response")
  PredTstSMOTE <- ifelse(Pred_testSMOTE>0.5, 1, 0)
  tableLRtestSMOTE <- table(Predicted = PredTstSMOTE, Actual = test4$diagnosis)
  tableLRtestSMOTE
  caret::confusionMatrix(tableLRtestSMOTE)
}
confusionMatrix(tableLRtestSMOTE)
```

Figure 7: LR Implementation Code

3.1 Artificial Neural Network

Figure 8 shows the Atificial Neural Network implementation[6] using the neuralnet() package with hidden layer 2. The activation function used is logistic. The threshold parameter in the neuralnet() is not used as it generated same output with and without the threshold. The training is repeated for 5 times by setting the 'rep' parameter to 5 and checked if it produced same results and the model successfully produced similar output.

[6]https://www.rdocumentation.org/packages/neuralnet/versions/1.44.2/topics/neuralnet

```
k=10
for (k in 1:k) {

set.seed(300)
Sam <- createDataPartition(DF$diagnosis, p= 0.80, list = FALSE)
Tr <- DF[Sam, ]
Te <- DF[-Sam, ]

#Normalizing the data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
DFnorm <- as.data.frame(lapply(DF[2:31], normalize))
DFnorm <- cbind(DFnorm, DF[1])
Trnorm <- DFnorm[1:455,]
Tenorm <- DFnorm[456:569,]

set.seed(300)
smoTr <- SMOTE(diagnosis ~., Trnorm, perc.over = 100, k = 5, perc.under = 200)
table(smoTr$diagnosis)

#converting the factor target variable to numeric as neural networks doesn't work on factors.
smoTr$diagnosis <- as.numeric(smoTr$diagnosis)-1
#Te$diagnosis <- as.numeric(Te$diagnosis)-1
Tenorm$diagnosis <- as.numeric(Tenorm$diagnosis)-1

#Building the model
#install.packages("neuralnet")
library(neuralnet)
nnF <- neuralnet(diagnosis ~ symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                 + smoothness_worst + smoothness_se + concavity_worst +
                   texture_se + concavity_se + concave.points_worst, data = smoTr,
                 hidden = 2 ,linear.output = FALSE, rep = 5,
                 lifesign = "minimal",  act.fct = "logistic", learningrate = NULL,
                 exclude = NULL)
#for classification linear.output = FALSE, lifesign = func will print during calculation
nnF$result.matrix
plot(nnF, rep = "best")

#testing the model
test_tempF <- subset(Tenorm, select = c("symmetry_mean", "symmetry_se", "compactness_worst", "concave.points_se",
                                        "smoothness_worst", "smoothness_se", "concavity_worst",
                                        "texture_se", "concavity_se", "concave.points_worst"))
head(test_tempF)
predict_resultsF <- compute(nnF, test_tempF)
resultsF <- data.frame(actual = Tenorm$diagnosis, prediction = predict_resultsF$net.result)

#To display confusion matrix
resultsCMF <- sapply(resultsF, round, digits=0)
resultsCMdfF <- data.frame(resultsCMF)
attach(resultsCMdfF)
tableANNF <- table(actual, prediction)
}
confusionMatrix(tableANNF)
```

Figure 8: ANN Implementation with 2 Hidden Layer

3.2 Experiments with Artificial Neural Network

The neural network is trained with several hidden layers i.e. with 2,3,4,5 and 6 hidden layers and the sensitivity of all these layers are analyzed and the one with highest sensitivity is used for the final model. Figure 9 and Figure 10 shows the code of implementation of ANN with 3 and hidden layers.

