National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Sushant Parte
Student ID: X18137440

School of Computing
National College of Ireland

Supervisor: Dr. Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sushant Parte
Student ID:	X18137440
Programme:	MSc Data Analytics **Year:** 2019-2020
Module:	MSc Research Project
Lecturer:	Dr. Vladimir Milosavljevic
Submission Due Date:	12/12/2019
Project Title:	Prediction of hosting Animal centre outcome using supervised machine learning models

Word Count:1314 **Page Count: 10**

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: .....

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

Sushant Parte
Student ID: X18137440

1 Introduction

The configuration manual demonstrates the software and hardware requirements to implement the research project, which includes screenshots to show the step wise implementation of the procedure for the research "Prediction of Hosting Animal center outcome based on supervised machine learning models". The aim of the research is to classify and predict different outcomes for Austin Animal shelter which is one of the biggest and largest no kill shelter in the world. For building the machine learning model we implemented 4 different classification model such as Logistic regression, XGboost, Random Forest and Neural Network.

2 System Specification

The purpose of this section is to have knowledge about the different hardware and software specification required to build the project-

2.1 Hardware specification:

The Research was performed on local computer system having configuration as-

1. Processor- Intel ® core ™ i5-8250U
2. RAM- 8 GB
3. Operating System- 64-bit Windows 10 OS
4. Hard-disk space – 500 GB

2.2 Software specification-

1. R – The research project is entirely implemented using the R programming Language. Each of the activities namely Data cleaning, Feature Engineering, model implementation, evaluation of models and building of heat maps for confusion matrix is performed using R. To replicate the research work we need to install R from "https://cran.r-project.org/bin/windows/base/" and to run the r code we need an open source platform which is provided by R studio which can be downloaded from https://rstudio.com/products/rstudio/download/

2. Microsoft Excel is used to derive basic visualizations and accessing the Raw data in the .csv format

3. Version used- The implemented project uses R- 3.5.2 Eggshell Igloo and RStudio-1.2.5019

3. Stepwise implementation guide

1. Download the Required Software to run R.
2. Install the R 3.5.2
3. Installing Required Packages to run the code
4. Set the appropriate Working Directory path.
5. Execute the code and derive the results as mentioned in later section.

4. Data preparation

This Section consists of process to convert the raw data into monitoring data using different vectorization approaches.

4.1 Importing packages and libraries-

This section includes the libraries and packages used to derive results and clean data and evaluate the results[1].

```
install.packages("VIM")
library(VIM)
install.packages("tidyverse")
library(tidyverse)
install.packages("mice")
library(mice)
install.packages("tidyr")
library(tidyr)
install.packages("ggplot2")
library(ggplot2)
install.packages("dplyr")
library(dplyr)
install.packages("lubridate")
library(lubridate)
install.packages("wesanderson")
library(wesanderson)
install.packages("caret")
library(caret)
install.packages("randomForest")
library(randomForest)
install.packages("ROCR")
library(ROCR)
install.packages("pROC")
library(pROC)
install.packages("xgboost")
library(xgboost)
install.packages("MLmetrics")
library(MLmetrics)
install.packages("glmnet")
library(glmnet)
```

[1] https://cran.r-project.org/web/packages/available_packages_by_name.html

1. *VIM* – Visualization and imputing missing value is package used to exploring data and visualizing missing values.
2. *Tidyverse* – This package is set of packages which work in harmony because they share common data representation.
3. *Mice* – This package implements different method to deal with missing data which creates multiple imputation for missing data.
4. *Lubridate*– This package is used to deal with date and time in easier format.
5. *Wesanderson* – This package is used for color palettes from Wes Anderson movies.
6. *Caret* – Classification and regression training package is a set of function used for creating predictive models.
7. *randomForest* – Implements random forest algorithm.
8. *Xgboost* – Implements xgboost algorithm.
9. *MLmetrics* – This package includes a collection of evaluation metrices.
10. *Glmnet* – This package is extremely important for fitting regression models.

