WESTERN SYDNEY
UNIVERSITY

Computing, Engineering & Mathematics

ASSIGNMENT / REPORT COVER SHEET

This sheet must be attached to all material being submitted for marking.

Student name: **Student number:**	Quoc Bao Nguyen 20289300	**Student name:** **Student number:**	
Sections completed individually		**Sections completed individually**	
Unit name & number:	Data Science - 301044		
Tutorial day and time:	Friday 1.00 pm - 3.00 pm		
Title of Assignment:	Diabetes Prediction Methodology		
Student Submitting the Assignment:	Quoc Bao Nguyen		
Date submitted:	October 15		

Student Declaration (must be signed)

Declaration:

☒ I hold a copy of this assignment if the original is lost or damaged.

☒ I hereby certify that no part of this assignment or product has been copied from any other student's work or from any other source except where due acknowledgement is made in the assignment.

☒ No part of the assignment/product has been written / produced for me by any other person except where collaboration has been authorised by the subject lecturer/tutor concerned

☒ I am aware that this work may be reproduced and submitted to plagiarism detection software programs for the purpose of detecting possible plagiarism ***(which may retain a copy on its database for future plagiarism checking)***

☒ I hereby certify that no part of this assignment or product has been submitted by me in another (previous or current) assessment, except where appropriately referenced, and with prior permission from the Lecturer/Tutor/ Unit Co-ordinator for this unit.

☒

Student signature and date:		15.10.2020
Student signature and date		

Note: An examiner or lecturer/tutor has the right to not mark this assignment if the above declaration has not been signed.

DIABETES PREDICTION METHODOLOGY

Quoc Bao Nguyen
School of Computing, Engineering and Mathematics
Western Sydney University
Locked Bag 1797
Penrith NSW 2751
20289300@student.westernsydney.edu.au

***Abstract*—The research paper analyzes some attributes that might affect the likelihood of having diabetes. From that, the paper has constructed some models to predict the Diabetes disease using some of the collected variables. At the end of the study, Glucose and BMI are the 2 variables that are used in the best performed model. However, some cofounding variables such as age, blood pressure and Insulin should be investigated more since they have a high correlation with the 2 chosen variables as well as they also have impacts on the target variables.**

I. INTRODUCTION

For many years, it has been a paradox that many Pima Indian women suffering from diabetes disease, triggering many researchers and scientists to dedicate themselves to explore the causes. The collected dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

The purpose of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. The data set to be studied contains some medical predictor variables such as number of times pregnant, BMI, insulin level, age, glucose, … and one target variable, Outcome. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females and at least 21 years old of Pima Indian heritage.

II. DATA EXPLORATION AND DATA PROCESSING

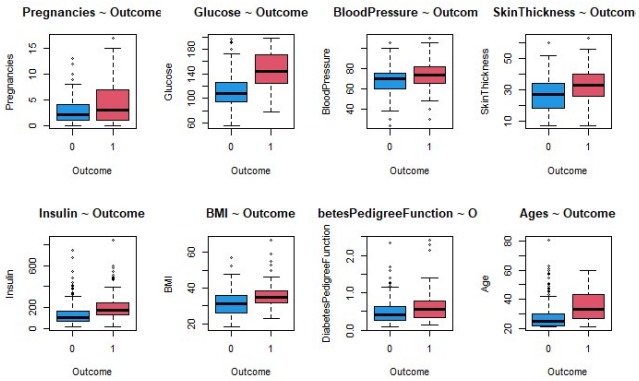
A. Data set

There were total 768 observations with 9 variables in the raw data set. The 9 variables were Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, Age, DiabetesPedigreeFunction and Outcome. The target variable Outcome has 2 integer values 0 and 1 that represent for "No diabetes" and "Diabetes" respectively. In order to ease the analyzing process, the target variable was converted from a numeric to a factor variable. Moreover, the data set still had some missing values that required to be removed from the data. The final data remains 392 observations with 9 variables. The data set was then broken down into 2 different data sets: a training data set with 274 observations (approximately 70% the total number of observations) and a testing data set with 118 observations. Looking to the variable Outcome, there was an imbalance in the observations between the two classes in this variable. 262 out of total 392 observations had the value of "0" (No diabetes), but only 130 observations had the value of "1" (Diabetes).