```
## with 3 hidden layers
set.seed(300)
nnF3 <- neuralnet(diagnosis ~ symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                  + smoothness_worst + smoothness_se + concavity_worst +
                    texture_se + concavity_se + concave.points_worst, data = smoTr,
                  hidden = 3 ,linear.output = FALSE,
                  lifesign = "minimal",  act.fct = "logistic", learningrate = NULL,
                  exclude = NULL)
#for classification linear.output = FALSE, lifesign = func will print during calculation
nnF3$result.matrix
plot(nnF3, rep = "best")

#testing the model
test_tempF3 <- subset(Tenorm, select = c("symmetry_mean", "symmetry_se", "compactness_worst", "concave.points_se",
                                         "smoothness_worst", "smoothness_se", "concavity_worst",
                                         "texture_se", "concavity_se", "concave.points_worst"))
head(test_tempF)
predict_resultsF3 <- compute(nnF3, test_tempF3)
resultsF3 <- data.frame(actual = Tenorm$diagnosis, prediction = predict_resultsF3$net.result)

#To display confusion matrix
resultsCMF3 <- sapply(resultsF3, round, digits=0)
resultsCMdfF3 <- data.frame(resultsCMF3)
attach(resultsCMdfF3)
tableANNF3 <- table(actual, prediction)
}
confusionMatrix(tableANNF3)
```

Figure 9: ANN with 3 Hidden Layer

```
### with 4 hidden layers
set.seed(300)
nnF4 <- neuralnet(diagnosis ~ symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                  + smoothness_worst + smoothness_se + concavity_worst +
                    texture_se + concavity_se + concave.points_worst, data = smoTr,
                  hidden = 4 ,linear.output = FALSE,
                  lifesign = "minimal",  act.fct = "logistic", learningrate = NULL,
                  exclude = NULL)
#for classification linear.output = FALSE, lifesign = func will print during calculation
nnF4$result.matrix
plot(nnF4, rep = "best")

#testing the model
test_tempF4 <- subset(Tenorm, select = c("symmetry_mean", "symmetry_se", "compactness_worst", "concave.points_se",
                                         "smoothness_worst", "smoothness_se", "concavity_worst",
                                         "texture_se", "concavity_se", "concave.points_worst"))
head(test_tempF4)
predict_resultsF4 <- compute(nnF4, test_tempF4)
resultsF4 <- data.frame(actual = Tenorm$diagnosis, prediction = predict_resultsF4$net.result)

#To display confusion matrix
resultsCMF4 <- sapply(resultsF4, round, digits=0)
resultsCMdfF4 <- data.frame(resultsCMF4)
attach(resultsCMdfF4)
tableANNF4 <- table(actual, prediction)
}
confusionMatrix(tableANNF4)
```

Figure 10: ANN with 4 Hidden Layer

Figure 11 and Figure 12 displays the code for implementation of hidden layer 5 and 6.

```
### with 5 hidden layers
set.seed(300)
nnF5 <- neuralnet(diagnosis ~ symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                  + smoothness_worst + smoothness_se + concavity_worst +
                    texture_se + concavity_se + concave.points_worst, data = smoTr,
                  hidden = 5 ,linear.output = FALSE,
                  lifesign = "minimal",  act.fct = "logistic", learningrate = NULL,
                  exclude = NULL)
#for classification linear.output = FALSE, lifesign = func will print during calculation
nnF5$result.matrix
plot(nnF5, rep = "best")

#testing the model
test_tempF5 <- subset(Tenorm, select = c("symmetry_mean", "symmetry_se", "compactness_worst", "concave.points_se",
                                         "smoothness_worst", "smoothness_se", "concavity_worst",
                                         "texture_se", "concavity_se", "concave.points_worst"))
head(test_tempF5)
predict_resultsF5 <- compute(nnF5, test_tempF5)
resultsF5 <- data.frame(actual = Tenorm$diagnosis, prediction = predict_resultsF4$net.result)

#To display confusion matrix
resultsCMF5 <- sapply(resultsF5, round, digits=0)
resultsCMdfF5 <- data.frame(resultsCMF5)
attach(resultsCMdfF5)
tableANNF5 <- table(actual, prediction)
}
confusionMatrix(tableANNF5)
```

Figure 11: ANN with 5 Hidden Layer

```
### with 6 hidden layers
set.seed(300)
nnF6 <- neuralnet(diagnosis ~ symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                  + smoothness_worst + smoothness_se + concavity_worst +
                    texture_se + concavity_se + concave.points_worst, data = smoTr,
                  hidden = 6 ,linear.output = FALSE,
                  lifesign = "minimal",  act.fct = "logistic", learningrate = NULL,
                  exclude = NULL)
#for classification linear.output = FALSE, lifesign = func will print during calculation
nnF6$result.matrix
plot(nnF6, rep = "best")