4.2 Data cleaning and feature engineering-

1. Setting up functions-

I. Convert to days- The function created helps in conversion of age to days[2].

```
convertToDays = function(n,num,age){
  y = vector(mode = "numeric", length = n)
  for (i in 1:n) {
    y[i] <- ifelse(grepl("year",age[i]),num[i] * 365,
                   ifelse(grepl("month", age[i]),num[i] * 30,
                          ifelse(grepl("week", age[i]),num[i] * 7,
                                 ifelse(grepl("day", age[i]),num[i], "" ))))
  }
  return (y)
}
```

II. Convert to period- the function created helps in conversion of time to period.

```
convertToPeriod = function(n,t){
  y = vector(mode = "character", length = n)
  for (i in 1:n) {
    y[i] <- ifelse(t[i]>=6 && t[i] < 12,"Morning",
                   ifelse(t[i]>=12 && t[i] < 18,"Afternoon",
                          ifelse(t[i]>=18 || t[i] < 06,"Night-time", "" )))
  }
  return (y)
}
```

[2] https://swcarpentry.github.io/r-novice-inflammation/02-func-R/

III. Convert to color- the function helps in conversion of mix and shades of color to basic colors.

```
convertToColor = function(n,t){
  y = vector(mode = "character", length = n)
  for (i in 1:n) {
    y[i] <- ifelse(t[i]=="Lilac"||t[i]=="Lynx"||t[i]=="Silver"||t[i]=="Blue"||t[i]=="Gray","Gray",
                   ifelse(t[i]=="Seal"||t[i]=="Black","Black",
                          ifelse(t[i]=="Agouti"||t[i]=="Chocolate"||t[i]=="Liver"||t[i]=="Ruddy"||t[i]=="Brown","Brown",
                                 ifelse(t[i]=="Apricot"||t[i]=="Flame"||t[i]=="Fawn"||t[i]=="Gold"||t[i]=="Tan"||t[i]=="Red"||t[i]=="Yellow"||
                                          t[i]=="Orange","Orange",
                                        ifelse(t[i]=="Cream"||t[i]=="Buff"||t[i]=="Pink"||t[i]=="White","White",
                                               ifelse(t[i]=="Calico"||t[i]=="Sable"||t[i]=="Tortie"||t[i]=="Torbie"||
                                                        t[i]=="Tricolor","Multi",NA))))))
  }
  return (y)
}
```

IV. Convert color to variable- the function helps in conversion of colors correctly predicted to variable.

```
createColorVar = function(n,t1,t2,color){
  y = vector(mode = "numeric", length = n)
  for (i in 1:n) {
    y[i] <- ifelse(t1[i]==color || t2[i]==color,1,0)
  }
  return (y)
}
```

2. Feature engineering –

The process of feature engineering helps in creating the monitoring data to fit the model and derive results for the same. This process is vectorized with creation of new predictive variables and to separate certain columns with multi categorical values[3].

[3] https://www.kaggle.com/haynakozo/r-feature-engineering-binning

```
# cleaning name
train$Name <- ifelse(nchar(train$Name) == 0, "Unknown", train$Name) #Change blank names to 'noname'
train$NameStatus <- if_else(train$Name == 'Unknown', 0, 1) #Add new variable indicating if name known or not

# cleaning sexuponoutcome
table(train$SexuponOutcome)  #'Neutered Male' most common
train$SexuponOutcome[train$SexuponOutcome == ""] = "Neutered Male"
train = tidyr::separate(train, SexuponOutcome, c("IntactStatus", "Sex"), sep = " ") #Separate 'SexuponOutcome' and add 'IntactStatus'
train$Sex[train$IntactStatus=="Unknown"] = "Unknown"

#cleaning date time
train$Year <- year(train$DateTime) #Add 'Year' predictor
train = train %>% mutate(Month = month.abb[month(train$Date)])
train$Weekday <- wday(train$DateTime, label=TRUE) #Add 'wday' predictor
train = tidyr::separate(train, DateTime , c("Date", "Time"), sep = " ", extra = "merge" ) #Separate 'DateTime'
train$Date = as.Date(train$Date)
train = tidyr::separate(train, Time, c("timeHr", "timeMin"), sep = ":", remove = FALSE, extra = "merge") #Separate 'Time'
train$timeHr = as.numeric(train$timeHr)
train$timePeriod = convertToPeriod(nrow(train), train$timeHr) #Add 'timePeriod' based on Time
train = subset(train, select = -c(timeHr,timeMin))

#cleaning breed
train = tidyr::separate(train, Breed, c("Breed", "isMix"), sep = " Mix") #Separate Breed, adding new predictor 'isMix'
train$isMix <- if_else(train$isMix == "", 1, 0, missing = 0) #1 if mixed breed, 0 else
```

