Configuration Manual

Abnormal Foetuses Classification Based on Cardiotocographic Recordings Using Machine and Deep Learning Algorithms

MSc Research Project
Data Analytics

Jassem Alhaj Tamer
Student ID: X15021301

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Malwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Jassem Alhaj Tamer		
Student ID:	X15021301		
Programme:	MSc in Data Analytics	**Year:**	2020
Module:	Research Project		
Lecturer:	Dr Catherine Mulwa		
Submission Due Date:	17/08/2020		
Project Title:	Abnormal Foetuses Classification Based on Cardiotocography Recordings Using Machine Learning and Deep Learning Algorithms		
Word Count:	**3620**	**Page Count 35**	

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: Jass

Date: 17/08/2020

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

Abnormal Foetuses Classification Based on Cardiotocography Recordings Using Machine and Deep Learning Algorithms

Jassem Alhaj Tamer

Student ID: X15021301

1 Introduction

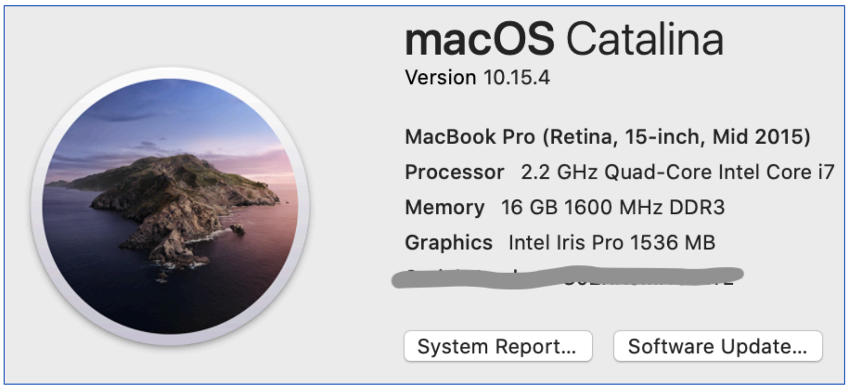
This configuration manual explains detailed steps applied to accomplish the objectives of classifying abnormal foetuses including hardware and software configuration, post-hoc analysis, data pre-processing, exploratory analysis, applied machine learning and deep learning algorithms along with the codes and results representation used to evaluate applied methods.

2 Environment Configuration

This section describes hardware and software configurations used as a platform to extract data from source to the final stages of models' evaluation.

2.1 Hardware Configuration

Regarding hardware configuration, the research project was performed on Apple Mac Book-pro laptop with the following specifications (refer Figure 1 and Figure 2).


macOS Catalina
Version 10.15.4

MacBook Pro (Retina, 15-inch, Mid 2015)
Processor 2.2 GHz Quad-Core Intel Core i7
Memory 16 GB 1600 MHz DDR3
Graphics Intel Iris Pro 1536 MB
[redacted]

Figure 1: CPU and Memory Configurations

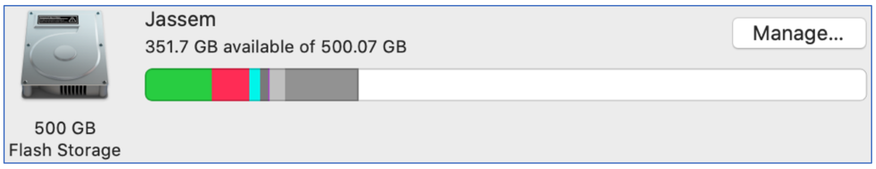
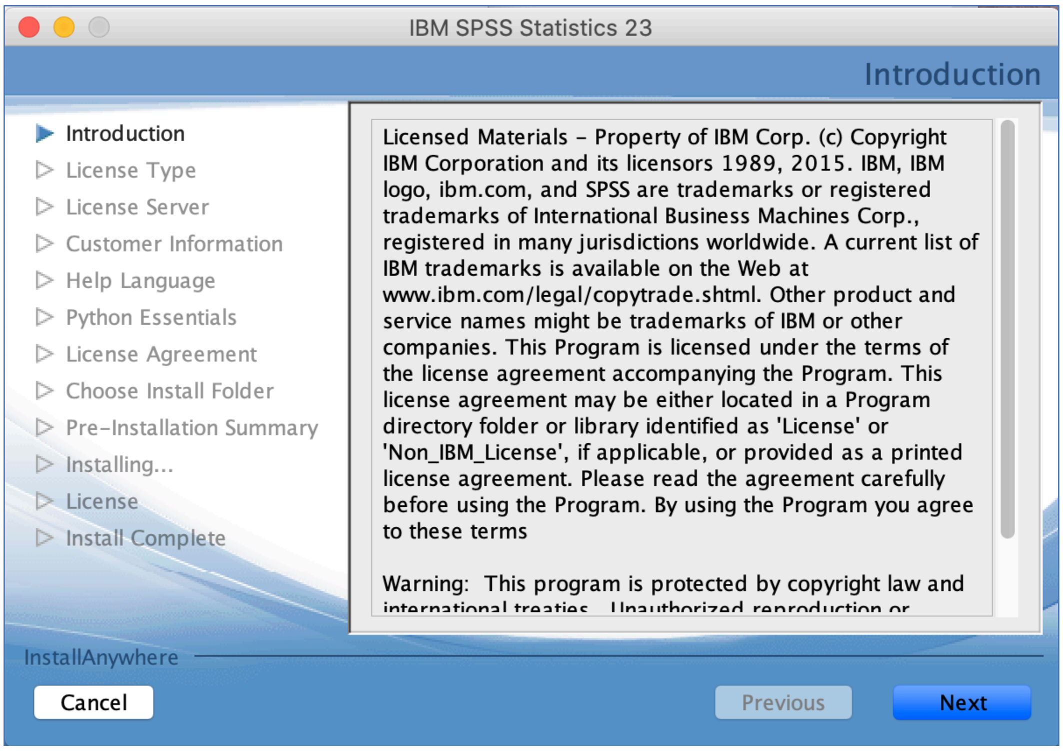


Figure 2: Hard Disk Configuration

2.2 Software Configuration

The Mac Book-pro laptop was equipped with Microsoft Office full package for Mac including Word, Excel and PowerPoint. Additional software such as Statistical Package for the Social Science "SPSS" version 23, the statistical and graphical computing tool "R" version 4.0.1, the user-friendly environment "RStudio" version 1.3.959 and data visualization tool "Tableau" version 2020.2.2 were installed to help in the classification of abnormal foetuses.

To perform the post-hoc test analysis incorporating Chi-square Goodness of Fit and Multivariate Analysis of Variance "MANOVA" approaches, IMB SPSS was installed (refer Figure 3 and Figure 4) from IBM official website[1].

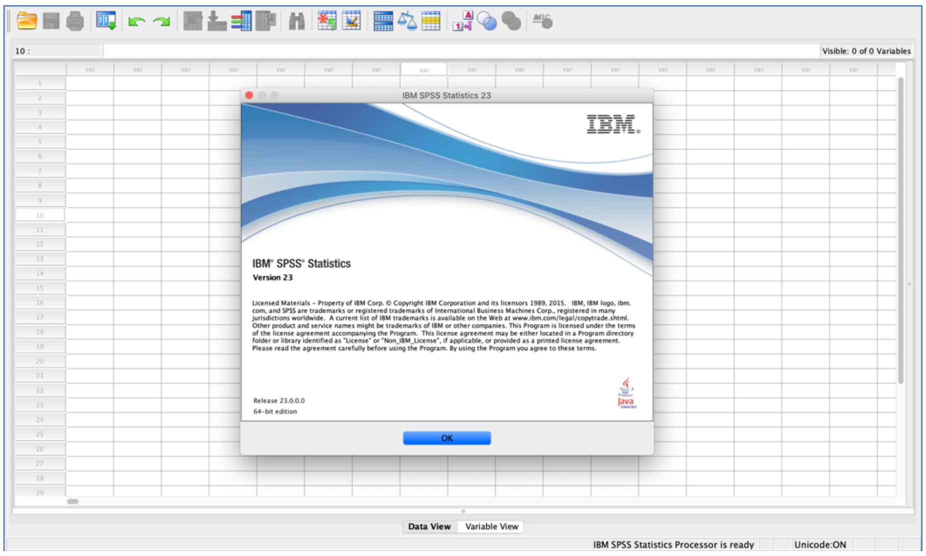


Licensed Materials – Property of IBM Corp. (c) Copyright IBM Corporation and its licensors 1989, 2015. IBM, IBM logo, ibm.com, and SPSS are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies. This Program is licensed under the terms of the license agreement accompanying the Program. This license agreement may be either located in a Program directory folder or library identified as 'License' or 'Non_IBM_License', if applicable, or provided as a printed license agreement. Please read the agreement carefully before using the Program. By using the Program you agree to these terms

Warning: This program is protected by copyright law and international treaties. Unauthorized reproduction or

Figure 3: IBM SPSS Installation Process

[1] https://www.ibm.com/support/pages/spss-statistics-230-now-available-download/



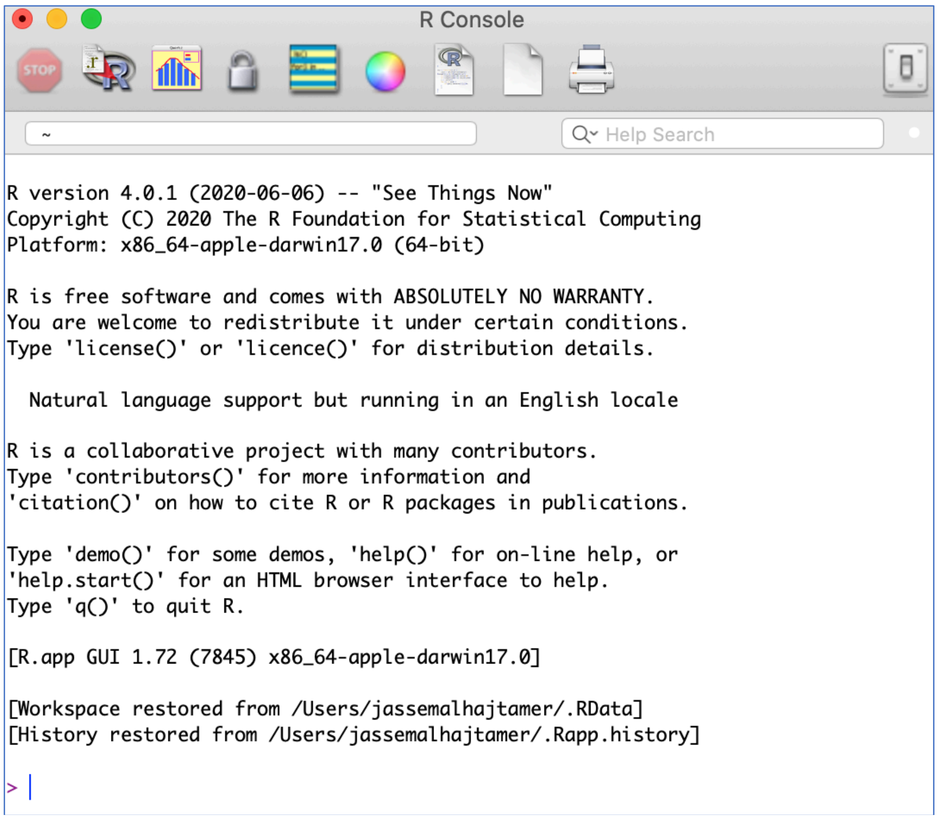
IBM® SPSS® Statistics
Version 23

Licensed Materials – Property of IBM Corp. © Copyright IBM Corporation and its licensors 1989, 2015. IBM, IBM logo, ibm.com, and SPSS are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies. This Program is licensed under the terms of the license agreement accompanying the Program. This license agreement may be either located in a Program directory folder or library identified as "License" or "Non_IBM_License", if applicable, or provided as a printed license agreement. Please read the agreement carefully before using the Program. By using the Program you agree to these terms.

Release 23.0.0.0
64-bit edition

Figure 4: SPSS Installation Process Completed

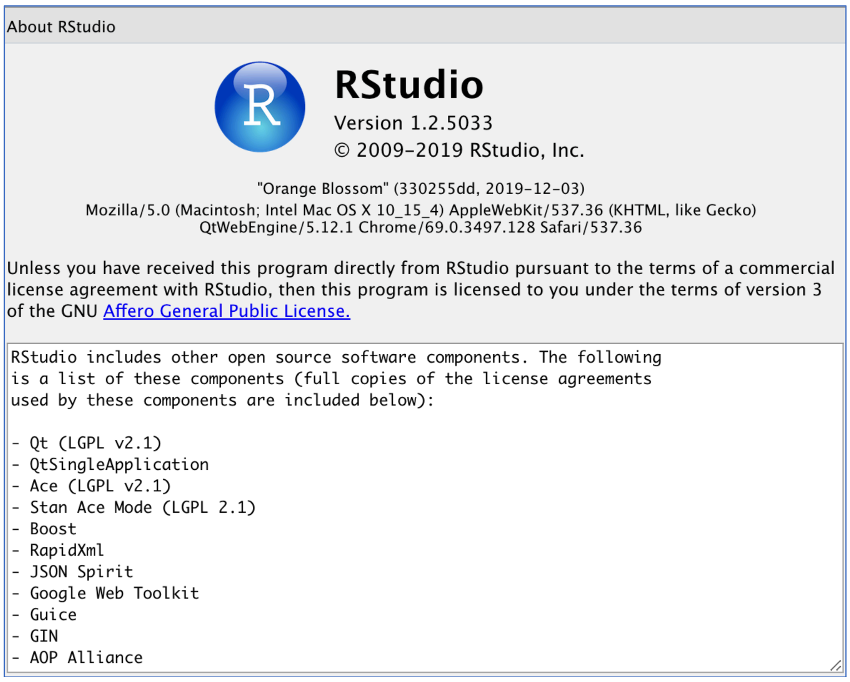
The R Console for Mac was installed from Cran website[1] (refer Figure 5 and Figure 6). Additionally, RStudio for Mac was installed from RStudio official website[2].



```
R version 4.0.1 (2020-06-06) -- "See Things Now"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.72 (7845) x86_64-apple-darwin17.0]

[Workspace restored from /Users/jassemalhajtamer/.RData]
[History restored from /Users/jassemalhajtamer/.Rapp.history]

> |
```

Figure 5: R Console Used as a Platform for RStudio

[1] https://cran.r-project.org/bin/macosx/
[2] https://rstudio.com/products/rstudio/download/


About RStudio

RStudio
Version 1.2.5033
© 2009-2019 RStudio, Inc.

"Orange Blossom" (330255dd, 2019-12-03)
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) QtWebEngine/5.12.1 Chrome/69.0.3497.128 Safari/537.36

Unless you have received this program directly from RStudio pursuant to the terms of a commercial license agreement with RStudio, then this program is licensed to you under the terms of version 3 of the GNU Affero General Public License.

```
RStudio includes other open source software components. The following
is a list of these components (full copies of the license agreements
used by these components are included below):

- Qt (LGPL v2.1)
- QtSingleApplication
- Ace (LGPL v2.1)
- Stan Ace Mode (LGPL 2.1)
- Boost
- RapidXml
- JSON Spirit
- Google Web Toolkit
- Guice
- GIN
- AOP Alliance
```

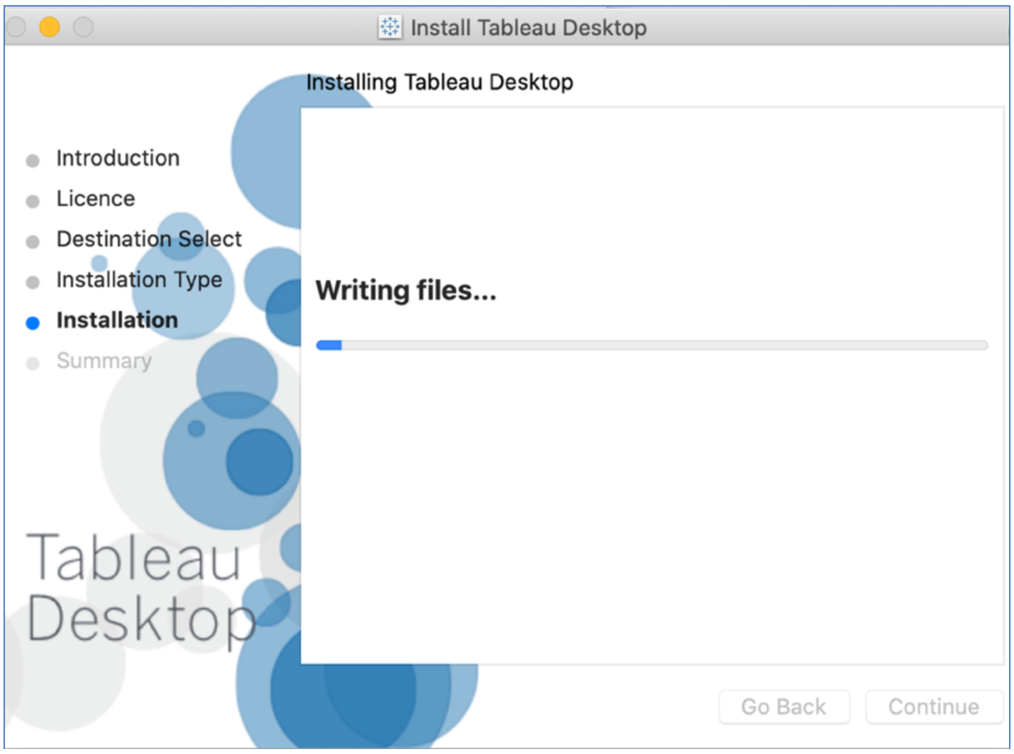
Figure 6: RStudio for R Codes Execution

The necessary R packages for exploratory analysis, data pre-processing and the developed machine learning and deep learning algorithms were installed as shown in Table 1.