Variables	Explanation
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration 2 hours in an oral glucose tolerance test (mg/dL)
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skin fold thickness (mm)
Insulin	2-Hour serum insulin (mlU/L)
BMI	Body mass index (weight in kg/(height in m)^2)
DiabetesPedigreeFunction	a function that represents how likely they are to get the disease by extrapolating from their ancestor's history
Age	Age
Outcome (Target variable)	Class variable (0 or 1)

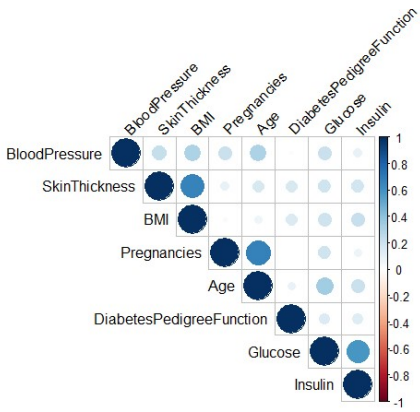
B. Data overview

Graph "2" and graph "8" below clearly show that the median line of Glucose and Age for people with "Diabete" lies outside of the boxplot of people with "No Diabete". Hence, there is likely to be a difference in the Plasma glucose concentration and the Age between the 2 groups. There is also a difference in Insulin between the 2 groups according to graph "5". Moreover, it can be seen that Age, Insulin and Glucose have much more outliners than other variables.



C. Correlation

According to the correlation matrix, there are 3 pairs of independent variables that have a high positive correlation with each other.



The first pair is BMI and SkinThickness with the correlation coefficient of around 0.66. The second one is Age and Pregnancies with the correlation coefficient of 0.68. Insulin and Glucose also have a high correlation (0.58). There are also some other variables that have medium positive correlation such as Age and BloodPressure (0.3), Age and Glucose (0.34), BloodPressure and BMI (0.3), … All the correlation indexes are positive except Prenancies and BMI (-0.03) along with DiabetesPedigreeFunction and BloodPressure (-0.02), but the correlation efficients are very close to 0.

III. Clustering

A. K-means cluster

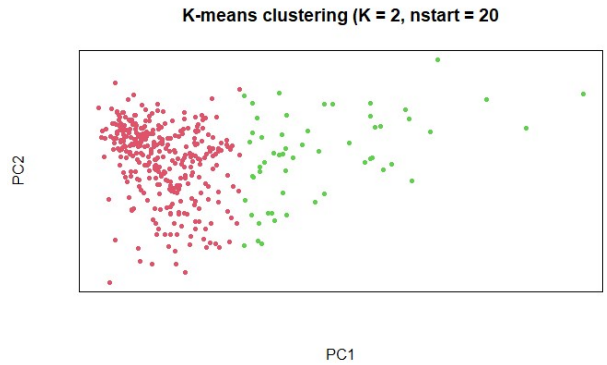
K-means clustering method was applied since this method seeks to partition the dataset into a pre-specified number of non-overlapping clusters. The K-means clustering method partitions the observations into K clusters such that the total within-cluster variation, summed all over K clusters, is minimum. The concept involved is "squared Euclidean distance", described as below:

$$d(X_i - X_{i'}) = \sqrt{\sum_{j=1}^{p}(X_{ij} - X_{i'j})^2}$$

where:

- *X is the data matrix*
- *$X_{i,j}$ is the value of the jth variable measured on the ith individual*
- *$d(X_i - X_{i'})$ is the Euclidean distance between the ith and i'^{th}*

In this case, we are interested in separate the observations into 2 groups: "Diabetes" and "No Diabetes". Hence, the chosen number of clusters is 2 (K=2). When constructing the model 2-means clustering process, the Total within-cluster sum of squares are compared to each other with different "nstart" (from 20 to 50). The minimum total within-cluster sum of square is 2,469,317, corresponding with various values of "nstart" (including nstart = 20).

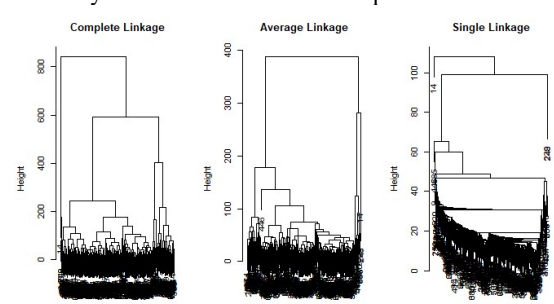


The models showed that there were 336 out of 392 observations belonging to one group and 56 remaining observations belonged to another group. The comparison between the k-means cluster and the true observations was showed below:

	K-means Cluster	
	Group 1	Group 2
"No Diabetes"	235	27
"Diabetes"	101	29

B. Hierarchical clustering

The "complete", "average" and "single" linkage clustering methods were applied, with Euclidean distance as the dissimilarity measure. The cut tree is equal to 2.