The above figure shows the process of cleaning the columns having multiple names and no name, separating the colounns sexupon outcome and cleaning date and time with sepating each of the cell in excel file as new predictive variables.

```
#cleaning ageuponoutcome
train <- tidyr::separate(train, AgeuponOutcome, c("num", "AgeInDays"), sep = " ")
train$num <- as.numeric(train$num)
train$AgeInDays <- convertToDays(nrow(train),train$num,train$AgeInDays)
train$num <- NULL

#cleaning Ageindays
train$AgeInDays[train$AgeInDays == ""] = NA
train$AgeInDays[train$AgeInDays==0] = NA  #22 records with '0 years' changed to NA

set.seed(42)
x1 = c("OutcomeType","NameStatus","AnimalType","IntactStatus","Sex","isMix","AgeInDays","Year","Month","Weekday","timePeriod")
miceDF = train[,x1]
miceDF[,x1] = lapply(miceDF[,x1],as.factor) #Change to correct class
miceDF$AgeInDays = as.numeric(as.character(miceDF$AgeInDays)) #Change Age to numeric
train$AgeInDays = mice::complete(mice(miceDF))$AgeInDays  #Perform mice

# vectorization for color
temp = separate(train, Color, c("color1", "color2"), sep = "/", remove = FALSE)
temp = separate(temp, color1, c("color1.1", "color1.2"), sep = " ", remove = FALSE, extra = "merge")
temp = separate(temp, color2, c("color2.1", "color2.2"), sep = " ", remove = FALSE, extra = "merge")
temp$RColor1 = convertToColor(nrow(temp),temp$color1.1)
temp$RColor2 = convertToColor(nrow(temp),temp$color2.1)
temp$RColor2[is.na(temp$RColor2)] = " "
train$isBlack = createColorVar(nrow(temp),temp$RColor1,temp$RColor2,"Black") #Black,Seal
train$isGray = createColorVar(nrow(temp),temp$RColor1,temp$RColor2,"Gray") #Gray,Blue,Lilac,Lynx,Silver
train$isBrown = createColorVar(nrow(temp),temp$RColor1,temp$RColor2,"Brown") #Brown,Agouti,Chocolate,Liver,Ruddy
train$isOrange = createColorVar(nrow(temp),temp$RColor1,temp$RColor2,"Orange") #Orange,Apricot,Flame,Fawn,Gold,Tan,Red,Yellow
train$isWhite = createColorVar(nrow(temp),temp$RColor1,temp$RColor2,"White") #White,Cream,Buff,Pink
```

In the above figure of the R code implemented we clean the different columns as ageuponoutcome by creating new predictive column. The color column having multiple colors were changed to different basic colors.

5. Splitting of data and K-fold cross validation-

The below figure shows the implementation of K- fold cross validation using the holdout approach to split the data into training and testing sets[4].

```
train = subset(train,select = -c(OutcomeSubtype,AnimalID,Name,Date,Time,Breed,Color))

#Change predictors to correct classes
str(train)
x2 = c("OutcomeType","AnimalType","IntactStatus","Sex","isMix","NameStatus","Year","Month","timePeriod","isBlack","isGray","isBrown","isOrange","
train[,x2] = lapply(train[,x2],as.factor)
str(train)
write.csv(train, "C:/Users/Sushant Parte/Desktop/thesis/my_file.csv")
inTr = createDataPartition(train$OutcomeType, p=.8, list = FALSE)
tr = train[inTr,]
te = train[-inTr,]

#caret used to tune Random Forest, Neural Networks, and XGBoosted
#Sets up 5-fold cross validation for train function of caret
set.seed(123)
cv.5.folds = createMultiFolds(tr$OutcomeType, k = 5, times = 5)
ctrl = trainControl(method="cv", number = 5, index = cv.5.folds,
                    classProbs = TRUE, summaryFunction = multiClassSummary)
```