Table 1: Required R Packages for Analysis and Graphs

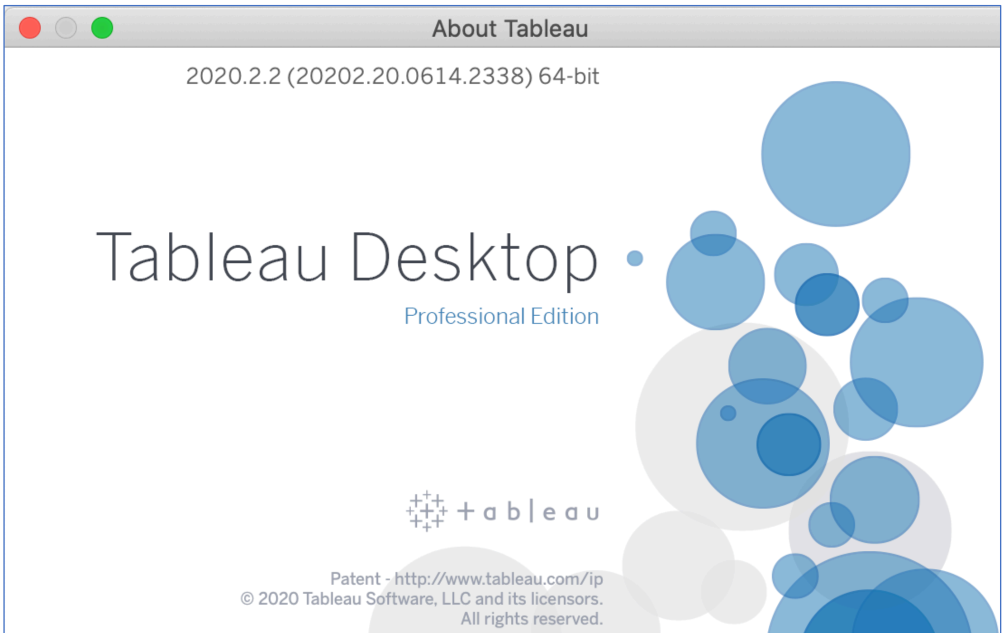
Package	Description	Version
caret	Classification and Regression Teaining	6.0-86
corrplot	Visualization of a correlationMatrix	0.84
ggplot2	Create Elegant Data Visualixations	3.3.1
GGaly	Extention to ggplot2	2.2.0
lattice	Trellis Graphics for R	0.20-41
doParallel	Foreach Parallel adaptor	1.0.15
tidyverse		1.3.0
ROSE	Random Over-sampling Examples	0.0-3
randomForest		4.6-14
e1071		1.7-3
C50	Decsion Tree and Rule-Based Models	0.1.3.1
pROC	Analyse ROC Curves	1.16.2
mlbench	Machine Learning Benchmark Problem	2.1-1
naivebayes		0.9.7
Matrix	Sparse and Dense Matrix Classes	1.2-18
dplyr		1.0.0
magrittr	A forward Pipe Operator	1.5
xgboost		1.0.0.2
keras		2.3.0.0
tensorflow		2.2.0
nnet	Feed-Forward Neural Networks	7.3-14

To perform results visualization part, Tableau was installed from the official website[1] using NCI student credentials for official registration (refer Figure 7 and Figure 8).



Writing files...

Figure 7: Tableau Installation Process



2020.2.2 (20202.20.0614.2338) 64-bit

Patent - http://www.tableau.com/ip
© 2020 Tableau Software, LLC and its licensors.
All rights reserved.

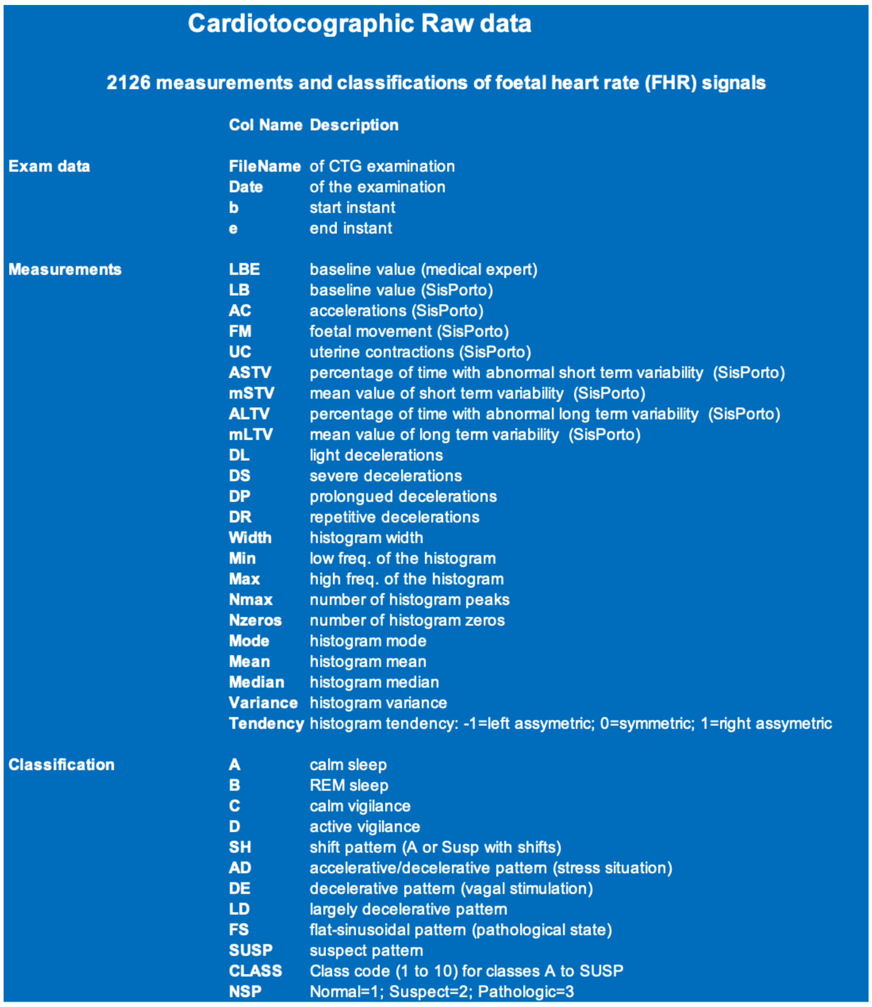
Figure 8: Tableau Installation Process Completed

[1] https://www.tableau.com/en-gb/trial/tableau-mac

3 Data Extraction, Post-hoc Test Analysis, Data Pre-processing and Exploratory Analysis

3.1 Data Extraction

Cardiotocography raw data was extracted from UCI Machine Learning Repository (Dua and Graff, 2017), (refer Figure 9).



Cardiotocographic Raw data

2126 measurements and classifications of foetal heart rate (FHR) signals

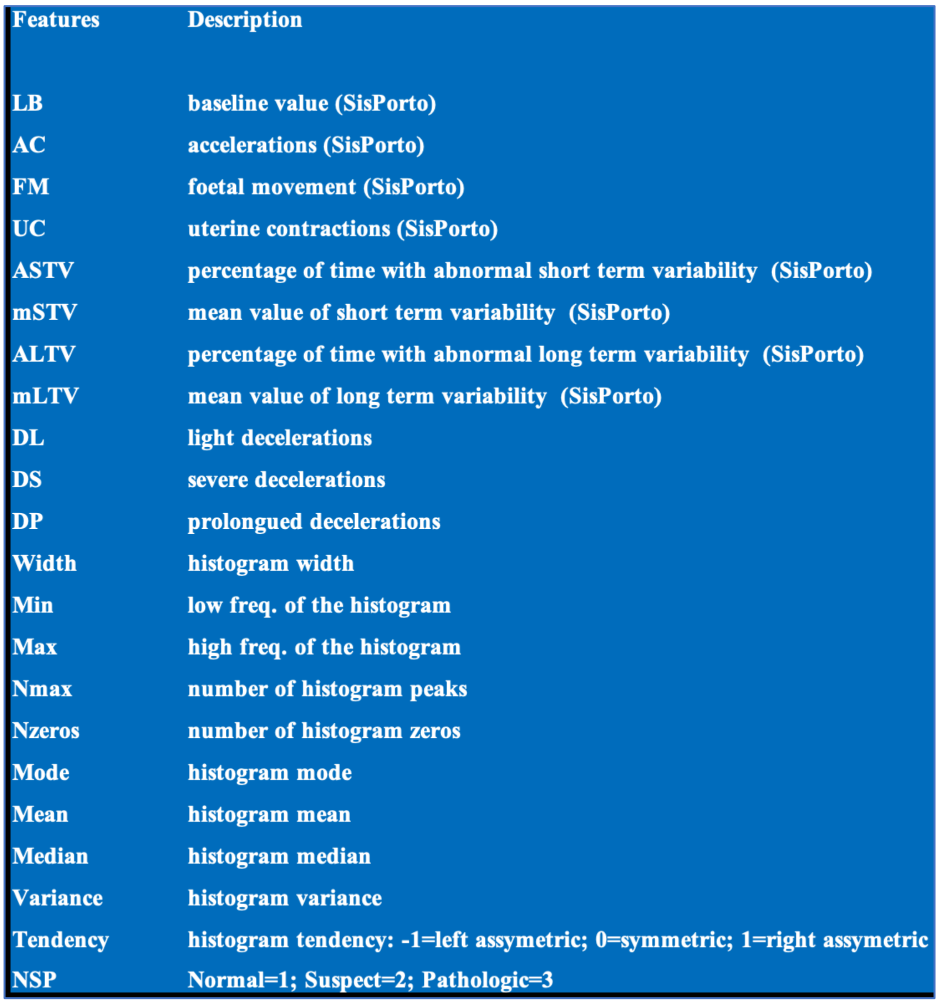
	Col Name	Description
Exam data	FileName	of CTG examination
	Date	of the examination
	b	start instant
	e	end instant
Measurements	LBE	baseline value (medical expert)
	LB	baseline value (SisPorto)
	AC	accelerations (SisPorto)
	FM	foetal movement (SisPorto)
	UC	uterine contractions (SisPorto)
	ASTV	percentage of time with abnormal short term variability (SisPorto)
	mSTV	mean value of short term variability (SisPorto)
	ALTV	percentage of time with abnormal long term variability (SisPorto)
	mLTV	mean value of long term variability (SisPorto)
	DL	light decelerations
	DS	severe decelerations
	DP	prolongued decelerations
	DR	repetitive decelerations
	Width	histogram width
	Min	low freq. of the histogram
	Max	high freq. of the histogram
	Nmax	number of histogram peaks
	Nzeros	number of histogram zeros
	Mode	histogram mode
	Mean	histogram mean
	Median	histogram median
	Variance	histogram variance
	Tendency	histogram tendency: -1=left assymetric; 0=symmetric; 1=right assymetric
Classification	A	calm sleep
	B	REM sleep
	C	calm vigilance
	D	active vigilance
	SH	shift pattern (A or Susp with shifts)
	AD	accelerative/decelerative pattern (stress situation)
	DE	decelerative pattern (vagal stimulation)
	LD	largely decelerative pattern
	FS	flat-sinusoidal pattern (pathological state)
	SUSP	suspect pattern
	CLASS	Class code (1 to 10) for classes A to SUSP
	NSP	Normal=1; Suspect=2; Pathologic=3

Figure 9: Raw CTG Data

Raw Data was converted from Excel format ".xls" to ".csv"[1]. In order to answer the research question, the exam data features "FileName", "Date", "b" and "e" were removed for being irrelevant to this study. Baseline value feature "LBE" recorded by medical expert was also removed because of holding duplicate values of baseline feature "LB" recorded by SisPorto cardiotocograph. Having the value of "0" for all entries, repetitive decelerations feature "DR" was also dropped.

[1] https://www.guru99.com/excel-vs-csv.html

The classification label "CLASS", which is a 10-class morphologic pattern recorded by medical experts was removed as it could cause overfitting with the existence of "NSP" targeted variable. As a such and in order to answer the research question, the measurement diagnostic variables and the classification feature NSP (refer Figure 10) were considered as the final CTG data to be tested by the post-hoc analysis to determine the power of this study.

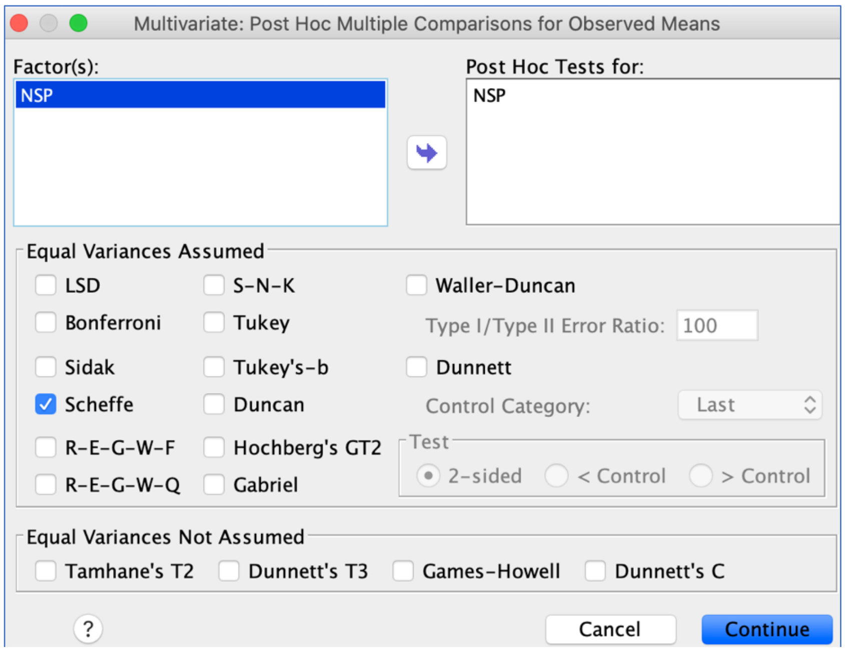


Features	Description
LB	baseline value (SisPorto)
AC	accelerations (SisPorto)
FM	foetal movement (SisPorto)
UC	uterine contractions (SisPorto)
ASTV	percentage of time with abnormal short term variability (SisPorto)
mSTV	mean value of short term variability (SisPorto)
ALTV	percentage of time with abnormal long term variability (SisPorto)
mLTV	mean value of long term variability (SisPorto)
DL	light decelerations
DS	severe decelerations
DP	prolongued decelerations
Width	histogram width
Min	low freq. of the histogram
Max	high freq. of the histogram
Nmax	number of histogram peaks
Nzeros	number of histogram zeros
Mode	histogram mode
Mean	histogram mean
Median	histogram median
Variance	histogram variance
Tendency	histogram tendency: -1=left assymetric; 0=symmetric; 1=right assymetric
NSP	Normal=1; Suspect=2; Pathologic=3

Figure 10: Final CTG Data

3.2 Post-hoc Analysis

The MANOVA was applied using "IBM SPSS". Since data points are not equal across different groups of the dependent variable, "Scheffe" parameter was used (refer Figure 11).



Multivariate: Post Hoc Multiple Comparisons for Observed Means

Factor(s): NSP

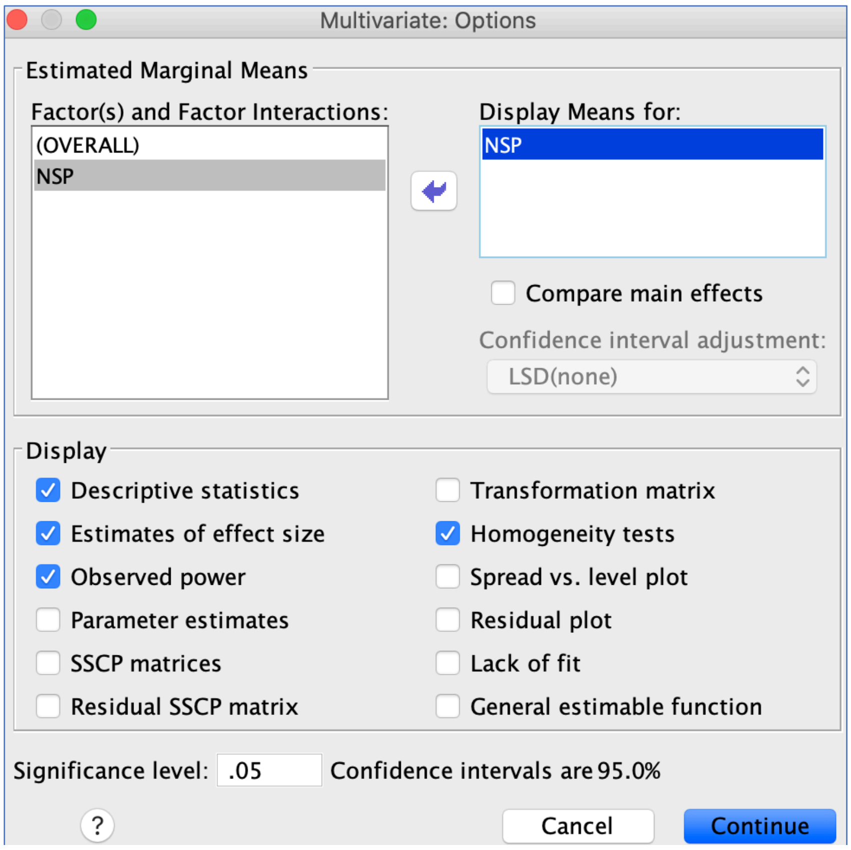
Post Hoc Tests for: NSP

Equal Variances Assumed
- [ ] LSD
- [ ] S-N-K
- [ ] Waller-Duncan
- [ ] Bonferroni
- [ ] Tukey
- Type I/Type II Error Ratio: 100
- [ ] Sidak
- [ ] Tukey's-b
- [ ] Dunnett
- [✓] Scheffe
- [ ] Duncan
- Control Category: Last
- [ ] R-E-G-W-F
- [ ] Hochberg's GT2
- [ ] R-E-G-W-Q
- [ ] Gabriel
- Test: [✓ 2-sided] < Control > Control

Equal Variances Not Assumed
- [ ] Tamhane's T2
- [ ] Dunnett's T3
- [ ] Games-Howell
- [ ] Dunnett's C

Figure 11: Post-hoc Test for Observed Means

The test analysis was performed at 0.05 significance level (refer Figure 12).