The comparison among the 3 different methods linkage cluster and the true observations is showed below:

	Complete		Average		Single	
	Group 1	Group 2	Group 1	Group 2	Group 1	Group 2
"No Diabetes"	260	2	252	10	262	0
"Diabetes"	129	1	117	13	129	1

In both the K-means clustering approach and the hierarchical clustering method, the model performance was evaluated by using the "purity" method.

To compute the "purity", each cluster was assigned to the class which is most frequent in the cluster. Then, the purity rate was measured by divide the number of correct assigned observations by the total number of observations (N). Therefore, the perfect "purity" rate should be 100% since it measured the total correct assigned observations while the bad one should be closed to the value of 0. The formula is described as follow:

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

Where
$\Omega = (w_1, w_2, \ldots w_k)$ *is clusters*
$C = (c_1, c_2, \ldots c_k)$ *is classes*

	K-means	Complete	Average	Single
Purity	67.34%	66.83%	67.6%	67.01%

According to the above result, the Average linkage hierarchical clustering has the best purity rate (67.6%).

IV. Classification Tree Modelling

A. Unpruned classification tree

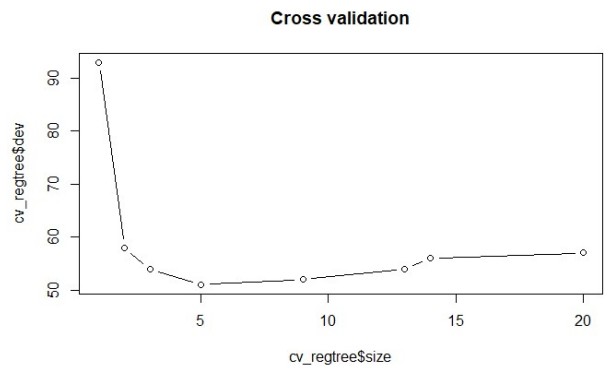
The target variable Outcome had already been converted into factor type in the data processing stage with "No diabetes" denoted by number "0" and "Diabete" denoted by number "1". The classification tree model using the training data set with all independent variables had 20 terminal nodes ("**Model 1**"). The misclassification error rate is 8.627% which is quite low. However, when fitting this tree model to the testing data, the misclassification rate dramatically increased to 23.35% which is considerable (The accuracy rate is 76.65%). The True Positive rate of the classification in the calculation is 63.41%

Confusion matrix of Model 1

Predict \ Actual	0	1
0	79	15
1	17	26

B. Pruned classifcation tree

The cross-validation plot witnessed a quick fall in the deviance when the size of the tree model increased from 0 to 5. The lowest deviance is achieved when the model size was 5. With the size more than 4, the deviance of the model increased. Hence, the best number of terminal nodes (with the lowest deviance and complexity) should be 5.



The 4 variables Glucose, Insulin, BMI and Age were used in the pruned tree model with size of 5 (**"Model 2"**). According to the model, people who have the glucose level less than 127.5 mg/dL two hours after drinking glucose solution and the 2-hour serum Insulin level lower than 143.5 mlU/L are likely to not have diabetes. In the other hand, people tend to have diabetes if their glucose level is more than 127.5 mg/dL along with their Body Mass Index greater than 29.6.

The misclassification rate of the Model 2 increased to 15.29% compared to the Model 1. This rate is 24.09% when calculated by using the testing data set (The accuracy rate = 75.91%). However, the True Positive rate is improved to 65.86%.

Confusion matrix of Model 2

Predict \ Actual	0	1
0	77	14
1	19	27

Even though the misclassification error rate of Model 2 was higher than that in Model 1, this rate when calculated with testing data shows not much differences between the two models. Moreover, not only the complexity of Model 2 (5 terminal nodes) is considerably lower than Model 1 (20 terminal nodes), but also the Precision rate of Model 2 is lower than Model 1. It can be concluded that Model 2 is better than Model 1 overall.

	Complexity	Accuracy	Precision
Model 1	20	76.65%	63.41%
Model 2	5	75.91%	65.86%

V. LOGISTIC REGRESSION MODELLING

*A. Multiple Logistic Regression Model ("**Model 3**")*

Because all variables were measured at different scales, making them contribute in-equally to the analysis, the data set is standardized in order to avoid bias when constructing models. The logistic regression was firstly fitted for Outcome against all other variables as follows:

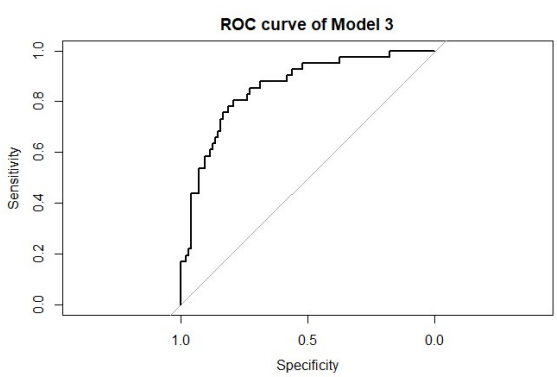
Outcome = Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age

The p-value corresponding with the coefficient of the variable Glucose is 1.20e-07 which is highly significant (***). Another significant variable is BMI with the p-value of 0.0111 (*). However, other variables are not significant due to the high p-values (greater than 5%).