6. Implementation of logistic regression-

The below code is used to implement the logistic regression model using the glmnet function we evaluated the model using the log loss metrics with building the confusion matrix for predicting the values[5].

```
#Logistic Regression
#Data preparation for glmnet
x <- model.matrix(OutcomeType~.,data = tr)
y <- tr$OutcomeType
train_rows <- sample(1:dim(x)[[1]], .70*dim(x)[[1]])
training.x <- x[train_rows,]
testing.x <- x[-train_rows,]

#glmnet with 5-fold cross validation
set.seed(567)
cvfit <- cv.glmnet(x, y, family="multinomial", type.multinomial = "grouped", parallel = TRUE, nfolds = 5)

plot(cvfit)  #cross validation curve
predGLM <- predict(cvfit, newx = testing.x, s = "lambda.min", type = "class")
confusionMatrix(factor(predGLM[,1]), factor(y[-train_rows])) #confusion matrix
```

```
Overall Statistics

               Accuracy : 0.6412
                 95% CI : (0.6293, 0.653)
    No Information Rate : 0.399
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4394

 Mcnemar's Test P-Value : < 2.2e-16
```

[4] https://rdrr.io/cran/DiceEval/man/crossValidation.html
[5] https://www.r-bloggers.com/how-to-perform-a-logistic-regression-in-r/

The above figure is an output for logistic model implemneted with calculated accuracy for the model and kappa score.

7. Implementation of Neural Network-

The below figure shows the process of implementing Neural Network model to classify the data of Austin Animal center and predict the values for the data by building confusion matrix[6].

```
#nn
set.seed(456)
nn.model = train(OutcomeType ~ ., data = tr, method = "nnet",
                 tuneLength = 5, trControl = ctrl, metric = "logLoss")
#The final values used for the model were size = 9 and decay = 0.1.
predNN = predict(nn.model, newdata = te, type = "raw")
confusionMatrix(predNN, te$OutcomeType) #confusion matrix
#Size & decay tuning plot
plot(nn.model, main = "NN Size & Decay Tuning",
     cex.axis = 1.5, cex.lab = 1.5)
```

```
Overall Statistics

               Accuracy : 0.6257
                 95% CI : (0.6126, 0.6387)
    No Information Rate : 0.4029
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4067

 Mcnemar's Test P-Value : NA
```

The above figure shows the output obtained for Neural Network model with accuracy as factor to determine the performance of model.

[6] https://www.rdocumentation.org/packages/caret/versions/4.47/topics/train

8. Implementation of Random Forest-

The below figure shows the implementation of Random forest model with the predict function to predict the outcome and build the confusion matrix.

```
#Rf
set.seed(321)
rf.model = train(OutcomeType ~ ., data = tr, method = "rf",
                 tuneLength = 3, ntree = 500, trControl = ctrl,metric = "logLoss")
#The final value used for the model was mtry = 19.
predRF = predict(rf.model, te, probability = TRUE)
confusionMatrix(predRF, te$OutcomeType) #confusion matrix
#Outcome type error rates & OOB plot
plot(rf.model$finalModel, main = "RF Outcome Type Error Rates",
     cex.axis = 1.5, cex.lab = 1.5, cex.main = 2)
legend("topright",c("OOB","Adopt","Death","Euthanasia","Returned", "Transfer"),lty=1:6,fill = 1:6)
```

```
Overall Statistics

               Accuracy : 0.6503
                 95% CI : (0.6373, 0.6631)
    No Information Rate : 0.4029
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4699

 Mcnemar's Test P-Value : < 2.2e-16
```

The above image shows the output generated after fitting the random forest model with certain accuracy gained to determine the performance of the model.

9. Implementation of XGboost-

The below figure shows the process to build the Xgboost model to derive classification results and predict the values for outcome using confusion matrix[7].