Multivariate: Options

Estimated Marginal Means

Factor(s) and Factor Interactions: (OVERALL), NSP

Display Means for: NSP

- [ ] Compare main effects

Confidence interval adjustment: LSD(none)

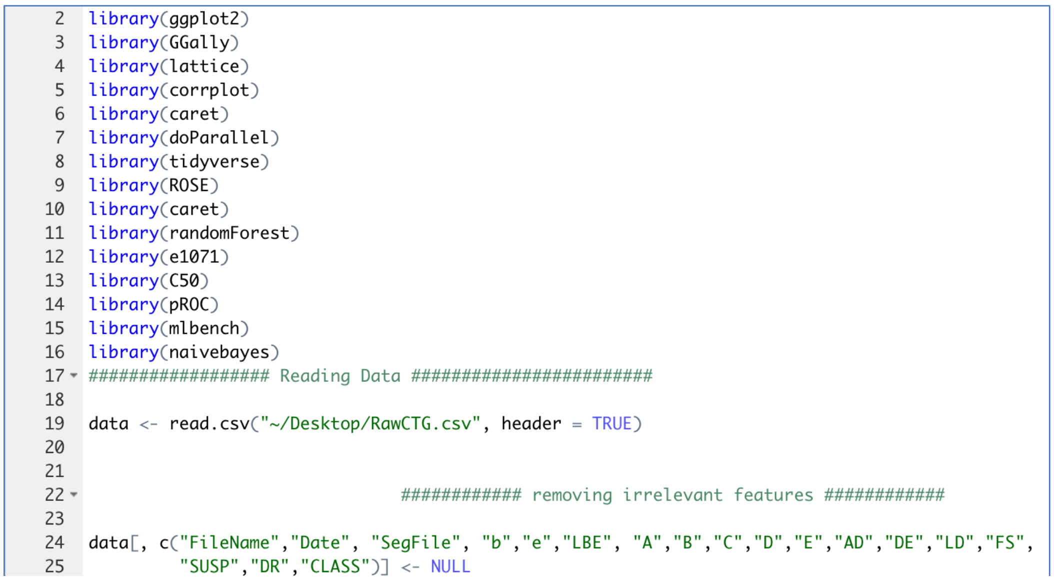
Display
- [✓] Descriptive statistics
- [ ] Transformation matrix
- [✓] Estimates of effect size
- [✓] Homogeneity tests
- [✓] Observed power
- [ ] Spread vs. level plot
- [ ] Parameter estimates
- [ ] Residual plot
- [ ] SSCP matrices
- [ ] Lack of fit
- [ ] Residual SSCP matrix
- [ ] General estimable function

Significance level: .05 Confidence intervals are 95.0%

Figure 12: Statistical Parameters and Significance Level

3.3 Data Pre-processing

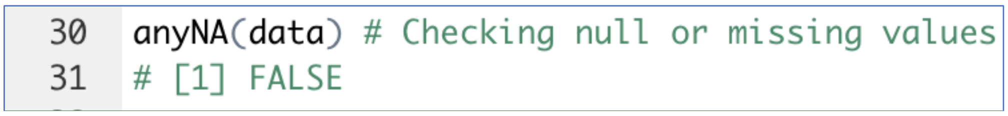
Data was pre-processed using R. Initially, irrelevant features were removed from raw data. Figure 13 shows the R codes used for this purpose. The source of R codes is from a previous knowledge in R during level 8 and 9 in data analytics.



```r
library(ggplot2)
library(GGally)
library(lattice)
library(corrplot)
library(caret)
library(doParallel)
library(tidyverse)
library(ROSE)
library(caret)
library(randomForest)
library(e1071)
library(C50)
library(pROC)
library(mlbench)
library(naivebayes)
################# Reading Data #######################

data <- read.csv("~/Desktop/RawCTG.csv", header = TRUE)


                                ########### removing irrelevant features ###########

data[, c("FileName","Date", "SegFile", "b","e","LBE", "A","B","C","D","E","AD","DE","LD","FS",
         "SUSP","DR","CLASS")] <- NULL
```

Figure 13: Reading Raw Data and Removing Irrelevant Features

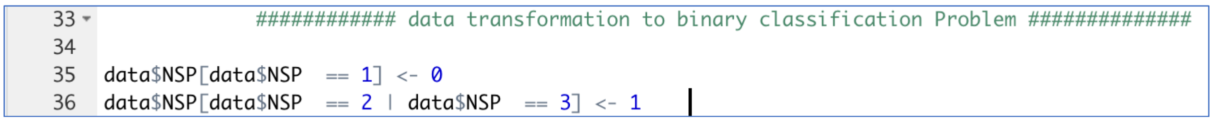
The CTG data was checked for any missing values. Fortunately, no missing values were found (refer Figure 14).



```r
anyNA(data) # Checking null or missing values
# [1] FALSE
```

Figure 14: Checking for Missing Values

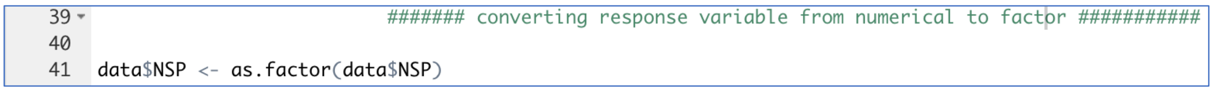
Response variable NSP has been transformed to binary classification where "Normal" class was coded to "0" and "Abnormal = Suspect or Pathologic" class was coded to "1" (refer Figure 15).



```r
             ############ data transformation to binary classification Problem ##############

data$NSP[data$NSP  == 1] <- 0
data$NSP[data$NSP  == 2 | data$NSP  == 3] <- 1
```

Figure 15: Coding Normal Class to 0 and Suspect or Pathologic Class to 1

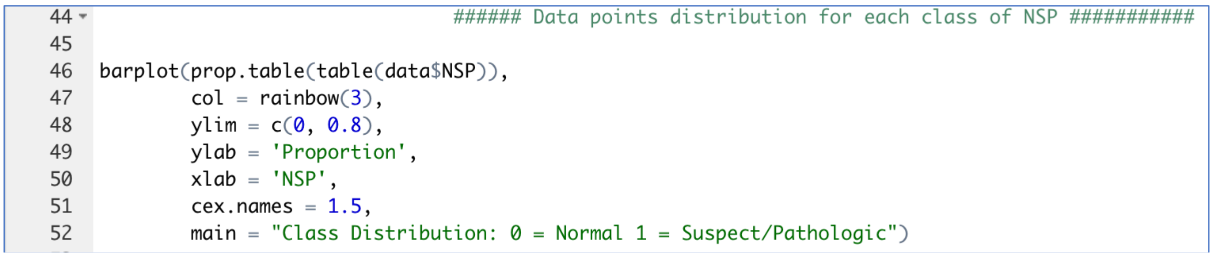
Additionally, NSP variable was converted to factor type (refer Figure 16).



```
                                        ####### converting response variable from numerical to factor ###########

data$NSP <- as.factor(data$NSP)
```

Figure 16: Converting Response Variable to Factor Type

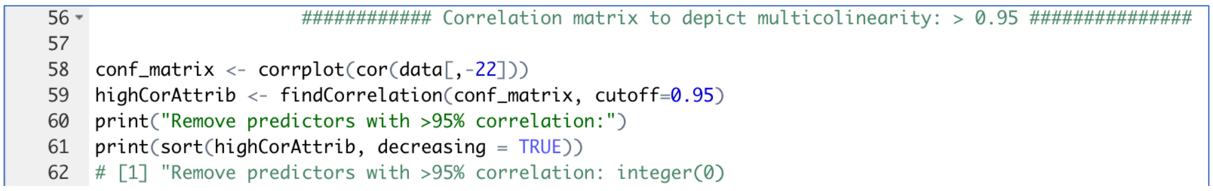
Data points distribution in each class of NSP variable was checked (refer Figure 17) where a class imbalance problem clearly appeared shown in Figure 5 in the technical report.



```
                                                ###### Data points distribution for each class of NSP ###########

barplot(prop.table(table(data$NSP)),
        col = rainbow(3),
        ylim = c(0, 0.8),
        ylab = 'Proportion',
        xlab = 'NSP',
        cex.names = 1.5,
        main = "Class Distribution: 0 = Normal 1 = Suspect/Pathologic")
```

Figure 17: Data Points Distribution for Each Class of Response Variable

Referring to correlation matrix (Figure 6) in the technical report, no multicollinearity problem has been found between independent variables which assures the suitability of independent features. The following R code (refer Figure 18) shows no variable bears a correlation greater than 95% to other counterparts.



```
                    ############ Correlation matrix to depict multicolinearity: > 0.95 ###############

conf_matrix <- corrplot(cor(data[,-22]))
highCorAttrib <- findCorrelation(conf_matrix, cutoff=0.95)
print("Remove predictors with >95% correlation:")
print(sort(highCorAttrib, decreasing = TRUE))
# [1] "Remove predictors with >95% correlation: integer(0)
```

Figure 18: Correlation Matrix to Depict Multicollinearity Problem

In order to avoid misleading performance of developed classifiers, upper and lower outliers in CTG dataset (refer Figure 19), were handled using a Winsorizing method in R (refer Figure 20).

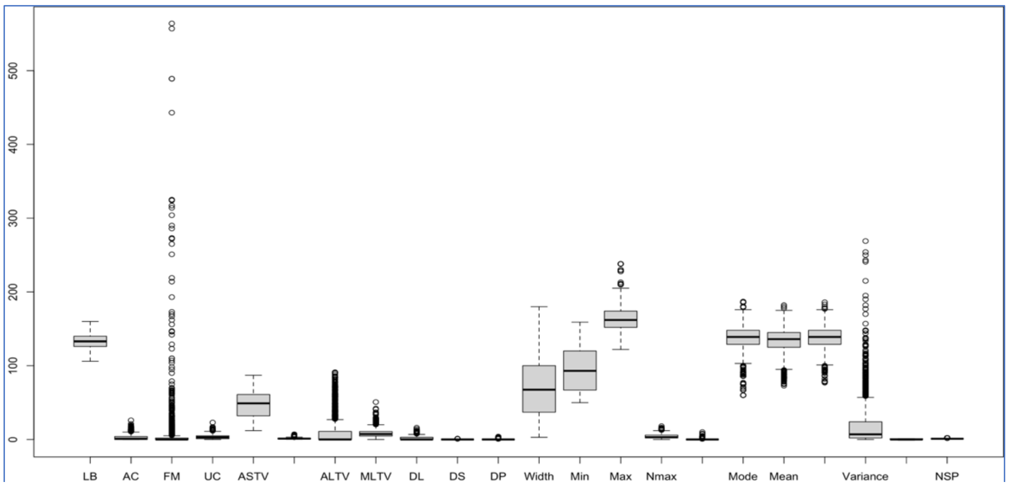
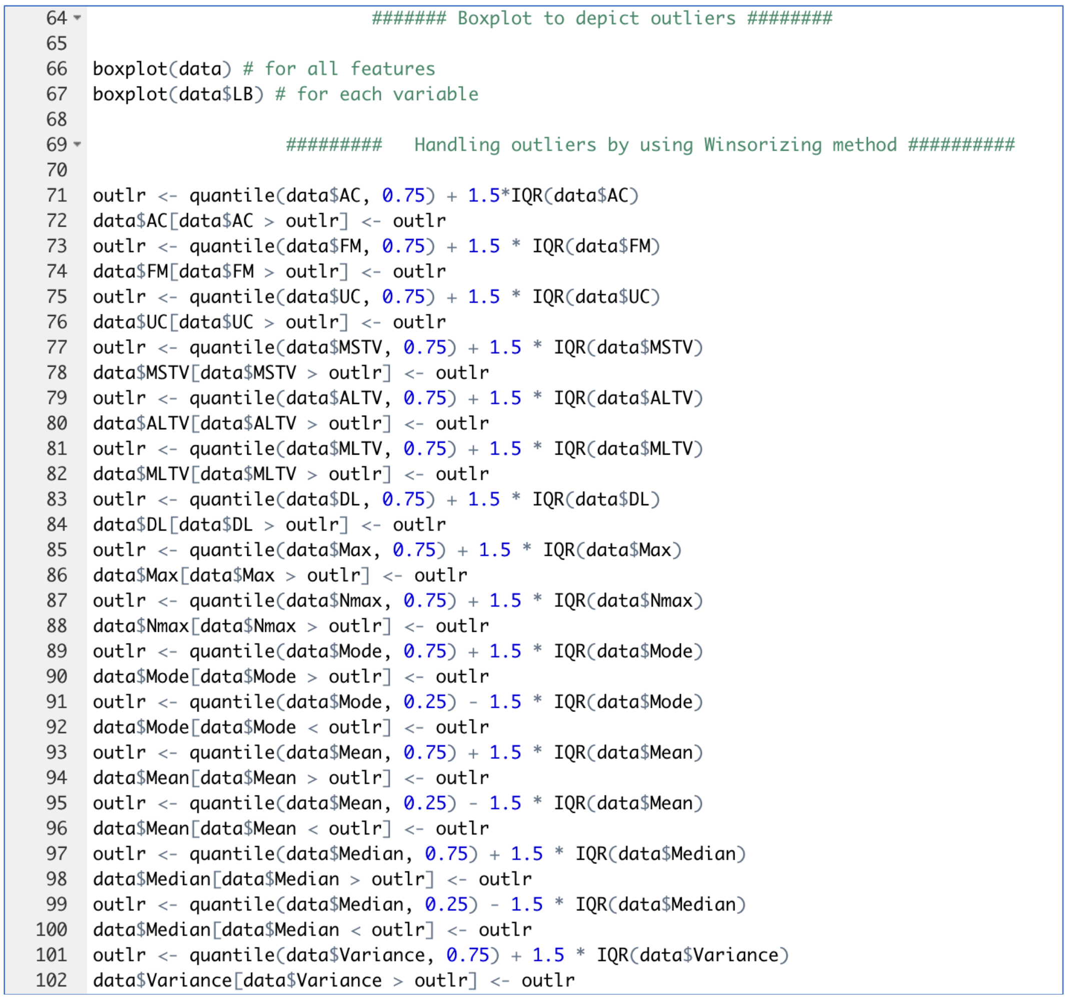


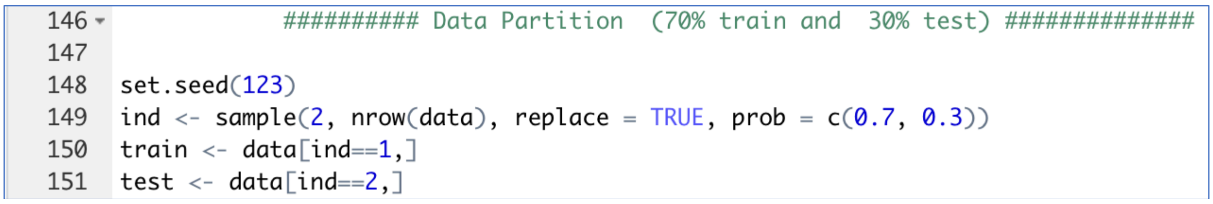
Figure 19: Boxplots for Upper and Lower Outliers



```r
                              ####### Boxplot to depict outliers ########

boxplot(data) # for all features
boxplot(data$LB) # for each variable

                  #########   Handling outliers by using Winsorizing method ##########

outlr <- quantile(data$AC, 0.75) + 1.5*IQR(data$AC)
data$AC[data$AC > outlr] <- outlr
outlr <- quantile(data$FM, 0.75) + 1.5 * IQR(data$FM)
data$FM[data$FM > outlr] <- outlr
outlr <- quantile(data$UC, 0.75) + 1.5 * IQR(data$UC)
data$UC[data$UC > outlr] <- outlr
outlr <- quantile(data$MSTV, 0.75) + 1.5 * IQR(data$MSTV)
data$MSTV[data$MSTV > outlr] <- outlr
outlr <- quantile(data$ALTV, 0.75) + 1.5 * IQR(data$ALTV)
data$ALTV[data$ALTV > outlr] <- outlr
outlr <- quantile(data$MLTV, 0.75) + 1.5 * IQR(data$MLTV)
data$MLTV[data$MLTV > outlr] <- outlr
outlr <- quantile(data$DL, 0.75) + 1.5 * IQR(data$DL)
data$DL[data$DL > outlr] <- outlr
outlr <- quantile(data$Max, 0.75) + 1.5 * IQR(data$Max)
data$Max[data$Max > outlr] <- outlr
outlr <- quantile(data$Nmax, 0.75) + 1.5 * IQR(data$Nmax)
data$Nmax[data$Nmax > outlr] <- outlr
outlr <- quantile(data$Mode, 0.75) + 1.5 * IQR(data$Mode)
data$Mode[data$Mode > outlr] <- outlr
outlr <- quantile(data$Mode, 0.25) - 1.5 * IQR(data$Mode)
data$Mode[data$Mode < outlr] <- outlr
outlr <- quantile(data$Mean, 0.75) + 1.5 * IQR(data$Mean)
data$Mean[data$Mean > outlr] <- outlr
outlr <- quantile(data$Mean, 0.25) - 1.5 * IQR(data$Mean)
data$Mean[data$Mean < outlr] <- outlr
outlr <- quantile(data$Median, 0.75) + 1.5 * IQR(data$Median)
data$Median[data$Median > outlr] <- outlr
outlr <- quantile(data$Median, 0.25) - 1.5 * IQR(data$Median)
data$Median[data$Median < outlr] <- outlr
outlr <- quantile(data$Variance, 0.75) + 1.5 * IQR(data$Variance)
data$Variance[data$Variance > outlr] <- outlr
```

Figure 20: Winsorising Approach to Handle Outliers

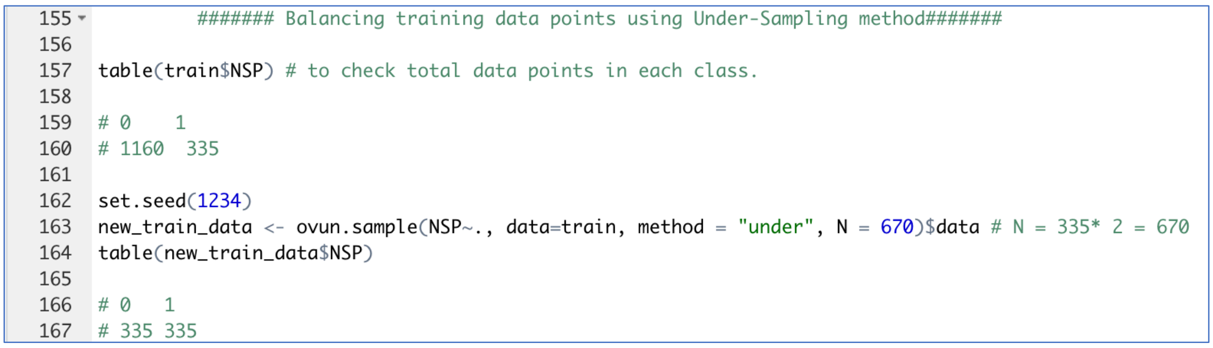
To apply machine learning and deep learning models, data was partitioned to 70% as a training dataset and 30% as a testing dataset (refer Figure 21).



```r
                ########## Data Partition  (70% train and  30% test) ###############

set.seed(123)
ind <- sample(2, nrow(data), replace = TRUE, prob = c(0.7, 0.3))
train <- data[ind==1,]
test <- data[ind==2,]
```

Figure 21: Data Partitioning to Train and Test Subsets

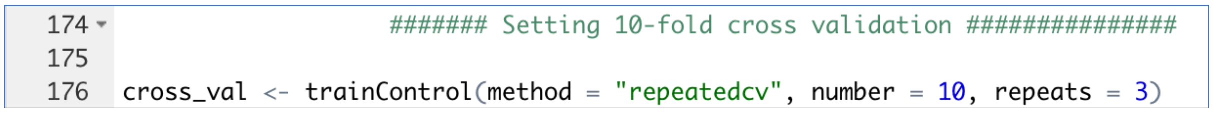
The above data partition resulted in an unbalanced training dataset due to the class imbalance problem already existed. To address this problem, under-sampling majority "Normal" class method was applied. To balance the two classes, "N" was given a value of 670 resulted from data points for abnormal foetuses which is 355 records times 2 (refer Figure 22).



```r
              ####### Balancing training data points using Under-Sampling method#######

table(train$NSP) # to check total data points in each class.

# 0    1
# 1160  335

set.seed(1234)
new_train_data <- ovun.sample(NSP~., data=train, method = "under", N = 670)$data # N = 335* 2 = 670
table(new_train_data$NSP)

# 0   1
# 335 335
```

Figure 22: Under-sampling Approach to Resolve Class Imbalance Problem

A cross-validation with 10-folds and 3 repeats is set up to be used when building classification models shown in Figure 23.