Confusion matrix of Model 3

Predict \ Actual	0	1
0	86	17
1	10	24

The accuracy rate calculated by using testing data is 80.29% but the precision rate is 58.5% which is not quite good. The AIC is 249.23 The area under the ROC curve is 0.8554



*B. Multiple Logistic Regression Model ("**Model 4**")*

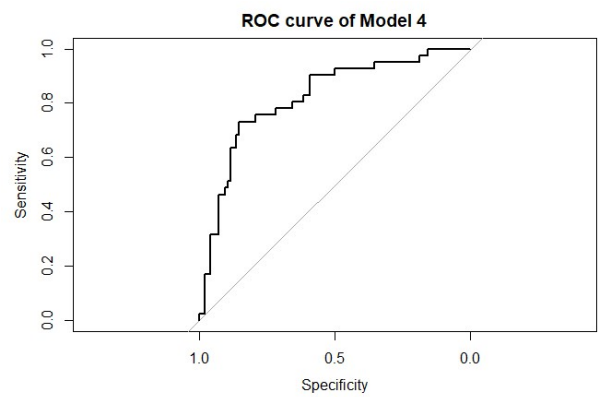
The significant variables in the above model are taken into this Model 4. The formula is described as below:

Outcome = Glucose + BMI

The p-values of the coefficient of Glucose and BMI is 2.51e-11 (***) and 0.000409 (***) respectively. The accuracy rate and precision rate is 79.56% and 58.5% that are not considerably different from the Model 3. The AIC also does not change much (249.22). The area under ROC curve even decreases to 0.8199.

Confusion matrix of Model 4

Predict \ Actual	0	1
0	85	17
1	11	24



C. Logistic regression model with polynomial terms.

The polynomial of degrees from 2 to 6 were considered with the 2 variables in Model 4: Glucose and BMI. The accuracy rate and precision rate were listed as follows:

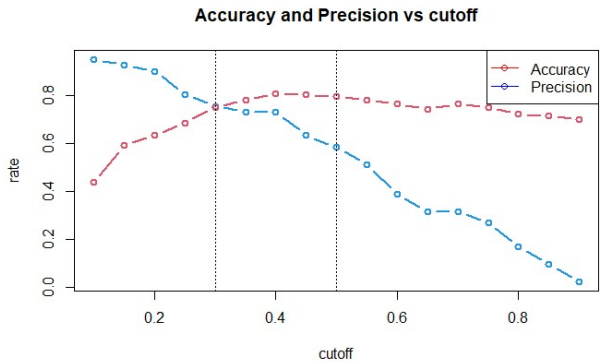
Polynomial of degree	2	3	4	5	6
Accuracy	78.8%	78.1%	77.4%	75.9%	75.9%
Precision	56.1%	53.7%	48.8%	46.3%	46.3%
AUC	0.82	0.80	0.78	0.77	0.77

According to the above summary table, the performance of the models decreased as the polynomial degree levels increased. Generally speaking, in logistics regression models, Model 4 has the least complexity level with acceptable performance.

D. Specifying the cut-off value for Model 4

Because there is a imbalance in distribution between people with "Diabetes" and people with "No Diabetes" and we

are more interested in predicting the people with "Diabetes", the precision rate plays an important role in all models. The Precision rate (True Positive rate) of Model 4 is quite low (58.5%) which is not much different from 50% since the cut-off value considered is "0.5". In this chapter, we are looking for an optimal cut-off value which not only increases the Precision rate but also keep the Accuracy rate as a significant value.



According to the above plot, the accuracy rate does not differ much when the cut-off value moves from 0.5 to 0.3 (from 79.58% to 75.18%). However, the Precision rate is considerably improved when the cut-off value decreases from 0.5 to 0.3 (from 58.5% to 75.61%). Although the optimal cut-off value is also dependent on other factors such as the cost of correct predictions and incorrect predictions, the Model 4 performs much better with the cut-off value of approximately 0.3. In other words, with the cut-off value of 0.3, Model 4 fits the data quite well with the accuracy rate of 75.18%, the precision rate of 75.61% and the AUC of 0.8199.