[7] https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/

```
#xgb
#Creates grid which parameters will be tuned for
#tuning for depth, gamma, colsample_bytree, and min_child_weight
xgb_grid = expand.grid(nrounds = 30, max_depth = c(3, 6, 10 ), eta = 0.2, gamma = c(0.1,.5,1),
                       colsample_bytree = c(0.4, 0.7, 1.0), min_child_weight = c(0.5, 1, 1.5), subsample = 1)
set.seed(234)
xgb_model = train(OutcomeType ~ ., data = tr, method = "xgbTree",
                  tuneGrid = xgb_grid, trControl = ctrl, metric = "logLoss")
#optimal parameters
#nrounds max_depth eta gamma colsample_bytree min_child_weight
#     30         6 0.2   0.1                1                1
predXGB = predict(xgb_model, newdata=te, type = "raw")
confusionMatrix(predXGB, te$OutcomeType) #confusion matrix
#Get importance from optimal XGBoost model
Impor = varImp(xgb_model)$importance
```

```
Overall Statistics

               Accuracy : 0.6533
                 95% CI : (0.6403, 0.666)
    No Information Rate : 0.4029
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.472

 Mcnemar's Test P-Value : NA
```

The above figure shows the output generated for XGboost algorithm to derive classification results and performance of Xgboost with accuracy as factor.

10. Building color pallet-

The color pallet was generated to store confusion matrix for each algorithm in matrix format to derive confusion matrix in form of heat map.

```
my_palette <- colorRampPalette(c("red", "yellow", "green"))(n = 299)
#Store confusion matrices for each algorithm
glm.ConfMat = as.matrix.data.frame(confusionMatrix(predGLM[,1], y[-train_rows])$table)
rf.ConfMat = as.matrix.data.frame(confusionMatrix(predRF, te$OutcomeType)$table)
nn.ConfMat = as.matrix.data.frame(confusionMatrix(predNN, te$OutcomeType)$table)
xgb.ConfMat = as.matrix.data.frame(confusionMatrix(predXGB, te$OutcomeType)$table)
```

11. Heat map for confusion matrix-

The Generated confusion matrix were converted to heat maps with colors showing different shades for the large and low values to define the results properly to evaluate the results and understand the classification sone by the algorithms[8].

```
#GLM heat map
heatmap.2(glm.ConfMat, cellnote = glm.ConfMat, notecex = 2 ,  main = "Logistic Regression Confusion Matrix Heatmap",
          notecol="black", density.info="none", trace="none", margins =c(12,9), col=my_palette,revC = TRUE,
          dendrogram="row", Colv="NA",Rowv = NULL, labRow = c("Adoption", "Died", "Euthanasia", "Returned", "Transfer"),
          labCol = c("Adoption", "Died", "Euthanasia", "Returned", "Transfer"), cexRow = 1.5, cexCol = 1.5, key = FALSE)
#RF heat map
heatmap.2(rf.ConfMat, cellnote = rf.ConfMat, notecex = 2 ,  main = "Random Forest Confusion Matrix Heatmap",
          notecol="black", density.info="none", trace="none", margins =c(12,9), col=my_palette,
          dendrogram="row", Colv="NA", labRow = c("Adoption", "Died", "Euthanasia", "Returned", "Transfer"),
          labCol = c("Adoption", "Died", "Euthanasia", "Returned", "Transfer"), cexRow = 1.5, cexCol = 1.5, key = FALSE)

#NN heat map
heatmap.2(nn.ConfMat, cellnote = nn.ConfMat, notecex = 2 ,  main = "Neural Network Confusion Matrix Heatmap",
          notecol="black", density.info="none", trace="none", margins =c(12,9), col=my_palette,
          dendrogram="row", Colv="NA", labRow = c("Adoption", "Died", "Euthanasia", "Returned", "Transfer"),
          labCol = c("Adoption", "Died", "Euthanasia", "Returned", "Transfer"), cexRow = 1.5, cexCol = 1.5, key = FALSE)
#XGB heat map
heatmap.2(xgb.ConfMat, cellnote = xgb.ConfMat, notecex = 2 ,  main = "XGBoost Confusion Matrix Heatmap",
          notecol="black", density.info="none", trace="none", margins =c(12,9), col=my_palette,
          dendrogram="row", Colv="NA", labRow = c("Adoption", "Died", "Euthanasia", "Returned", "Transfer"),
          labCol = c("Adoption", "Died", "Euthanasia", "Returned", "Transfer"), cexRow = 1.5, cexCol = 1.5)
```

12. Conclusion-

The configuration manual proposes the actual implementation of proposed project in Rprogramming language. The implementation, claning and evaluation process are been implemented in given figures above which can be used to develop a machine learning project.

[8] http://sebastianraschka.com/Articles/heatmaps_in_r.html