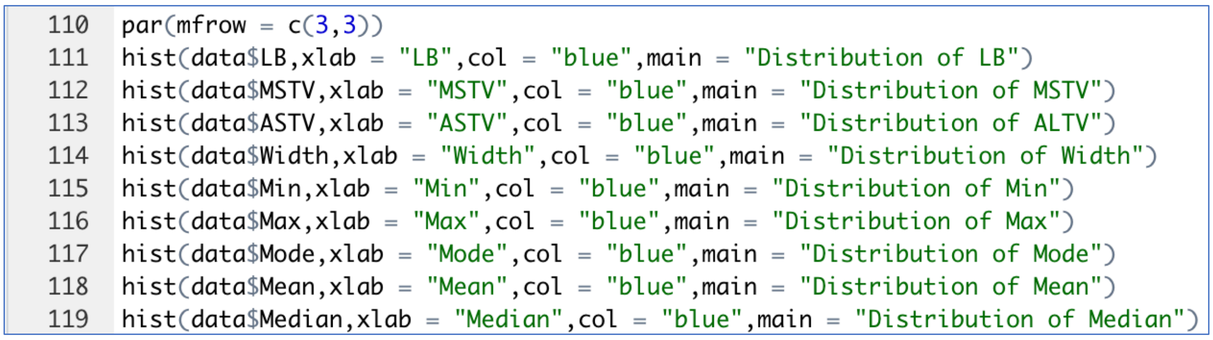


```r
                    ####### Setting 10-fold cross validation ###############

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

Figure 23: 10-fold Cross-Validation Technique

3.4 Exploratory Analysis

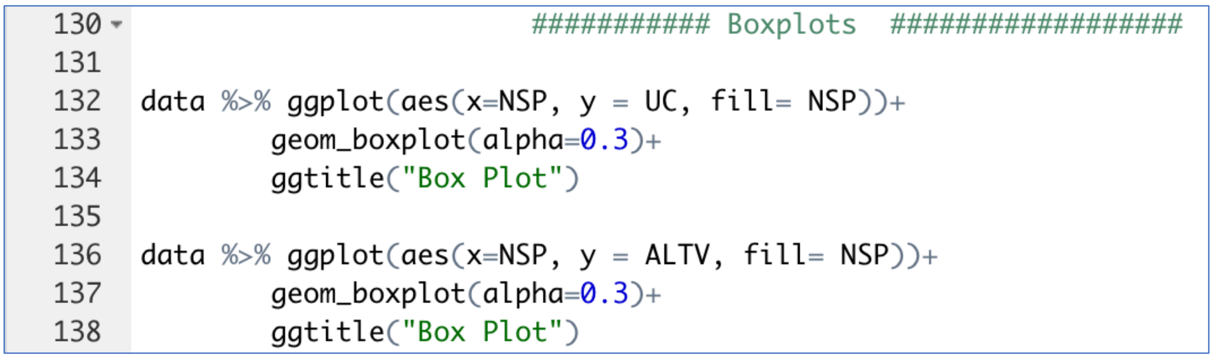
Histograms that were used to explore the distribution of data points and for each independent variable (refer Figure 8 in technical report), showed majority of variables tended to a bell-shaped normal distribution. R codes used for this task are shown Figure 24.



```r
par(mfrow = c(3,3))
hist(data$LB,xlab = "LB",col = "blue",main = "Distribution of LB")
hist(data$MSTV,xlab = "MSTV",col = "blue",main = "Distribution of MSTV")
hist(data$ASTV,xlab = "ASTV",col = "blue",main = "Distribution of ALTV")
hist(data$Width,xlab = "Width",col = "blue",main = "Distribution of Width")
hist(data$Min,xlab = "Min",col = "blue",main = "Distribution of Min")
hist(data$Max,xlab = "Max",col = "blue",main = "Distribution of Max")
hist(data$Mode,xlab = "Mode",col = "blue",main = "Distribution of Mode")
hist(data$Mean,xlab = "Mean",col = "blue",main = "Distribution of Mean")
hist(data$Median,xlab = "Median",col = "blue",main = "Distribution of Median")
```

Figure 24: Histograms to Explore Data Points Distribution

The significant correlation between the response and other predictors was represented by boxplots using "ggplot" package (refer Figure 10 in technical report). The R codes used for visualisation are shown in Figure 25.



```
                                        ###########  Boxplots   ##################

data %>% ggplot(aes(x=NSP, y = UC, fill= NSP))+
        geom_boxplot(alpha=0.3)+
        ggtitle("Box Plot")

data %>% ggplot(aes(x=NSP, y = ALTV, fill= NSP))+
        geom_boxplot(alpha=0.3)+
        ggtitle("Box Plot")
```

Figure 25: Boxplots to Explore Significant Correlations

Additionally, a correlation between other independent variables was visualized as shown in Figure 26 and 27. While a positive correlation between "LB" and "Median" features was seen, the "Width" feature showed a negative correlation to "Min".

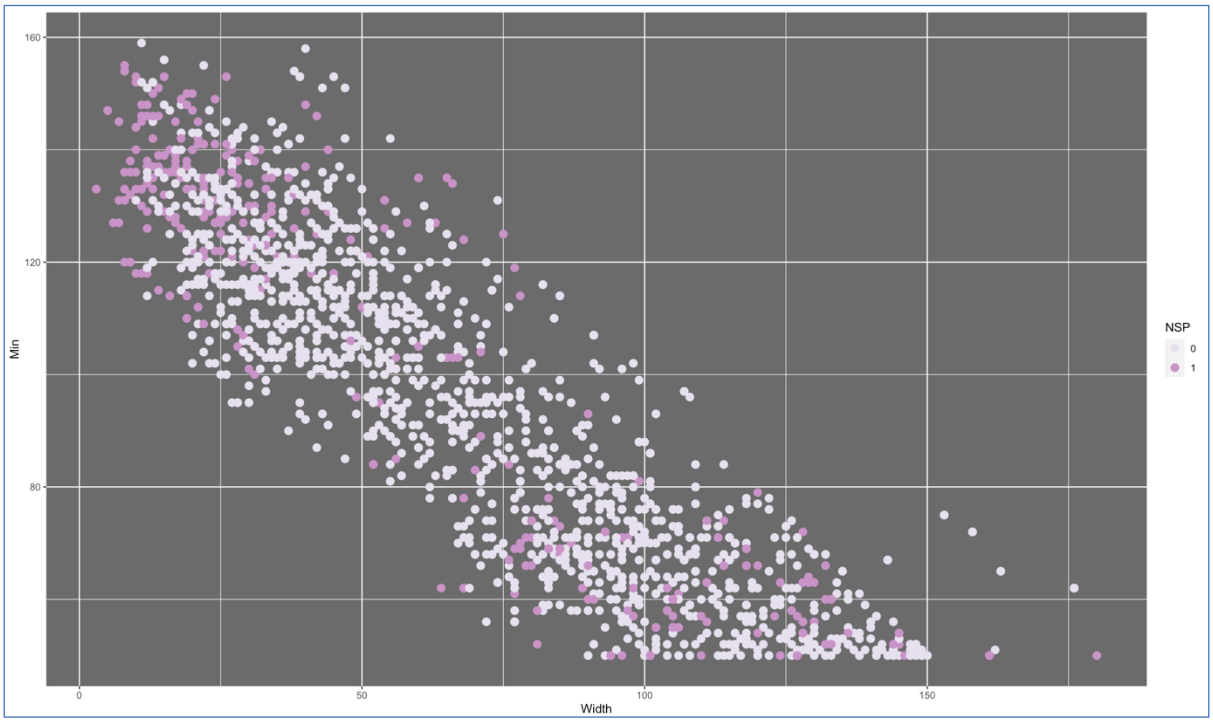


Figure 26: Correlation Between Width and Min Features

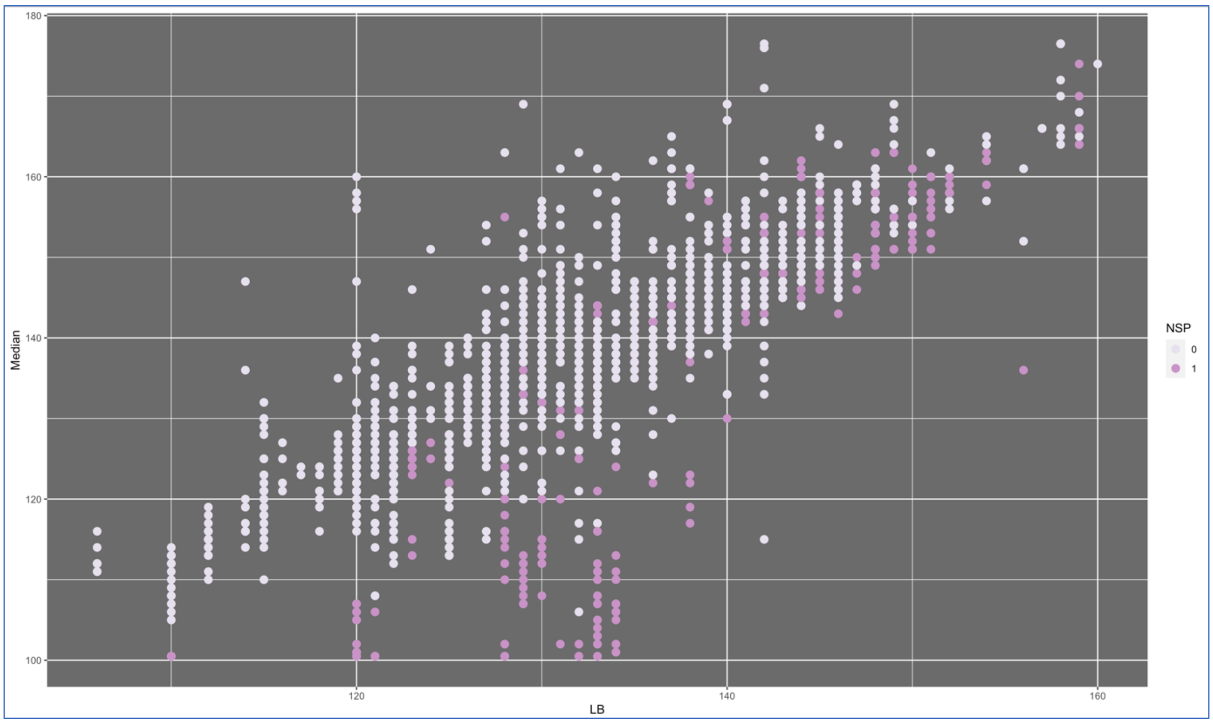
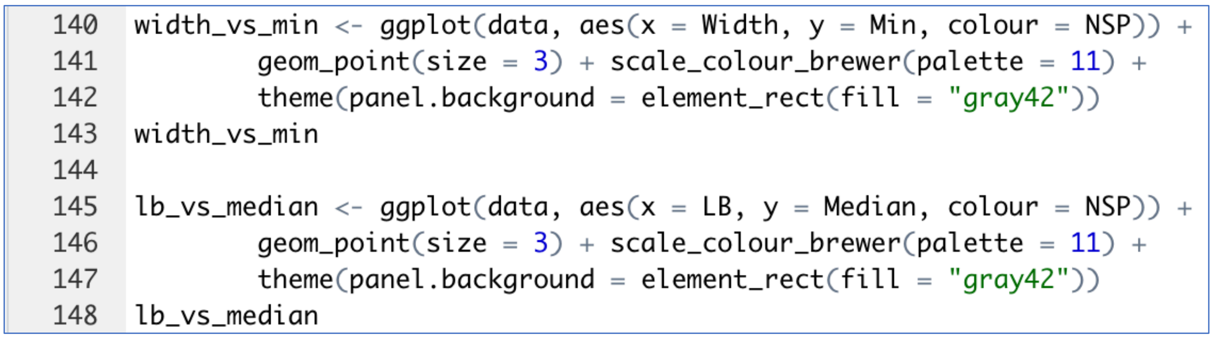


Figure 27: Correlation Between LB and Median Features

The R codes used for the above two visualizations are in Figure 28.



```
width_vs_min <- ggplot(data, aes(x = Width, y = Min, colour = NSP)) +
        geom_point(size = 3) + scale_colour_brewer(palette = 11) +
        theme(panel.background = element_rect(fill = "gray42"))
width_vs_min

lb_vs_median <- ggplot(data, aes(x = LB, y = Median, colour = NSP)) +
        geom_point(size = 3) + scale_colour_brewer(palette = 11) +
        theme(panel.background = element_rect(fill = "gray42"))
lb_vs_median
```

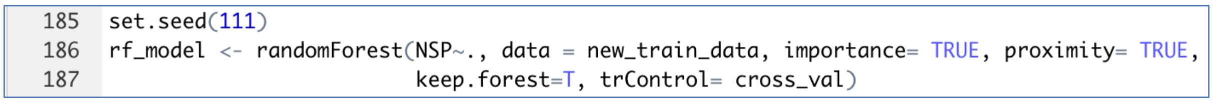
Figure 28: R Codes to Depict Correlation Between Independent Variables

4 Implementation

A purpose-based machine learning and deep learning algorithms were developed using "RStudio" with the necessary "R" packages and performance-effective tuning parameters that were needed to run models properly.

4.1 Implementation of Random Forest Model

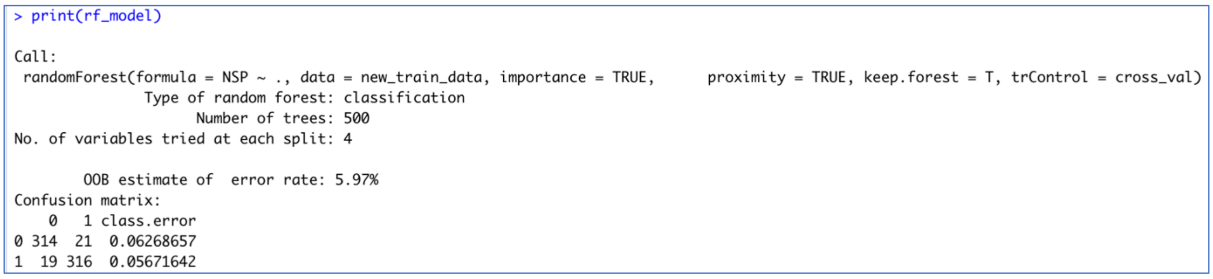
The random forest "RF" model was built along with some tuning parameters (refer Figure 29). The "set.seed" function was used for repeatability.



```
set.seed(111)
rf_model <- randomForest(NSP~., data = new_train_data, importance= TRUE, proximity= TRUE,
                         keep.forest=T, trControl= cross_val)
```

Figure 29: Random Forest Supervised Machine Learning Algorithm

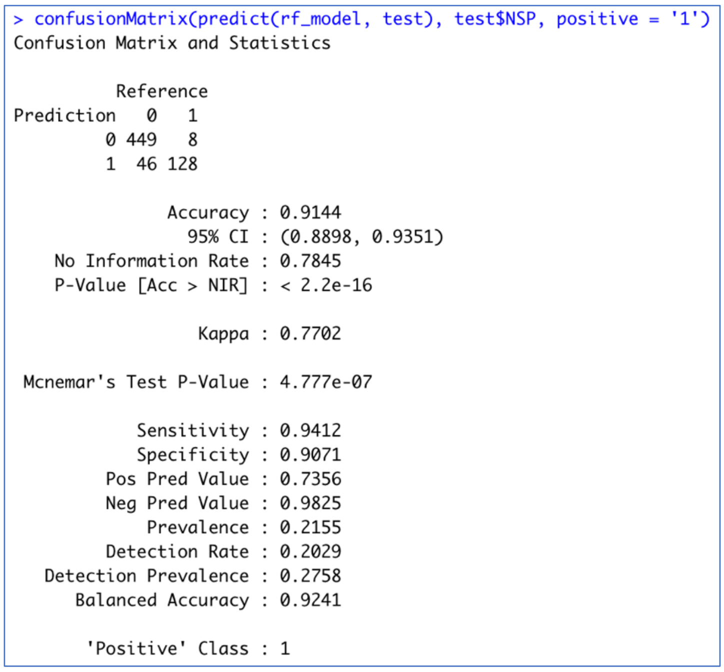
When printing RF developed model (refer Figure 30), it tells the model was built with 500 trees and the number of variables tried at each split "mtry = 4". Out of bag error estimate "OOB" is 5.97% which is quite good estimate. "OBB" data, which is not in bootstrap sample, was used to estimate the classification error for each bootstrap iteration. Confusion matrix in Figure 30 showed RF classifying class "1" at a less error of 5.6% compared to an error of 6.2% for class "0".



```
> print(rf_model)

Call:
 randomForest(formula = NSP ~ ., data = new_train_data, importance = TRUE,      proximity = TRUE, keep.forest = T, trControl = cross_val)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 5.97%
Confusion matrix:
    0   1 class.error
0 314  21  0.06268657
1  19 316  0.05671642
```

Figure 30: RF Model OOB Estimate Error

To evaluate the RF classification model, accuracy, recall and specificity were considered as main performance metrics. The confusion matrix based on test dataset returned RF model performing well at classifying abnormal fetuses represented by class "1" at 94.12% compared to 90.71% for normal fetuses represented by class "0" (refer Figure 31).

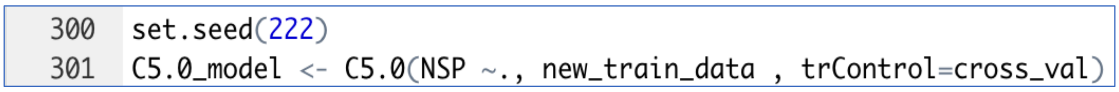


```
> confusionMatrix(predict(rf_model, test), test$NSP, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 449   8
         1  46 128

               Accuracy : 0.9144
                 95% CI : (0.8898, 0.9351)
    No Information Rate : 0.7845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7702

 Mcnemar's Test P-Value : 4.777e-07

            Sensitivity : 0.9412
            Specificity : 0.9071
         Pos Pred Value : 0.7356
         Neg Pred Value : 0.9825
             Prevalence : 0.2155
         Detection Rate : 0.2029
   Detection Prevalence : 0.2758
      Balanced Accuracy : 0.9241

       'Positive' Class : 1
```

Figure 31: RF Confusion Matrix and Statistics

4.2 Implementation of C5.0 Decision Tree Model

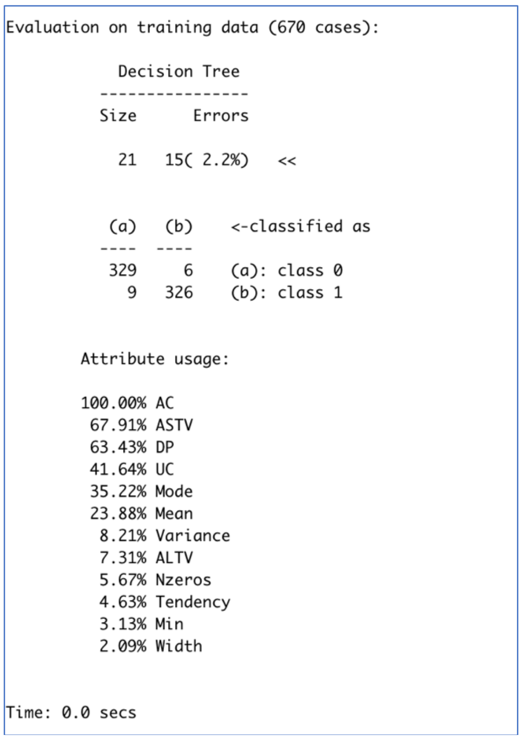
The model was created as shown in Figure 32.