VI. Support Vector Machine

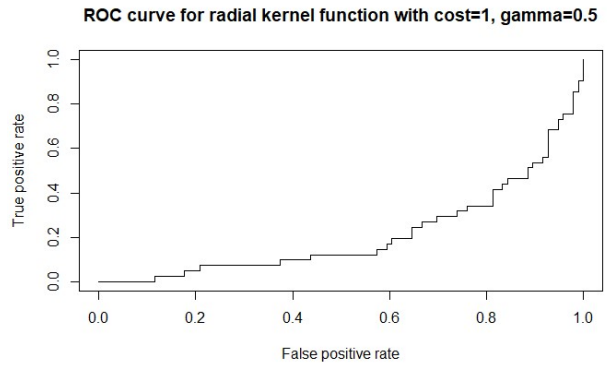
The model is fitted with the linear kernel function with the various values of cost (from 0.001 to 1000). The result is as below:

Cost	Error	Dispersion
1e-03	0.3503077	0.11405167
1e-02	0.2436923	0.06574591
1e-01	0.2436923	0.06878757
1e+00	0.2512308	0.05130794
1e+01	0.2552308	0.05521079
1e+02	0.2552308	0.05521079
1e+03	0.2552308	0.05521079

From the above result, it can be clearly seen that the optimal value of cost when using 10-fold cross validation is 0.01 since the error is minimum (0.2436923). Thus, the optimal model when using a linear kernel function was when cost is 0.01. The best model obtained consists of ***168 support vectors*** with 85 support vectors in one class and 83 support vectors in another class.

When the polynomial kernel function is used, the best parameters are ***cost = 10*** and ***degree = 5*** since the error is minimum at this value. The optimal model thus fitted when using a polynomial kernel function has 130 support vectors with 72 support vectors in one class and 58 support vectors in the other class. The error is 0.2553846 in this case

It can be seen clearly that when using radial kernel function, the best parameters are ***cost = 1 and gamma = 0.5*** since the error is minimum at this value (0.2396923). The function has 200 support vectors with 111 support vectors in one class and 89 support vectors in the other class. ("**Model 5**")



Out of all the different models fitted, error was minimum in the optimal model obtained when using radial kernel function. The accuracy rate (75.18%) and precision rate (60.98%) are calculated by using the testing data set. Both rates are quite acceptable.

VII. Comparison of Final Models

A. Summary of all constructed supervised models

Model 1: The unpruned classification tree model uses the training data set with all independent variables has 20 terminal nodes.

Model 2: The pruned classification tree includes 4 variables (Glucose, Insulin, BMI and Age) and have the size of 5 terminal nodes.

Model 3: Multiple logistic regression fitted for the target variable Outcome against all other variables.

Outcome = Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age

- *Estimate of intercept = α = -0.93535*
- *Estimate of slope of Pregnancies = β_1 = 0.30840*
- *Estimate of slope of Glucose = β_2 = 1.09336*
- *Estimate of slope of BloodPressure = β_3 = 0.02043*
- *Estimate of slope of SkinThickness = β_4 = 0.15665*
- *Estimate of slope of Insulin = β_5 = -0.03398*
- *Estimate of slope of BMI= β_6 = 0.56958*
- *Estimate of slope of DiabetesPedigreeFunction = β_7 = 0.14342*
- *Estimate of slope corresponding to Age = β_8 = 0.22996*

Hence, the model is:

*Outcome = -0.93535 + 0.30840*Pregnancies + 1.09336*Glucose + 0.02043*BloodPressure + 0.15665*SkinThickness + -0.03398*Insulin + 0.56958*BMI + 0.14342*DiabetesPedigreeFunction + 0.22996*Age*

Model 4: Only 2 independent variables Glucose and BMI (that have significant p-value) are taken into the model. The model has used the cut-off value of approximately 0.3.

Outcome = Glucose + BMI

- *Estimate of intercept = α = -0.8694*
- *Estimate of slope of Glucose = β_1 = 1.2161*
- *Estimate of slope of BMI = β_2 = 0.5992*

Hence, the model is:

*Outcome = -0.8694 +1.2161*Glucose +0.5992* BMI*

Model 5: In support vector machine methods, the radial kernel function with cost = 1 and gamma = 0.5 has the best performance. This function has 200 support vectors with 111 support vectors in one class and 89 support vectors in the other class.