#testing the model
test_tempF6 <- subset(Tenorm, select = c("symmetry_mean", "symmetry_se", "compactness_worst", "concave.points_se",
                                         "smoothness_worst", "smoothness_se", "concavity_worst",
                                         "texture_se", "concavity_se", "concave.points_worst"))
head(test_tempF6)
predict_resultsF6 <- compute(nnF6, test_tempF6)
resultsF6 <- data.frame(actual = Tenorm$diagnosis, prediction = predict_resultsF6$net.result)

#To display confusion matrix
resultsCMF6 <- sapply(resultsF6, round, digits=0)
resultsCMdfF6 <- data.frame(resultsCMF6)
attach(resultsCMdfF6)
tableANNF6 <- table(actual, prediction)
}
confusionMatrix(tableANNF6)
```

Figure 12: ANN with 6 Hidden Layer

3.3 Extreme Gradient Boosting

Figure 13 presents the code for the Xgboost algorithm. The "xgboost" package is installed first for the model to run. Similar to the trainControl() passed with 10-fold cross validation arguments, some of the arguments are passed inside the expand.grid() to tune the algorithm i.e. the learning rate, depth of the tree, number of iterations, etc. Both trainControl() and expand.grid() are then passed inside the train() function to train the model.

```r
#Setting working directory
setwd("C:/Users/Simitha Shetty/Desktop/Research Preoject/Week 2/Coding")
Datasetxg <- read.csv("WisconsinDataset.csv", stringsAsFactors = FALSE)

#Checking the structure
str(Datasetxg)
#changing factor to number
Datasetxg$diagnosis = factor(Datasetxg$diagnosis, levels = c("B","M"), labels = c(0,1))
str(Datasetxg)

#Checking the distribution of target variable
summary(Datasetxg)
table(Datasetxg$diagnosis)
#with Percentage (IMBALANCE in the dataset)
prop.table(table(Datasetxg$diagnosis))

#Setting train and test data
#install.packages("caret")
library(caret)
set.seed(300)
Samplexg <- createDataPartition(Datasetxg$diagnosis, p= 0.80, list = FALSE)
trainxg <- Datasetxg[Samplexg, ]
testxg <- Datasetxg[-Samplexg, ]

#Balancing the Data as it has imbalance
Smotexg <- SMOTE(diagnosis ~., trainxg, perc.over = 100, k = 5, perc.under = 200)
table(Smotexg$diagnosis)

#For 10-fold Cross Validation
trControlxg <- trainControl(method = "cv", number = 10)
#tuning the model
parametersxg <- expand.grid(eta=0.1, colsample_bytree=c(0.5,0.8), max_depth=c(2,3), nrounds = 100,
                            gamma=0, min_child_weight=2)
#eta= learning rate, randomly choosing no of columns at a time, max_dept is the depth of the tree, nrounds = number of iterations,
#gamma=minimum loss reduction to make further partition of leaf node, min_child_weight= minimum no of instances in each node


#Implementing Xgboost
#install.packages("xgboost")
library(xgboost)
modelxg <- train(diagnosis ~symmetry_mean + symmetry_se + compactness_worst + concave.points_se
                 + smoothness_worst + smoothness_se + concavity_worst +
                   texture_se + concavity_se + concave.points_worst, data = Smotexg, method = "xgbTree",
                 trControl = trControlxg, tuneGrid = parametersxg)

#on test data
predictxg <- predict(modelxg, newdata = testxg)
confusionMatrix(table(predictxg, testxg$diagnosis))
```

Figure 13: Xgboost Implementation code

All these figures show the implementation of models on Dataset 1. Similarly, the same code has been used for the implementation of models on Dataset 2.

References

Nitesh V. Chawla, Kevin W. Bowyer, L. O. H. W. P. K. (2006). Smote: Synthetic minority over-sampling technique, *Journal of Artificial Intelligence Research* pp. 321–357.