```
set.seed(222)
C5.0_model <- C5.0(NSP ~., new_train_data , trControl=cross_val)
```

Figure 32: Building C5.0 Algorithm

A summary of C5.0 model (refer Figure 33) showed 21 decision trees are needed to classify abnormal foetuses.



```
Evaluation on training data (670 cases):

	    Decision Tree
	  ----------------
	  Size      Errors

	    21   15( 2.2%)   <<


	   (a)   (b)    <-classified as
	  ----  ----
	   329     6    (a): class 0
	     9   326    (b): class 1


	Attribute usage:

	100.00% AC
	 67.91% ASTV
	 63.43% DP
	 41.64% UC
	 35.22% Mode
	 23.88% Mean
	  8.21% Variance
	  7.31% ALTV
	  5.67% Nzeros
	  4.63% Tendency
	  3.13% Min
	  2.09% Width


Time: 0.0 secs
```

Figure 33: A Summary of C5.0 Algorithm

According to C5.0 algorithm, (AC) feature is the most important variable for the tree to be split at (refer Figure 34).

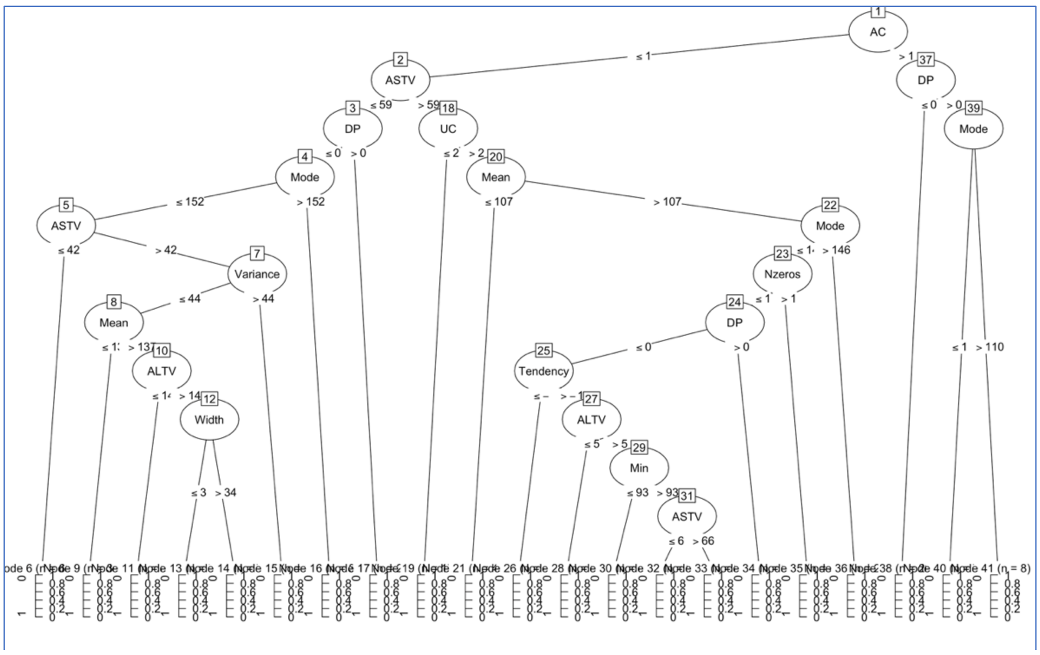
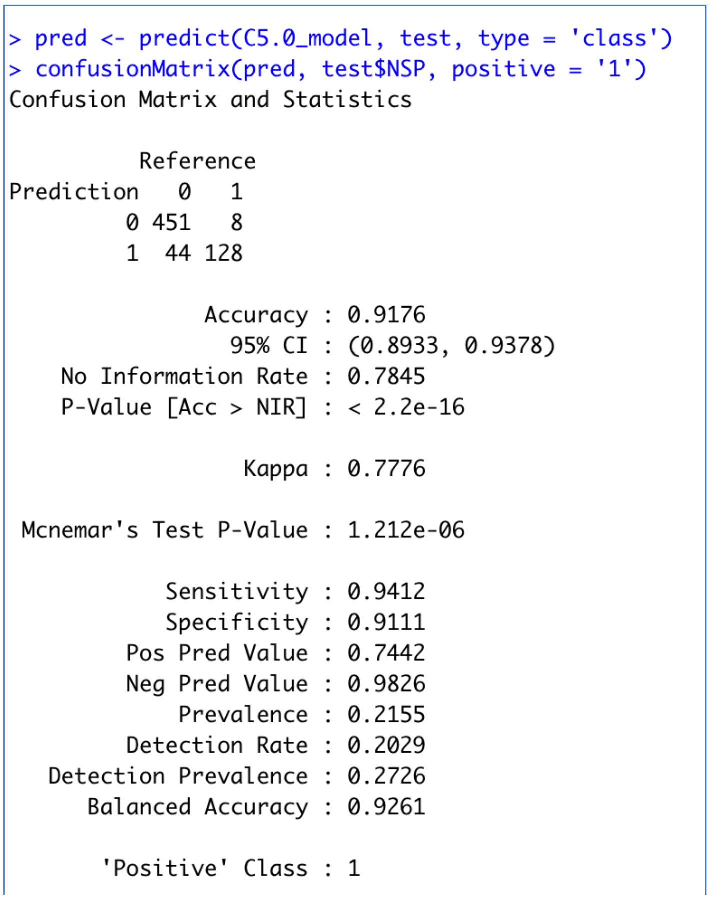


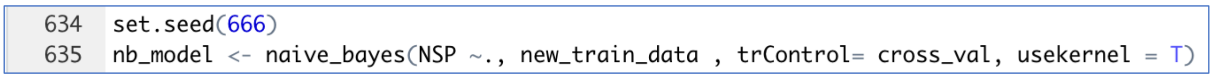
Figure 34: Visualisation of C5.0 Decision Tree Algorithm

The model performed better (refer Figure 35) at classifying abnormal foetuses at sensitivity of 94.12% compared to 91.11% specificity for classifying normal foetuses.


```
> pred <- predict(C5.0_model, test, type = 'class')
> confusionMatrix(pred, test$NSP, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 451   8
         1  44 128

               Accuracy : 0.9176
                 95% CI : (0.8933, 0.9378)
    No Information Rate : 0.7845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7776

 Mcnemar's Test P-Value : 1.212e-06

            Sensitivity : 0.9412
            Specificity : 0.9111
         Pos Pred Value : 0.7442
         Neg Pred Value : 0.9826
             Prevalence : 0.2155
         Detection Rate : 0.2029
   Detection Prevalence : 0.2726
      Balanced Accuracy : 0.9261

       'Positive' Class : 1
```
Figure 35: C5.0 Decision Tree confusion Matrix and Statistics

4.3 Implementation of Naïve Bayes Model

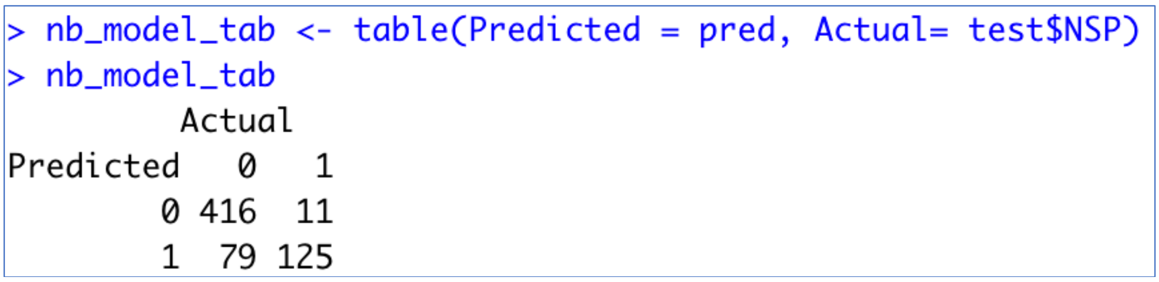
The NB model was built as shown in Figure 36.



```
set.seed(666)
nb_model <- naive_bayes(NSP ~., new_train_data , trControl= cross_val, usekernel = T)
```

Figure 36: Developing NB Model

Accuracy, recall and specificity were calculated from the resulted classification table (refer Figure 37) showing NB classified abnormal foetuses at a recall of 92%.



```
> nb_model_tab <- table(Predicted = pred, Actual= test$NSP)
> nb_model_tab
         Actual
Predicted   0   1
        0 416  11
        1  79 125
```

Figure 37: NB Confusion Matrix

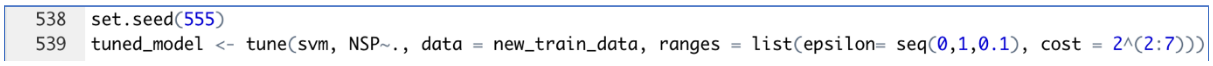
$$Acc = \frac{(125 + 416)}{(125 + 416 + 11 + 79)} = 0.8573693$$

$$Recall = \frac{125}{(125 + 11)} = 0.9191176$$

$$Specificity = \frac{416}{(416 + 79)} = 0.840404$$

4.4 Implementation of Support Vector Machine Model

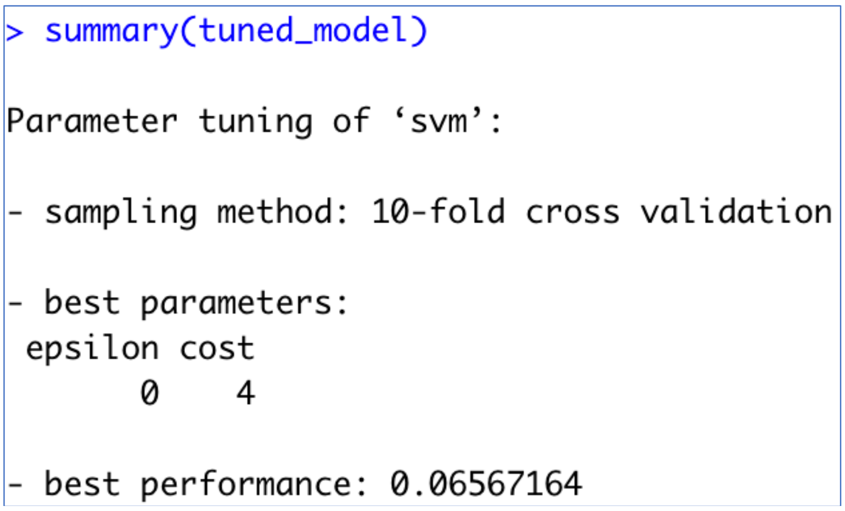
To get the best Support Vector Machine (SVM) model with the best kernel and the best parameter "epsilon" and "cost" values, a tuned SVM model was built with a sequence of numbers, starting from 0 and goes up to 1 with an increment of 0.1 for "epsilon" parameter. To avoid either overfitting or underfitting, then a large range of numbers was used to capture the optimal value for "cost" parameter (refer Figure 38).



```
set.seed(555)
tuned_model <- tune(svm, NSP~., data = new_train_data, ranges = list(epsilon= seq(0,1,0.1), cost = 2^(2:7)))
```

Figure 38: Building Tuned SVM Model to Get the Best SVM Model

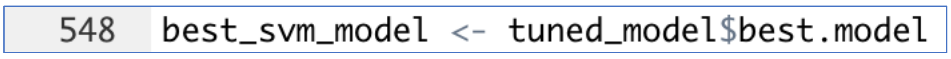
The tuned SVM model returned the value of "0" is the best "epsilon" parameter and the value of "4" is the best for "cost" as shown in Figure 39.



```
> summary(tuned_model)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 epsilon cost
       0    4

- best performance: 0.06567164
```

Figure 39: Summary of SVM Tuned Model

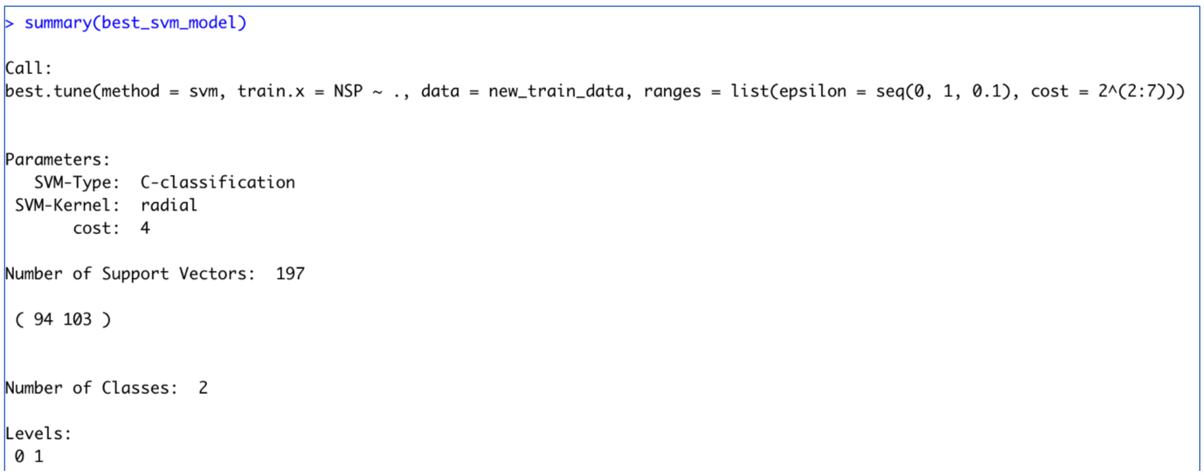
Then, the best SVM model was obtained from the tuned SVM one (refer Figure 40)



```
best_svm_model <- tuned_model$best.model
```

Figure 40: The Best SVM Model

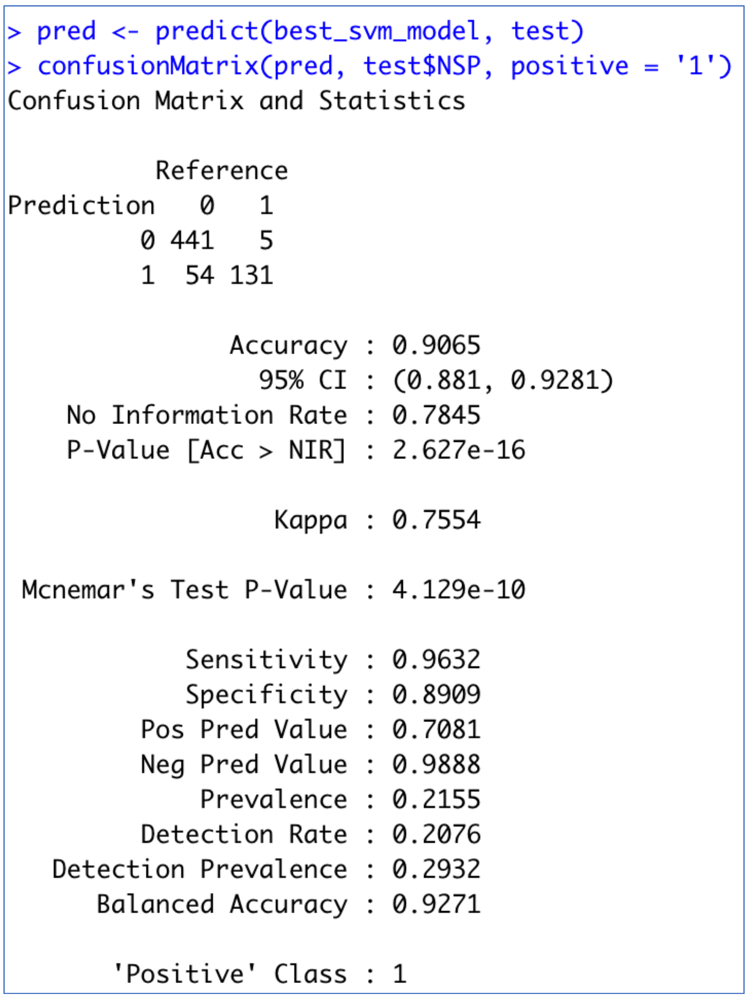
The best SVM model (refer Figure 41) with a best kernel "radial", "cost" = 4 and "epsilon" = 0 generated 197 support vectors, 94 vectors are for class 0 and 103 vectors for class 1.