B. Model Selection and Accuracy

The comparison table

	Complexity	Accuracy	Precision
Model 1	High	76.65%	63.41%
Model 2	Medium	75.91%	65.86%
Model 3	High	80.29%	58.5%
Model 4	Low	75.18%	75.61%
Model 5	High	75.18%	60.98%

According to the above comparison table, Model 3 has the highest accuracy rate compared to others. However, the precision rate is low (58.5%) which is not much further from 50% ((by chance). Moreover, due to the imbalance of the data distribution between the 2 classes of Outcome and the usefulness of predicting correctly patients with "Diabetes" in order to provide suitable treatments, the precision rate is crucial in this case. All variables are used in Model 3 but the p-values of Pregnancies, BloodPressure, SkinThickness, Insulin, DiabetesPedigreeFunction, Age are not significant. In addition, the complexity of Model 3 is high since it uses all variables, costing more in collecting data.

In the other hand, Model 4 (the multiple logistic regression) has the lowest complexity since it takes only 2 variables Glucose and BMI into account. Even though its accuracy rate is the lowest (75.18%), this accuracy rate is quite acceptable, plus the precision rate of this model is the highest (75.61%). Therefore, Model 4 should be the best model to predict Diabetes.

VIII. CONCLUSION

The study was conducted to explore and analyze some factors that might contribute to diabetes such as Age, Body Mass Index, Glucose level, Insulin level, times of Pregenancies, Blood pressure, … From the analysis, some methods were applied and fitted against the dataset. The best model was constructed, showing that the glucose level and the Body Mass Index are the 2 most efficient variables to predict the patient with Diabetes. Hence, it can be concluded that the glucose level and the weight have a correlation with the diabetes disease. Additionally, according to the above analysis especially the correlation matrix analysis and , further study should be conducted to take into account other factors such as age, blood pressure and Insulin because they might be confounding factors in the final model.

ACKNOWLEDGMENT

The author would like to thank Dr. Liwan Liyanage for her help in making this work possible. The knowledge and support from Dr.Liwan Liyanage are extremely intuitive and crucial to the success of this paper.

REFERENCES

[1] Michael Steinbach, George Karypis, and Vipin Kumar (2000). "*A comparison of document clustering techniques*". In Workshop on Text Mining, KDD, 2000.

[2] Kim H and Park H (2007). "*Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis*". _Bioinformatics (Oxford, England)_, *23*(12), pp. 1495-502. ISSN 1460-2059.

[3] Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. "*In Proceedings of the Symposium on Computer Applications and Medical Care"* (pp. 261--265). IEEE Computer Society Press.

[4] Wu, H. Xiong, J. Chen, and W. Zhou (2007). "*A generalization of proximity functions for k-means*". In ICDM.

[5] Ying CH, Peng J, Lee KL, Ingersoll GM (2002). "*An Introduction to Logistic Regression Analysis and Reporting*". J. Educ. Res. 96:1

APPENDIX

Final-Project.R

asus

2020-10-14

```
#Importing data
library("Hmisc")
```

```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(tree)
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:survival':
##
##     aml

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

1

```
library(plotROC)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following object is masked from 'package:plotROC':
##
##     ggroc

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.0-2
```

```
library(e1071)
```

```
##
## Attaching package: 'e1071'

## The following object is masked from 'package:Hmisc':
##
##     impute
```

```
library(ROCR)
library(caret)
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster
```

```
D <- read.csv("diabetes.csv")
str(D)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
```

```
##  $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

```
head(D)
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0
```

```
dim(D)
```

```
## [1] 768   9
```

```
table(D$Outcome)
```

```
##
##   0   1
## 500 268
```

```
#Data processing
i = 1
while (i <= dim(D)[1]) {
  for (j in (2:8)) {
    if ((is.na(D[i,j])) || (D[i,j] == 0)) {
      D = D[-i,]
      i = i-1
      break
    }
  }
  i = i+1
}

#Data exploring
dim(D)
```

```
## [1] 392   9
```

```
head(D)
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 7           3      78            50            32      88 31.0
## 9           2     197            70            45     543 30.5
## 14          1     189            60            23     846 30.1
## 15          5     166            72            19     175 25.8
##   DiabetesPedigreeFunction Age Outcome
## 4                    0.167  21       0
## 5                    2.288  33       1
## 7                    0.248  26       1
## 9                    0.158  53       1
## 14                   0.398  59       1
## 15                   0.587  51       1
```

```
table(D$Outcome)
```