```
> summary(best_svm_model)

Call:
best.tune(method = svm, train.x = NSP ~ ., data = new_train_data, ranges = list(epsilon = seq(0, 1, 0.1), cost = 2^(2:7)))


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  4

Number of Support Vectors:  197

 ( 94 103 )


Number of Classes:  2

Levels:
 0 1
```

Figure 41: Summary of Best SVM Model

The confusion matrix returned the best SVM recorded a recall of 96.32% at classifying abnormal foetuses (refer Figure 42).

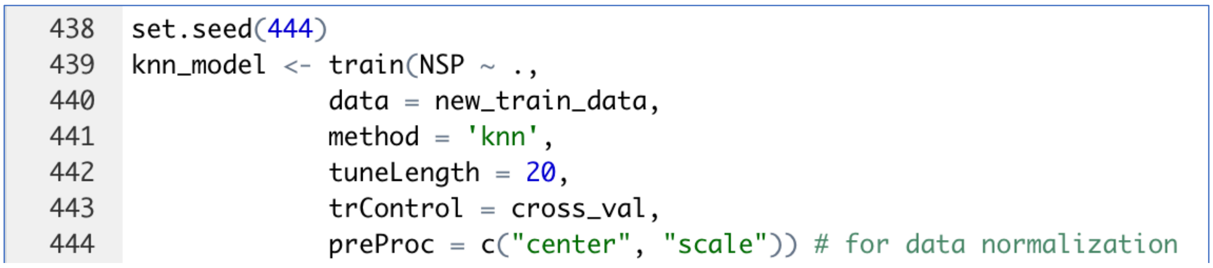


```
> pred <- predict(best_svm_model, test)
> confusionMatrix(pred, test$NSP, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 441   5
         1  54 131

               Accuracy : 0.9065
                 95% CI : (0.881, 0.9281)
    No Information Rate : 0.7845
    P-Value [Acc > NIR] : 2.627e-16

                  Kappa : 0.7554

 Mcnemar's Test P-Value : 4.129e-10

            Sensitivity : 0.9632
            Specificity : 0.8909
         Pos Pred Value : 0.7081
         Neg Pred Value : 0.9888
             Prevalence : 0.2155
         Detection Rate : 0.2076
   Detection Prevalence : 0.2932
      Balanced Accuracy : 0.9271

       'Positive' Class : 1
```

Figure 42: Best SVM Model Confusion Matrix

4.5 Implementation of K-Nearest Neighbour

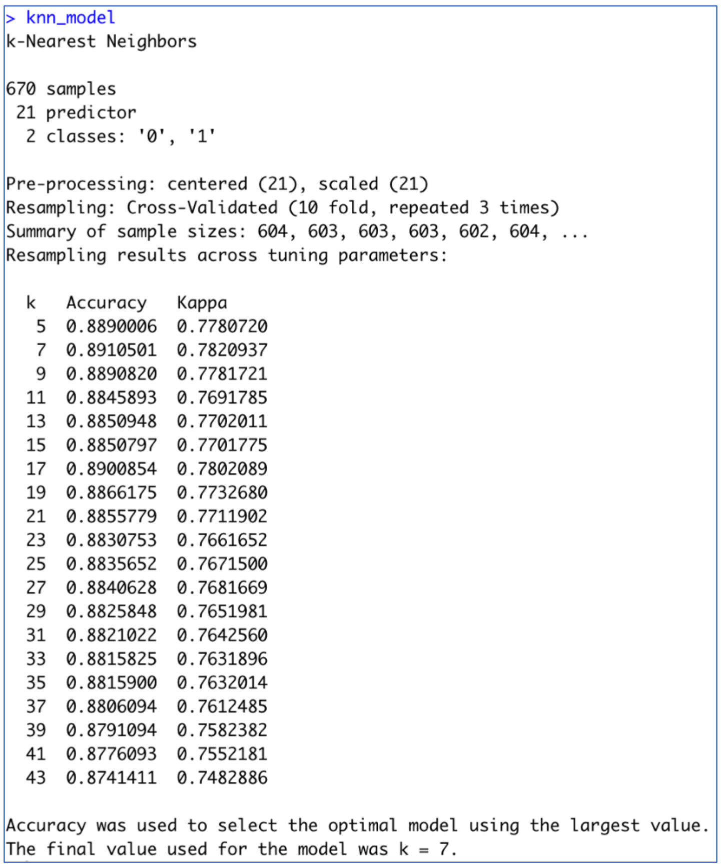
K-Nearest Neighbour (KNN) model was created (refer Figure 43) with performance-effective tuning parameters. Since there are high and low data points, a normalization technique was applied when building the model.



```
set.seed(444)
knn_model <- train(NSP ~ .,
                   data = new_train_data,
                   method = 'knn',
                   tuneLength = 20,
                   trControl = cross_val,
                   preProc = c("center", "scale")) # for data normalization
```

Figure 43: Building KNN Algorithm

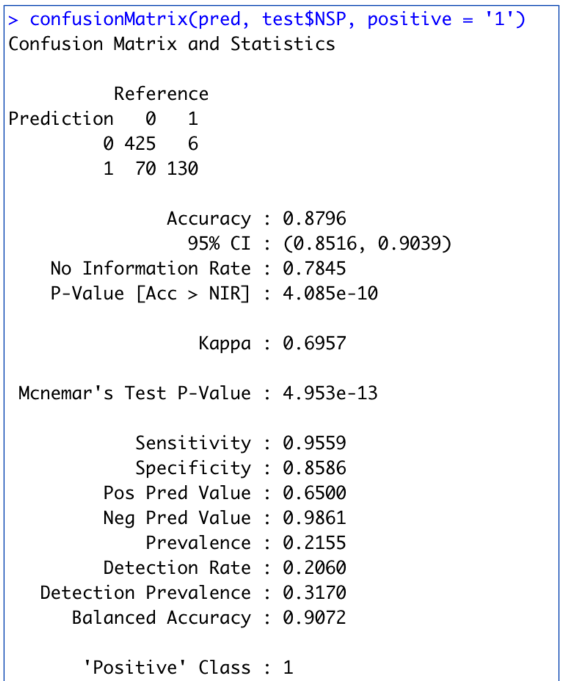
The model reverted the optimal value for "K" is 7 based on accuracy as a performance metric (refer Figure 44).



```
> knn_model
k-Nearest Neighbors

670 samples
 21 predictor
  2 classes: '0', '1'

Pre-processing: centered (21), scaled (21)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 604, 603, 603, 603, 602, 604, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.8890006  0.7780720
   7  0.8910501  0.7820937
   9  0.8890820  0.7781721
  11  0.8845893  0.7691785
  13  0.8850948  0.7702011
  15  0.8850797  0.7701775
  17  0.8900854  0.7802089
  19  0.8866175  0.7732680
  21  0.8855779  0.7711902
  23  0.8830753  0.7661652
  25  0.8835652  0.7671500
  27  0.8840628  0.7681669
  29  0.8825848  0.7651981
  31  0.8821022  0.7642560
  33  0.8815825  0.7631896
  35  0.8815900  0.7632014
  37  0.8806094  0.7612485
  39  0.8791094  0.7582382
  41  0.8776093  0.7552181
  43  0.8741411  0.7482886

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 7.
```

Figure 44: Optimal Value for K

The KNN confusion matrix (refer Figure 45) showed KNN well classified abnormal foetuses at recall of 96% compared to classifying the opposite class at a specificity of 86%.

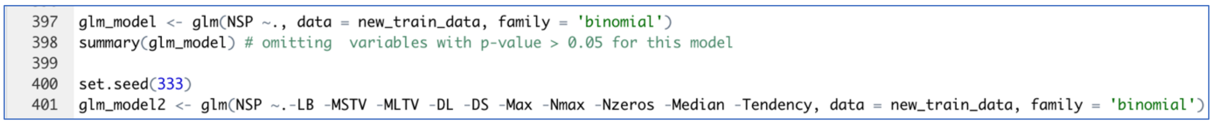


```
> confusionMatrix(pred, test$NSP, positive = '1')
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 425   6
         1  70 130

               Accuracy : 0.8796
                 95% CI : (0.8516, 0.9039)
    No Information Rate : 0.7845
    P-Value [Acc > NIR] : 4.085e-10

                  Kappa : 0.6957

 Mcnemar's Test P-Value : 4.953e-13

            Sensitivity : 0.9559
            Specificity : 0.8586
         Pos Pred Value : 0.6500
         Neg Pred Value : 0.9861
             Prevalence : 0.2155
         Detection Rate : 0.2060
   Detection Prevalence : 0.3170
      Balanced Accuracy : 0.9072

       'Positive' Class : 1
```

Figure 45: KNN Confusion Matrix

4.6 Implementation of Generalized Linear Model

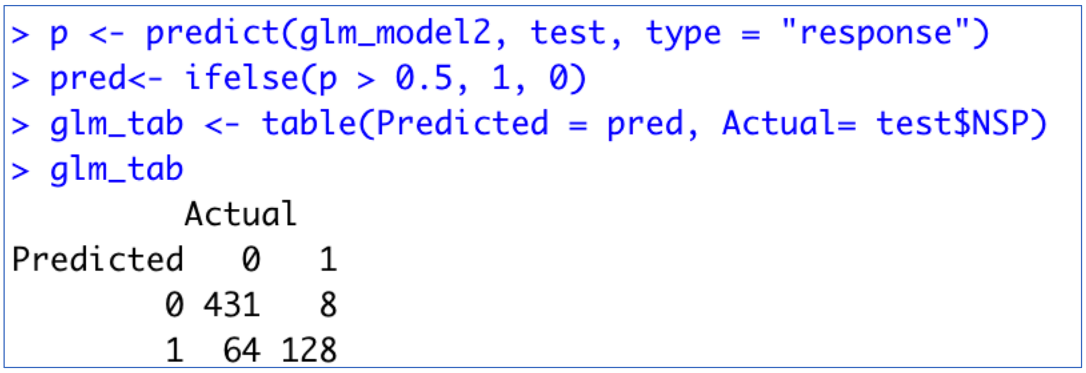
Generalized Linear Model (GLM) was built with "binomial" family as shown in the following Figure 46.



```
glm_model <- glm(NSP ~., data = new_train_data, family = 'binomial')
summary(glm_model) # omitting  variables with p-value > 0.05 for this model

set.seed(333)
glm_model2 <- glm(NSP ~.-LB -MSTV -MLTV -DL -DS -Max -Nmax -Nzeros -Median -Tendency, data = new_train_data, family = 'binomial')
```

Figure 46: Generalised Linear Model

A table matrix (refer Figure 47) was followed to evaluate the model's performance where the model returned less misclassification errors for abnormal foetuses compare to normal ones.



```
> p <- predict(glm_model2, test, type = "response")
> pred<- ifelse(p > 0.5, 1, 0)
> glm_tab <- table(Predicted = pred, Actual= test$NSP)
> glm_tab
         Actual
Predicted   0   1
        0 431   8
        1  64 128
```

Figure 47: Generalised Linear Model Table Matrix

$$Acc = \frac{(128 + 431)}{(128 + 431 + 8 + 64)} = 0.8858954$$

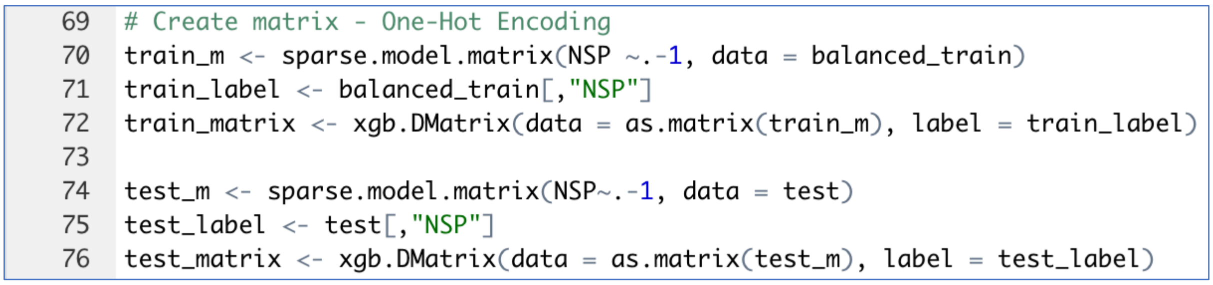
$$Recall = 128\frac{1}{(128 + 8)} = 0.9411765$$

$$Specificity = 431\frac{1}{(431 + 64)} = 0.8707071$$

4.7 Implementation of Extreme Gradient Boosting Model

As a preparing stage, train and test datasets were converted to a matrix form using one hot encoding technique to build Extreme Gradient Boosting (XGBoost) algorithm (refer Figure 48). The R codes for XGBoost algorithm were taken from prof. Bharatendra Rai's online machine learning and deep learning tutorials[1] with some modifications made by me.

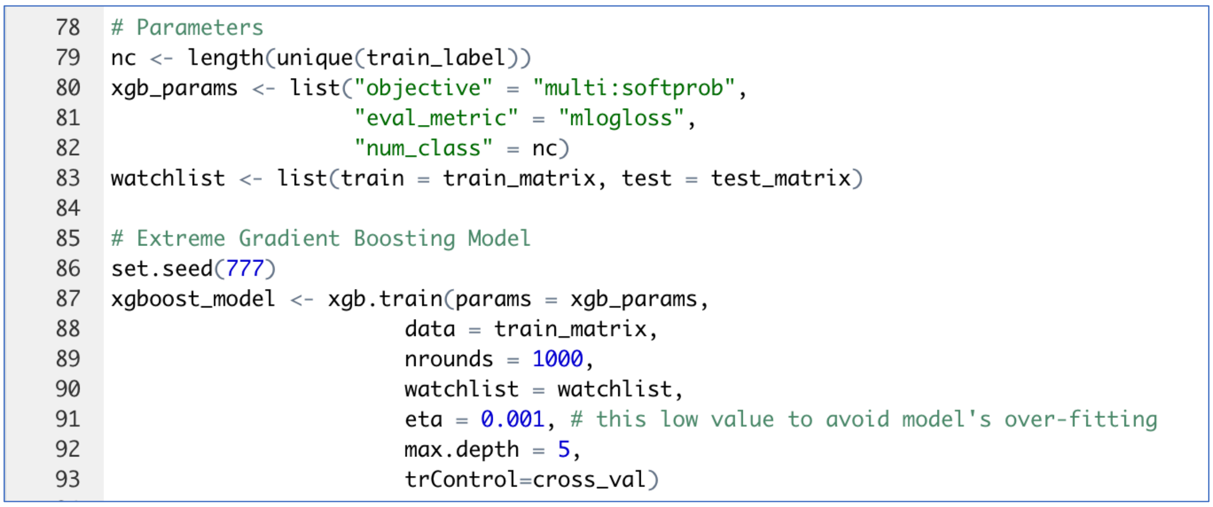
[1] https://www.youtube.com/watch?v=woVTNwRrFHE&t=1058s



```r
# Create matrix - One-Hot Encoding
train_m <- sparse.model.matrix(NSP ~.-1, data = balanced_train)
train_label <- balanced_train[,"NSP"]
train_matrix <- xgb.DMatrix(data = as.matrix(train_m), label = train_label)

test_m <- sparse.model.matrix(NSP~.-1, data = test)
test_label <- test[,"NSP"]
test_matrix <- xgb.DMatrix(data = as.matrix(test_m), label = test_label)
```

Figure 48: One-Hot Encoding to Convert Train and Test Datasets to Matrix Form

Additionally, effectively selected parameters were applied to build the model and to support the model's performance (refer Figure 49). Parameter "num_class" is the number of classes in response variable, which is 2 in this research case. The "max.depth" means the maximum tree depth which takes a default value of 6. This parameter was changed to 5 for better abnormal foetuses' classification. Most importantly is the value of "eta", low values prevent the model's overfitting and vice versa. The last parameter is a 10-fold cross validation.



```r
# Parameters
nc <- length(unique(train_label))
xgb_params <- list("objective" = "multi:softprob",
                   "eval_metric" = "mlogloss",
                   "num_class" = nc)
watchlist <- list(train = train_matrix, test = test_matrix)

# Extreme Gradient Boosting Model
set.seed(777)
xgboost_model <- xgb.train(params = xgb_params,
                           data = train_matrix,
                           nrounds = 1000,
                           watchlist = watchlist,
                           eta = 0.001, # this low value to avoid model's over-fitting
                           max.depth = 5,
                           trControl=cross_val)
```

Figure 49: Building Extreme Gradient Boosting Model

Feature importance, based on "Gain" values, was obtained from the developed XGBoost model. It is obvious that "ASTV" is the most important independent feature (refer Figure 50).

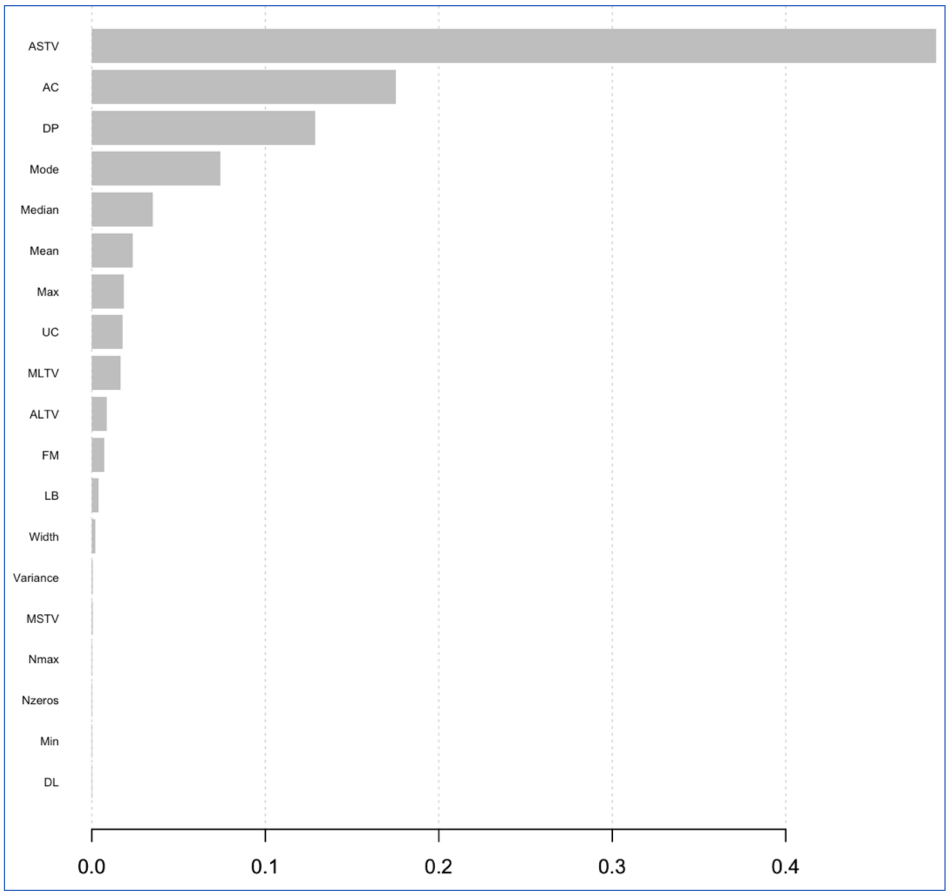
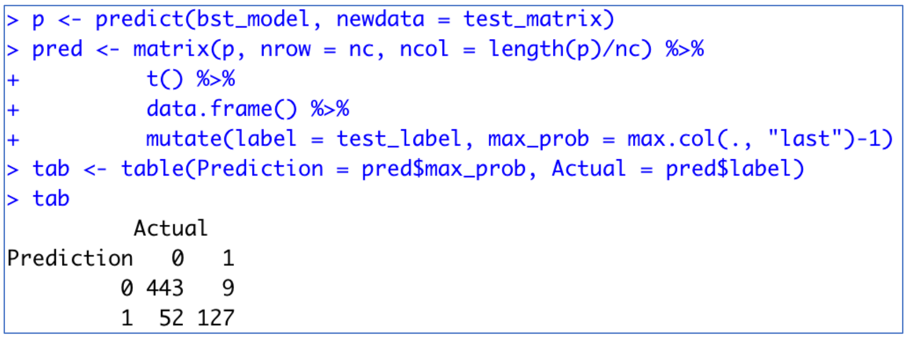


Figure 50: Feature Importance Based on XGBoost Gini Values

The XGBoost model's performance was evaluated by a confusion matrix (refer Figure 51) where the model classified abnormal class at recall of 93% and an accuracy of 90%.



```
> p <- predict(bst_model, newdata = test_matrix)
> pred <- matrix(p, nrow = nc, ncol = length(p)/nc) %>%
+          t() %>%
+          data.frame() %>%
+          mutate(label = test_label, max_prob = max.col(., "last")-1)
> tab <- table(Prediction = pred$max_prob, Actual = pred$label)
> tab
          Actual
Prediction   0   1
         0 443   9
         1  52 127
```

Figure 51: XGBoost Confusion Matrix

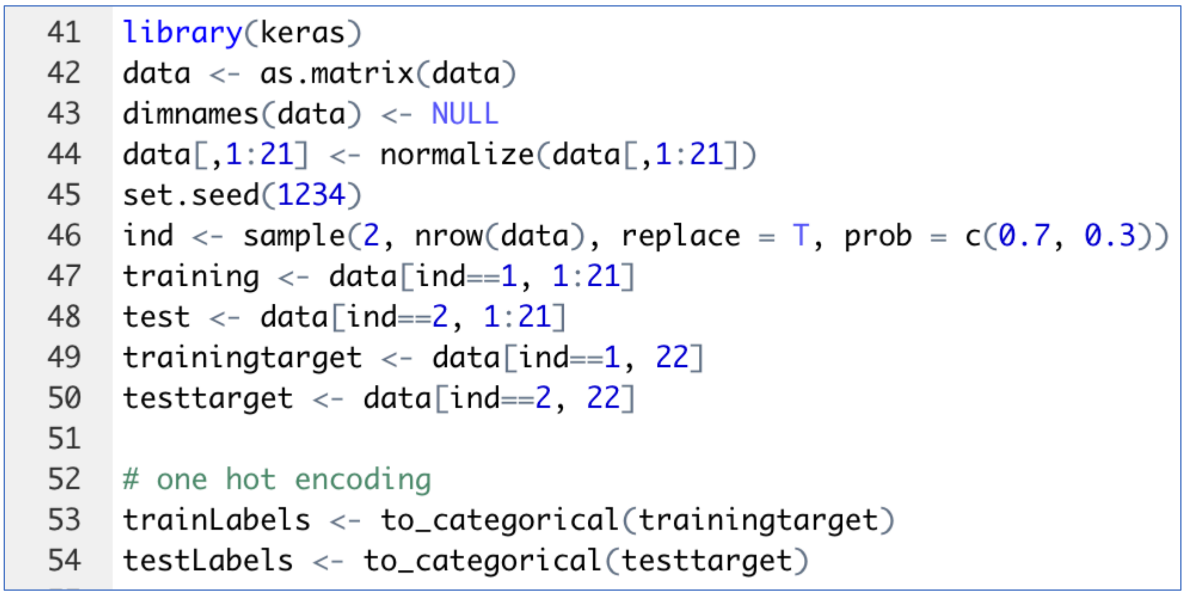
$$Acc = \frac{(127 + 443)}{(127 + 443 + 9 + 52)} = 0.9033281$$

$$Recall = \frac{127}{(127 + 9)} = 0.9338235$$

$$Specificity = \frac{443}{(443 + 52)} = 0.8949495$$

4.8 Implementation of Deep Learning Multilayer Perceptron Neural Networks

Multiple Multilayer Perceptron Neural Networks (MLPNNs) with different hidden layers were applied to select the best performing MLPNN regarding classification of abnormal foetuses. Figure 52 shows that data has been transformed to matrix form and all data points were normalized. Data then partitioned to 70% as train data and 30% as test data. The response variable in both train and test data was converted to categorical type using one hot encoding technique. The R codes for XGBoost algorithm were taken from prof. Bharatendra Rai's online machine learning and deep learning tutorials[1] with some modifications made by me.



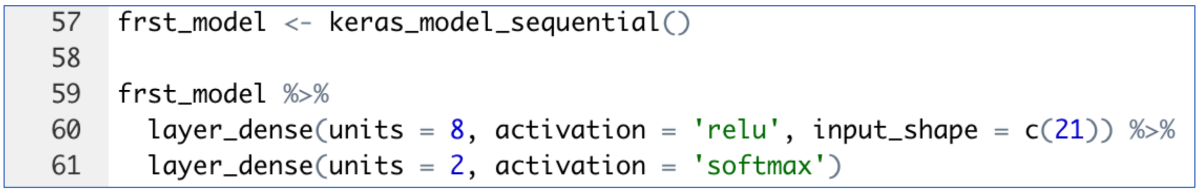
```r
library(keras)
data <- as.matrix(data)
dimnames(data) <- NULL
data[,1:21] <- normalize(data[,1:21])
set.seed(1234)
ind <- sample(2, nrow(data), replace = T, prob = c(0.7, 0.3))
training <- data[ind==1, 1:21]
test <- data[ind==2, 1:21]
trainingtarget <- data[ind==1, 22]
testtarget <- data[ind==2, 22]

# one hot encoding
trainLabels <- to_categorical(trainingtarget)
testLabels <- to_categorical(testtarget)
```

Figure 52: Data Preparation for MLPNN Algorithms

4.8.1 First MLPNN Model

The first MLPNN model was built with one hidden layer bearing 8 neurons (refer Figure 53). "input_shape = 21" as we have 21 independent variables and "units = 2" because there are two different groups in NSP.

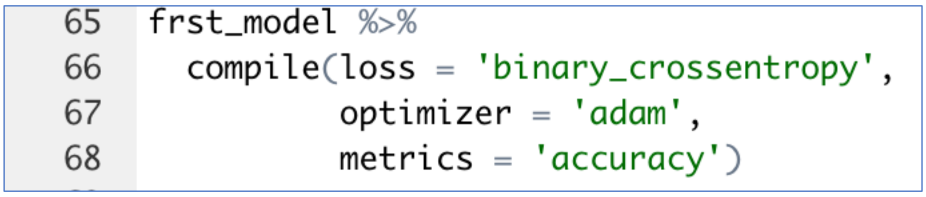
[1] https://www.youtube.com/watch?v=hd81EH1g1bE&t=811s



```r
frst_model <- keras_model_sequential()

frst_model %>%
  layer_dense(units = 8, activation = 'relu', input_shape = c(21)) %>%
  layer_dense(units = 2, activation = 'softmax')
```

Figure 53: The First MLPNN Model

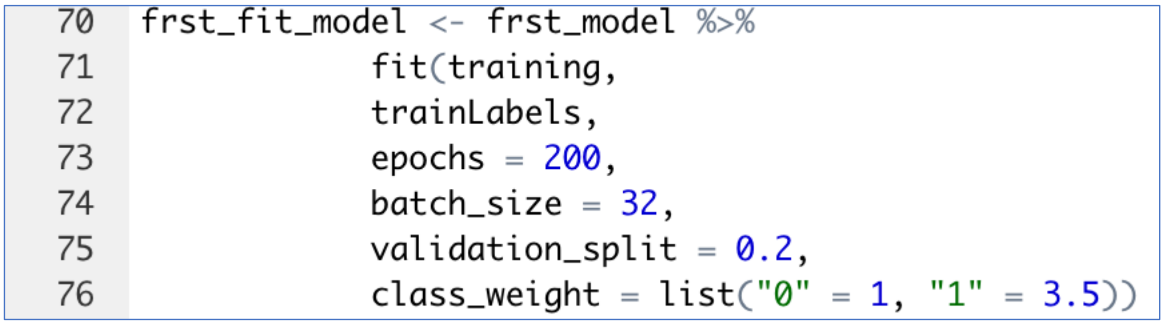
The first model was compiled by "compile" function from "Keras" package (refer Figure 54). The "adam" optimization algorithm was used for "optimizer" as it is a popular algorithm in deep learning field.



```r
frst_model %>%
  compile(loss = 'binary_crossentropy',
          optimizer = 'adam',
          metrics = 'accuracy')
```

Figure 54: The First MLPNN Model Compile

The first model was fitted with a normalized train dataset and performance-enhancing parameters as shown in Figure 55. The "epochs = 200" is the number of iterations, "batch_size = 32" refers to the number of samples that be used per gradient. To resolve class imbalance problem, "class_weight" parameter was used with a weight for class "0" = 1 referring to 1655 instances of normal foetuses and a weight for abnormal foetuses' class "1" = 3.5 resulting from 1655 divided by 471 which is the number of instances for abnormal foetuses' class.



```r
frst_fit_model <- frst_model %>%
              fit(training,
              trainLabels,
              epochs = 200,
              batch_size = 32,
              validation_split = 0.2,
              class_weight = list("0" = 1, "1" = 3.5))
```

Figure 55: The First MLPNN Model Fit

As the first model is running, the two plots (refer Figure 56) were seen. First plot explains the loss based on training data "blue line" and the loss based on 20% validation data "green line". Ideally, as training loss decreases, validation loss should decrease, otherwise there is an overfitting problem. The second plot is for accuracy where after about 30 iterations, model's accuracy started to increase.

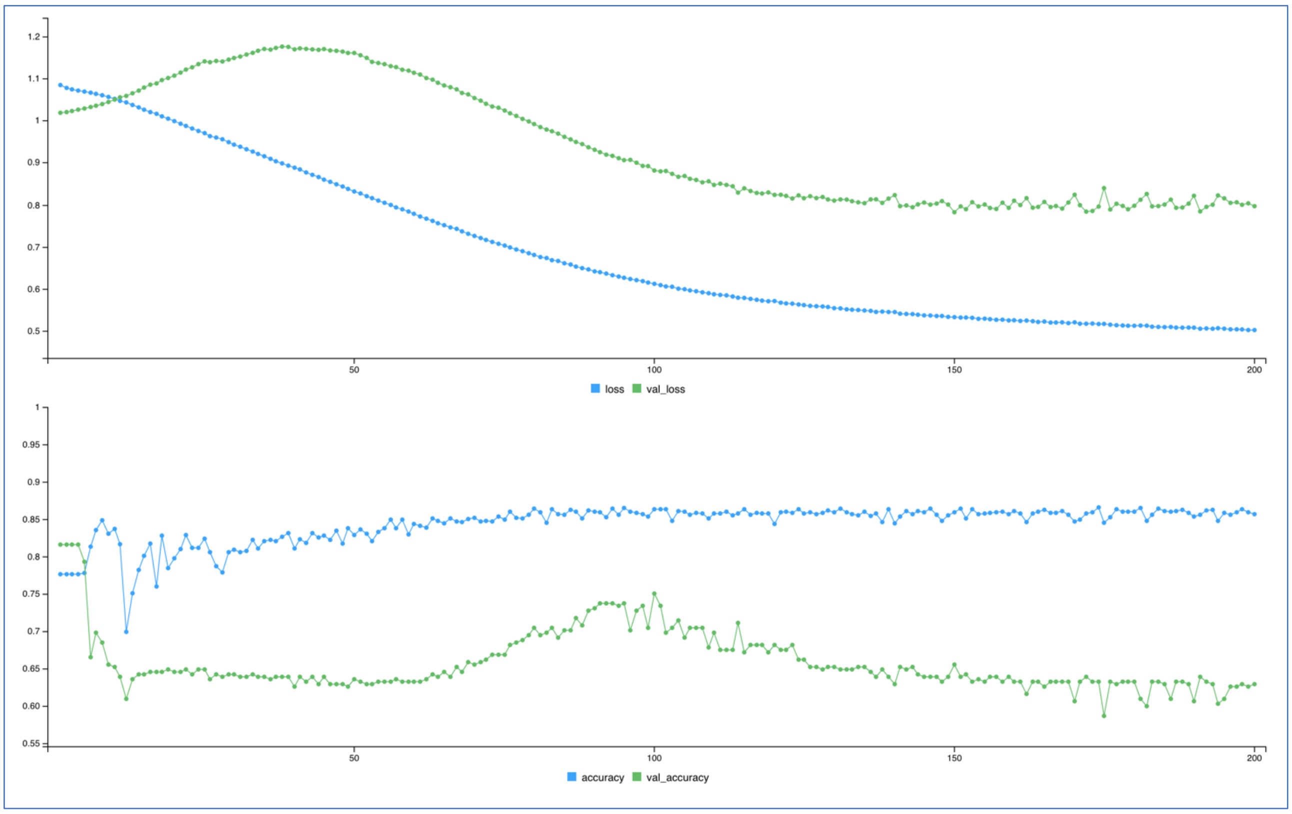
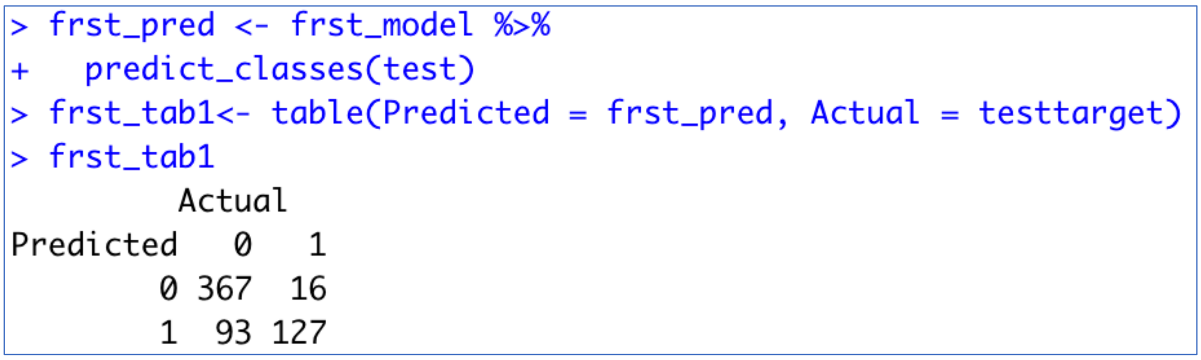


Figure 56: The First MLPNN Model's Performance

The following confusion matrix based on test dataset (refer Figure 57) was used to evaluate the first MLPNN model's performance where abnormal foetuses were classified at a recall of 89%.



```
> frst_pred <- frst_model %>%
+   predict_classes(test)
> frst_tab1<- table(Predicted = frst_pred, Actual = testtarget)
> frst_tab1
         Actual
Predicted   0   1
        0 367  16
        1  93 127
```

Figure 57: The First MLPNN Model Confusion Matrix

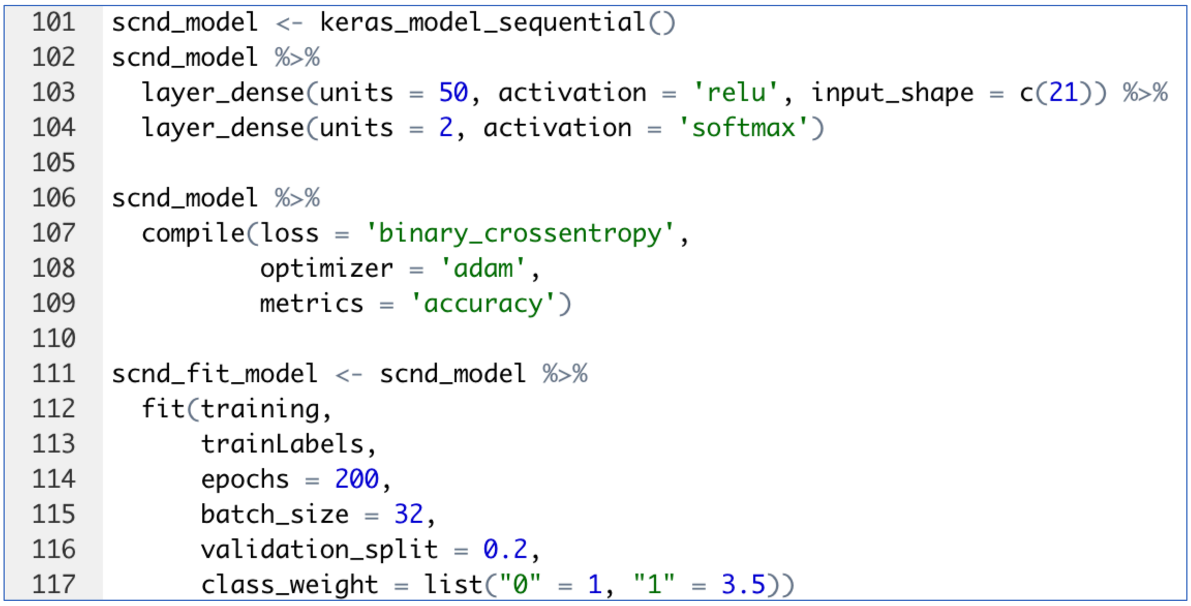
$$Acc = \frac{(127 + 367)}{(127 + 367 + 16 + 93)} = \ 0.8192371$$

$$Recall = \frac{127}{(127 + 16)} = \ 0.8881119$$

$$Specificity = \frac{367}{(367 + 93)} = \ 0.7978261$$

4.8.2 Second MPNN Model

The second model was built with the same procedures mentioned above with one hidden layer bearing 50 neurons (refer Figure 58).