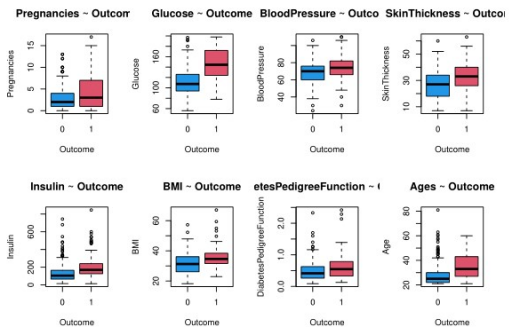
```
##
##   0   1
## 262 130
```

```
summary(D)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   : 56.0   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.:21.00
##  Median : 2.000   Median :119.0   Median : 70.00   Median :29.00
##  Mean   : 3.301   Mean   :122.6   Mean   : 70.66   Mean   :29.15
##  3rd Qu.: 5.000   3rd Qu.:143.0   3rd Qu.: 78.00   3rd Qu.:37.00
##  Max.   :17.000   Max.   :198.0   Max.   :110.00   Max.   :63.00
##     Insulin            BMI         DiabetesPedigreeFunction      Age
##  Min.   : 14.00   Min.   :18.20   Min.   :0.0850           Min.   :21.00
##  1st Qu.: 76.75   1st Qu.:28.40   1st Qu.:0.2697           1st Qu.:23.00
##  Median :125.50   Median :33.20   Median :0.4495           Median :27.00
##  Mean   :156.06   Mean   :33.09   Mean   :0.5230           Mean   :30.86
##  3rd Qu.:190.00   3rd Qu.:37.10   3rd Qu.:0.6870           3rd Qu.:36.00
##  Max.   :846.00   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.3316
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```
par(mfrow=c(2,4))
boxplot(D$Pregnancies-D$Outcome, data=D, col=c(4,2), xlab="Outcome", ylab="Pregnancies", main="Pregnanc
boxplot(D$Glucose-D$Outcome, data=D, col=c(4,2), xlab="Outcome", ylab="Glucose", main="Glucose - Outcom
boxplot(D$BloodPressure-D$Outcome, data=D, col=c(4,2), xlab="Outcome", ylab="BloodPressure", main="Bloo
```

```
boxplot(D$SkinThickness-D$Outcome, data=D, col=c(4,2), xlab="Outcome", ylab="SkinThickness", main="Skin
boxplot(D$Insulin-D$Outcome, data=D, col=c(4,2), xlab="Outcome", ylab="Insulin", main="Insulin - Outcom
boxplot(D$BMI-D$Outcome, data=D, col=c(4,2), xlab="Outcome", ylab="BMI", main="BMI - Outcome")
boxplot(D$DiabetesPedigreeFunction-D$Outcome, data=D, col=c(4,2), xlab="Outcome", ylab="DiabetesPedigre
boxplot(D$Age-D$Outcome, data=D, col=c(4,2), xlab="Outcome", ylab="Age", main="Ages - Outcome")
```

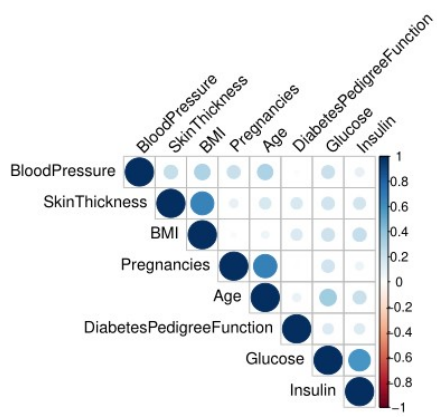



```
par(mfrow=c(1,1))
r = cor(D[1:8])
round(r,2)
```

```
##                        Pregnancies Glucose BloodPressure SkinThickness
## Pregnancies                   1.00    0.20          0.21          0.09
## Glucose                       0.20    1.00          0.21          0.20
## BloodPressure                 0.21    0.21          1.00          0.23
## SkinThickness                 0.09    0.20          0.23          1.00
## Insulin                       0.08    0.58          0.10          0.18
## BMI                          -0.03    0.21          0.30          0.66
## DiabetesPedigreeFunction      0.01    0.14         -0.02          0.16
## Age                           0.68    0.34          0.30          0.17
##                       Insulin   BMI DiabetesPedigreeFunction  Age
## Pregnancies              0.08 -0.03                     0.01 0.68
## Glucose                  0.58  0.21                     0.14 0.34
## BloodPressure            0.10  0.30                    -0.02 0.30
## SkinThickness            0.18  0.66                     0.16 0.17
## Insulin                  1.00  0.23                     0.14 0.22
```

```
## BMI                          0.23  1.00                     0.16 0.07
## DiabetesPedigreeFunction     0.14  0.16                     1.00 0.09
## Age                          0.22  0.07                     0.09 1.00
```

```
corrplot(r, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```




```
#Clustering
# K-means
X = D[,1:8]
m = c()
for (i in (20:50)) {
  set.seed(20289300)
  km = kmeans(X, centers = 2, nstart=i)
  m[i] = km$tot.withinss
}
m
```