```r
scnd_model <- keras_model_sequential()
scnd_model %>%
  layer_dense(units = 50, activation = 'relu', input_shape = c(21)) %>%
  layer_dense(units = 2, activation = 'softmax')

scnd_model %>%
  compile(loss = 'binary_crossentropy',
          optimizer = 'adam',
          metrics = 'accuracy')

scnd_fit_model <- scnd_model %>%
  fit(training,
      trainLabels,
      epochs = 200,
      batch_size = 32,
      validation_split = 0.2,
      class_weight = list("0" = 1, "1" = 3.5))
```

Figure 58: Create, Compile and Fit Second MLPNN Model

Loss and accuracy plots for the second model are shown in Figure 59. Compared to first model's accuracy, second plot shows accuracy increased after about 15 iterations.

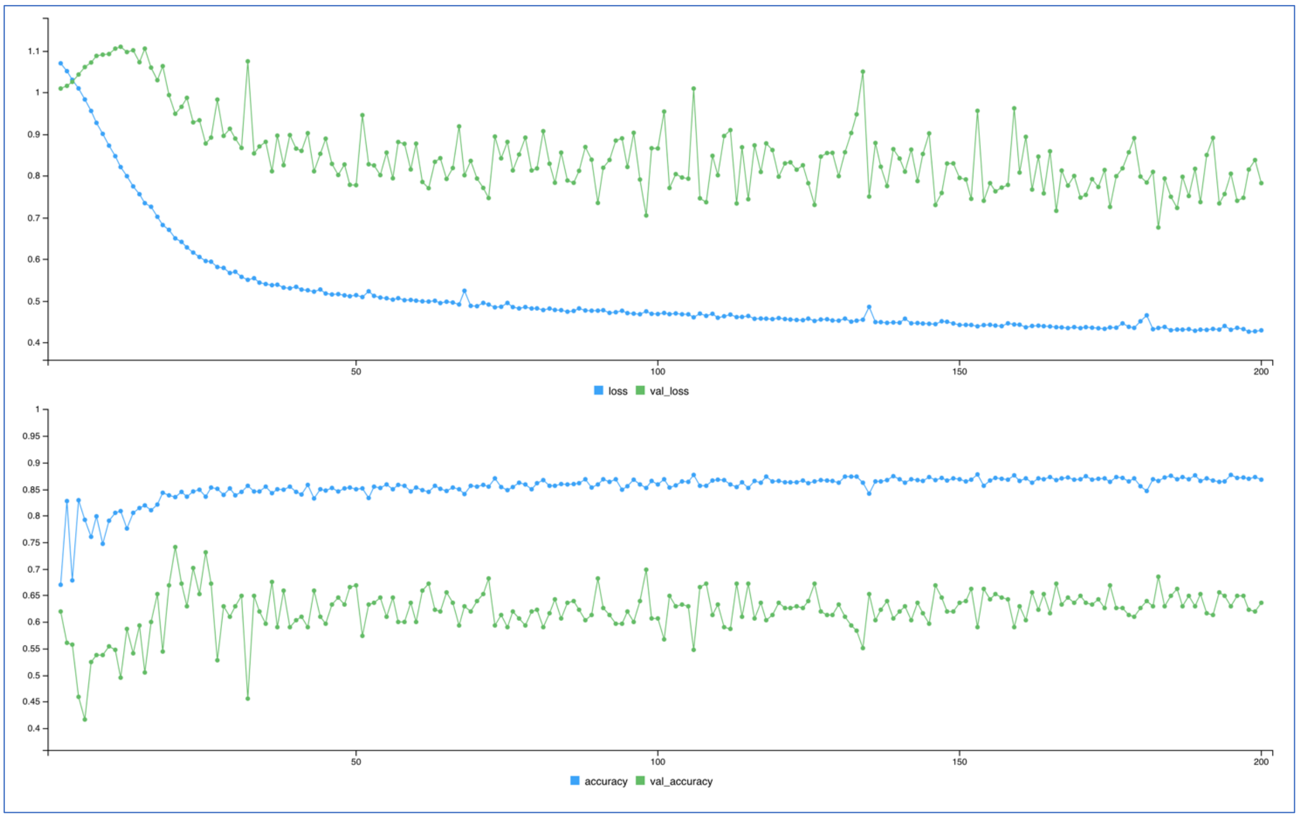
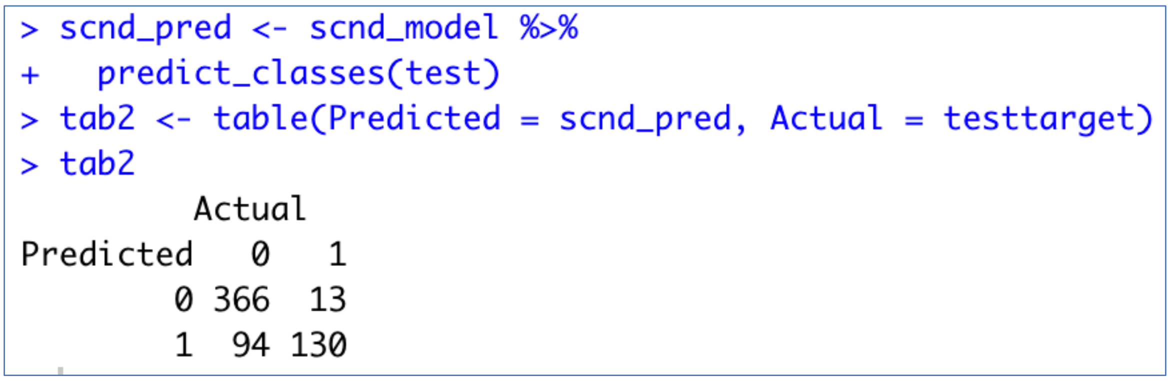


Figure 59: The Second MLPNN Model's Performance

Confusion matrix of the second model based on test dataset is shown in Figure 60 where the model classified abnormal foetuses at a recall of 91%.



```
> scnd_pred <- scnd_model %>%
+   predict_classes(test)
> tab2 <- table(Predicted = scnd_pred, Actual = testtarget)
> tab2
         Actual
Predicted   0   1
        0 366  13
        1  94 130
```

Figure 60: The Second MLPNN Model Confusion Matrix

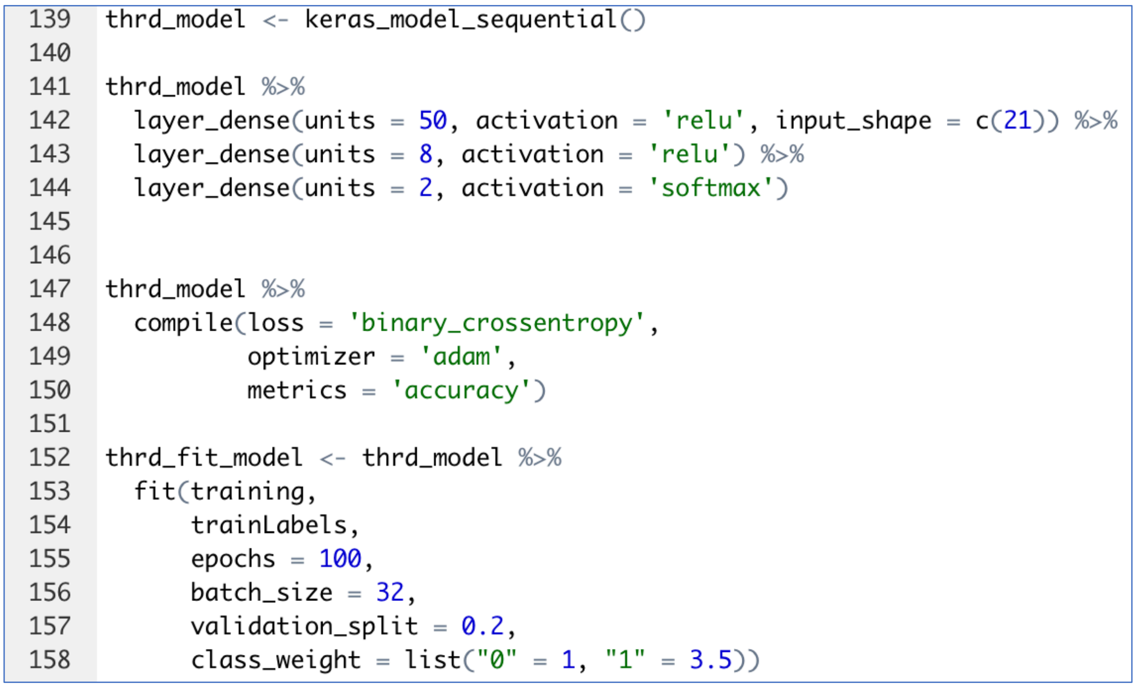
$$Acc = \frac{(130 + 366)}{(130 + 366 + 13 + 94)} = \ 0.8225539$$

$$Recall = \frac{130}{(130 + 13)} = \ 0.9090909$$

$$Specificity = \frac{366}{(366 + 94)} = \ 0.7956522$$

4.8.3 Third MPNN Model

The third MLPNN model was built by combining two different hidden layers that were used in the first and second models to enhance the classification performance (refer Figure 61).



```
thrd_model <- keras_model_sequential()

thrd_model %>%
  layer_dense(units = 50, activation = 'relu', input_shape = c(21)) %>%
  layer_dense(units = 8, activation = 'relu') %>%
  layer_dense(units = 2, activation = 'softmax')


thrd_model %>%
  compile(loss = 'binary_crossentropy',
          optimizer = 'adam',
          metrics = 'accuracy')

thrd_fit_model <- thrd_model %>%
  fit(training,
      trainLabels,
      epochs = 100,
      batch_size = 32,
      validation_split = 0.2,
      class_weight = list("0" = 1, "1" = 3.5))
```

Figure 61: Create, Compile and Fit Third MLPNN Model

Compared to first and second MLPNN model, the third MLPNN model showed accuracy increased after about 5 iterations (refer Figure 62)

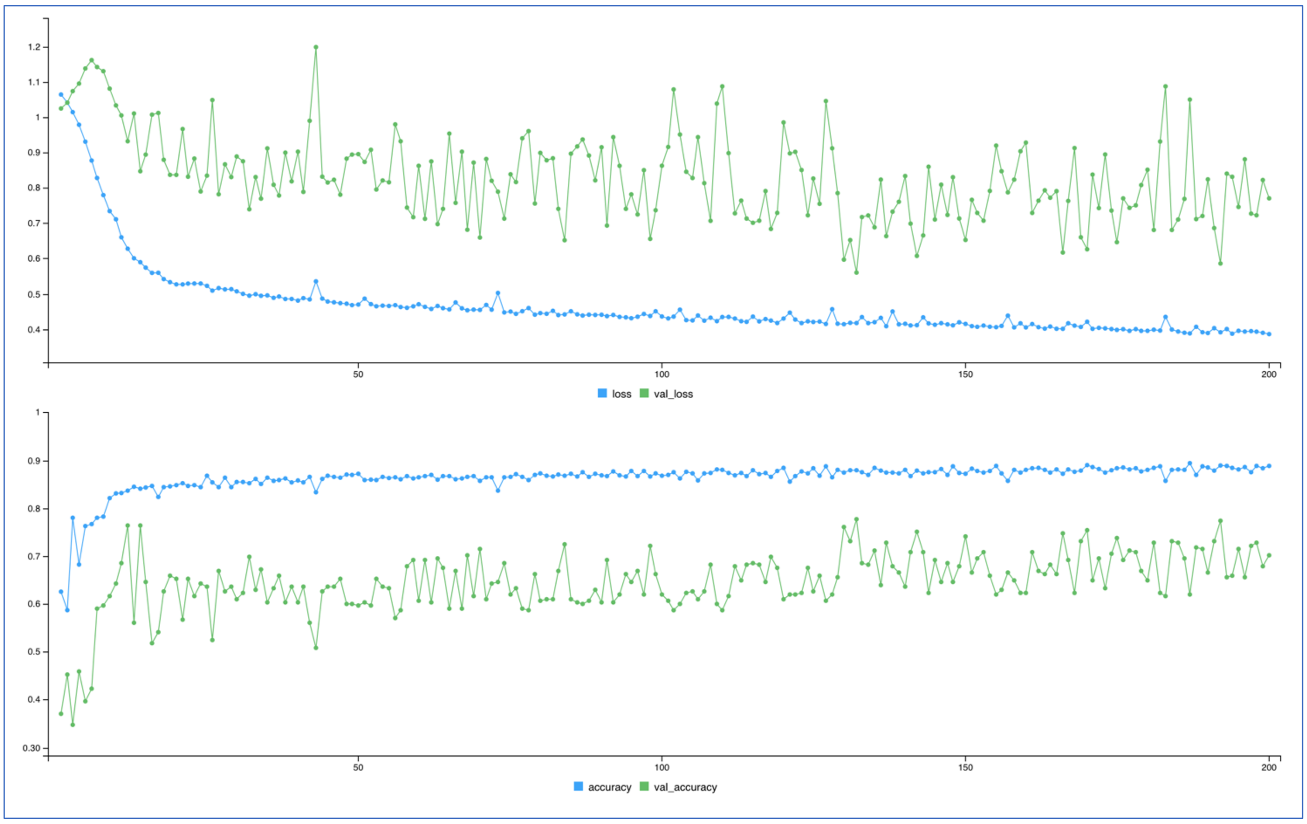
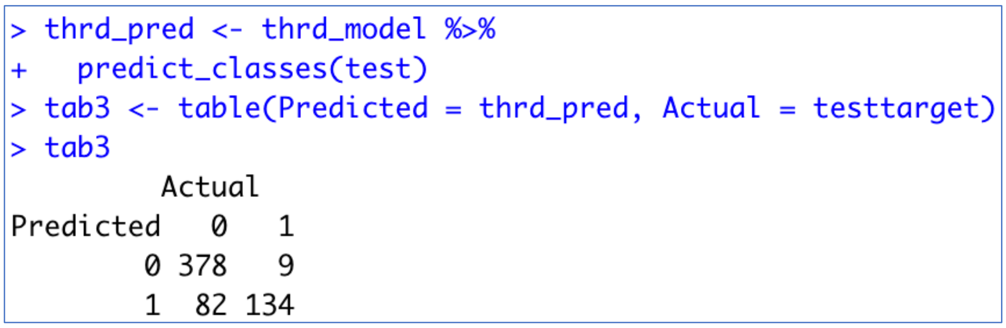


Figure 62: The Third MLPNN Model's Performance

Confusion matrix for the third MLPNN model based on test dataset is shown in Figure 63 where abnormal foetuses were classified at a higher recall of 94%.



```
> thrd_pred <- thrd_model %>%
+   predict_classes(test)
> tab3 <- table(Predicted = thrd_pred, Actual = testtarget)
> tab3
         Actual
Predicted   0   1
        0 378   9
        1  82 134
```

Figure 63: The Third MLPNN Model Confusion Matrix

$$Acc = \frac{(134 + 378)}{(134 + 378 + 9 + 82)} = 0.8490879$$

$$Recall = \frac{134}{(134 + 9)} = 0.9370629$$

$$Specificity = \frac{378}{(378 + 82)} = 0.8217391$$

Comparing the three different MLPNN models, the third model overweighed the other two models regarding accuracy and recall which are the most significant performance metrics for classifying abnormal foetuses. As a such, the results of third MLPNN model were considered to be compared with other applied machine learning algorithms. The above results of MLPNNs are not repeatable as slightly different results will be generated each time the model is running because of randomisation element.

5 Evaluation and Results

The gained results were represented by Tableau for a comparison in order to select the best performing model. The C5.0 decision tree and RF models scored the highest accuracy compared to SVM (refer Figure 64).

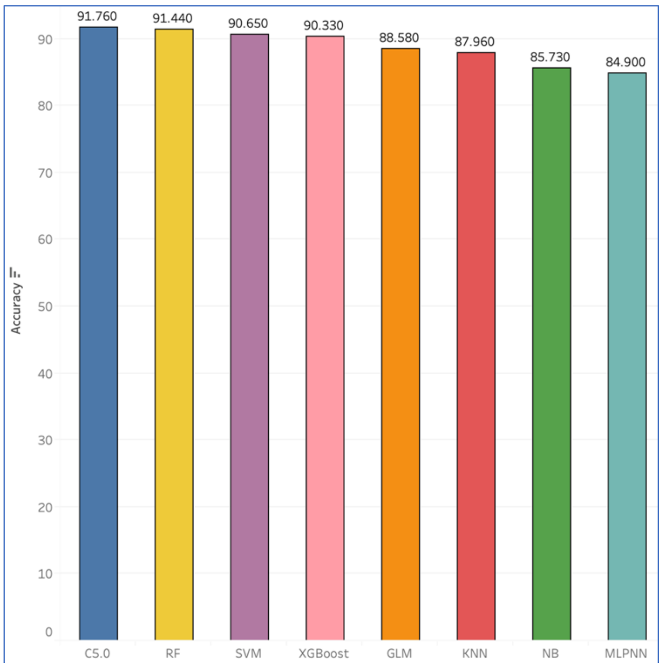


Figure 64: Accuracy as a Performance Metric for Evaluation

Despite the highest accuracy of C5.0 and RF, they performed better at classifying normal foetuses which does not serve the research question of this project (refer Figure 65).

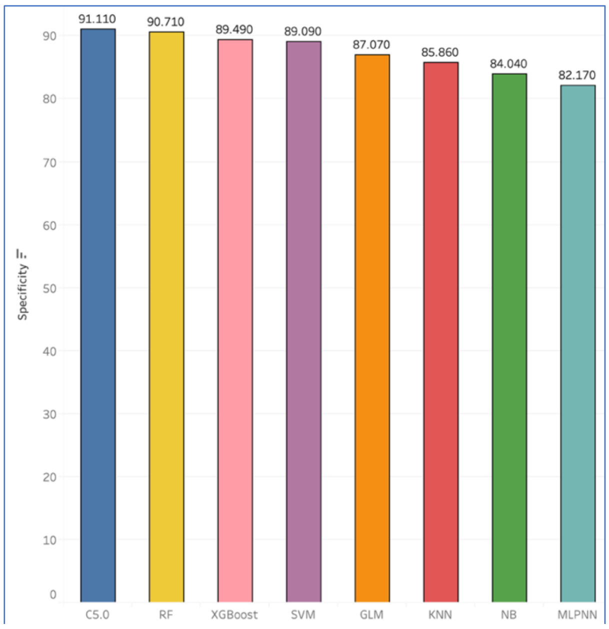


Figure 65: Specificity as a Performance Metric for Evaluation

Among machine learning and deep learning algorithms applied in this project, the SVM model performed as the best classifier to classify abnormal foetuses at a recall of 96.32% compared to a specificity of 89.09% (refer Figure 66).

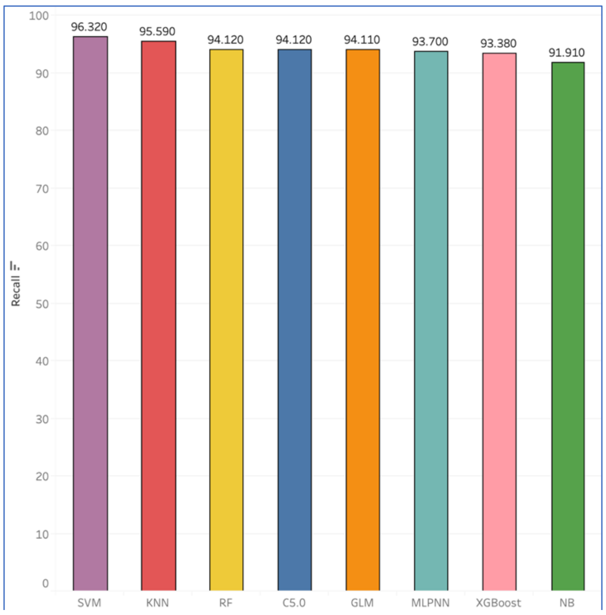


Figure 66: Recall as a Performance Metric for Evaluation

The hardware and software environments helped to fulfil this research project allowing research question and sub-question to be answered comprehensively. Additionally, these environments paved the way to the implementation of machine learning and deep learning algorithms that resulted in the development of SVM algorithm capable of classifying abnormal foetuses at a reliable scale.

6 Conclusion

The software and hardware environments explained above played a key role in the classification of abnormal foetuses. Post-hoc test analysis, performed by SPSS, could determine the power of study by proving the significant difference between the means of NSP different groups and the observed power value of each independent variable. The application of Artificial Intelligence, represented by its subsets; machine learning and deep learning algorithms, added a synergy to healthcare sector saving their time and money through the identification of abnormal foetuses. The user-friendly environments which are R and RStudio helped in data exploration and analysis. The data visualization tool, Tableau, also reverted informative visualizations for the gained results.

References

Dua, D., Graff, C., 2017. UCI Machine Learning Repository [WWW Document]. University of California, Irvine, School of Information and Computer Sciences. URL http://archive.ics.uci.edu/ml

Ocak, H., Ertunc, H.M., 2013. Prediction of fetal state from the cardiotocogram recordings using adaptive neuro-fuzzy inference systems. Neural Comput & Applic 23, 1583–1589. https://doi.org/10.1007/s00521-012-1110-3