```
##  [1]       NA       NA       NA       NA       NA       NA       NA       NA       NA       NA
## [10]       NA       NA       NA       NA       NA       NA       NA       NA       NA       NA
## [19]       NA 2469317 2469317 2469317 2469317 2469317 2469317 2469317 2469317 2469317
## [28] 2469317 2469317 2469317 2469317 2469317 2469317 2469317 2469317 2469317 2469317
## [37] 2469317 2469317 2469317 2469317 2469317 2469317 2469317 2469317 2469317 2469317
## [46] 2469317 2469317 2469317 2469317 2469317
```

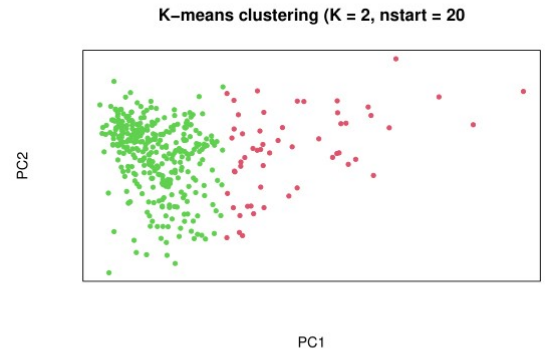
```
km = kmeans(X, centers = 2, nstart=20)
km$size
```

```
## [1]  56 336
```

```
km$betweenss
```

```
## [1] 3593670
```

```
pp = prcomp(X)
plot(pp$x[,1:2], col=fitted(km, "classes")+1,
     xaxt="n", yaxt="n", pch=20, main = "K-means clustering (K = 2, nstart = 20")
```




```
fitted(km, "classes")
```

```
##   4   5   7   9  14  15  17  19  20  21  25  26  28  29  32  33  36  40  41  44
##   2   2   2   1   1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  51  52  53  54  55  57  58  60  64  69  70  71  72  74  83  86  88  89  92  93
##   2   2   2   1   1   1   2   2   2   2   2   2   2   1   2   2   2   2   2   2
##  95  96  98  99 100 104 106 108 109 110 111 112 113 115 120 121 123 126 127 128
##   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   2
## 129 131 133 135 136 137 138 140 143 145 148 151 153 154 157 158 159 160 162 163
##   2   2   2   2   2   2   2   1   2   1   2   2   2   1   2   2   2   2   2   1
```

```
## 166 170 172 174 175 176 178 182 187 188 189 190 192 196 198 199 200 204 205 207
##   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   1   2   2   1
## 209 214 215 216 217 218 221 224 225 226 229 230 232 233 235 237 242 244 245 248
##   2   2   2   1   2   2   1   2   2   2   1   2   1   2   2   2   2   2   2   1
## 249 253 255 259 260 261 266 272 274 276 278 280 282 283 286 287 288 289 290 291
##   1   2   2   1   2   2   2   2   2   2   2   1   2   2   2   1   2   2   2   2
## 292 293 294 296 297 298 299 302 303 306 307 308 309 310 312 313 314 316 317 319
##   2   2   2   2   1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 321 324 326 327 329 330 332 335 336 339 341 342 346 347 349 354 357 359 360 361
##   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   2   1   1
## 365 366 369 370 371 373 374 375 376 377 378 380 381 383 384 385 386 389 390 391
##   1   2   2   2   1   2   2   2   1   2   2   2   2   2   2   2   1   2   2
## 393 394 396 397 403 406 410 412 413 414 415 416 420 421 422 423 425 426 428 429
##   1   2   1   2   2   2   1   2   1   2   2   1   2   2   2   2   2   1   2   2
## 430 432 433 442 443 446 447 448 449 450 451 453 455 458 459 460 461 463 466 467
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 468 470 477 478 479 481 483 484 486 487 488 491 494 498 499 500 501 504 507 508
##   2   2   2   2   2   1   2   2   2   1   1   2   2   2   2   2   2   2   2   2
## 509 512 515 516 517 520 521 522 527 528 529 531 533 535 539 540 541 542 544 545
##   2   2   2   2   2   1   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 546 547 548 549 552 554 555 556 562 563 564 566 567 568 569 570 573 574 575 576
##   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   2   1   2
## 577 585 589 592 594 595 596 598 600 604 607 608 609 610 611 612 613 615 618 621
##   2   1   2   2   2   2   2   2   2   2   1   2   1   2   2   2   1   2   2   2
## 624 626 632 634 638 639 640 641 645 646 647 648 649 651 652 653 655 656 657 658
##   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   1   2   2
## 660 663 664 666 669 670 671 673 674 680 681 683 686 689 690 693 694 696 697 699
##   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   1   2   2
## 701 705 708 710 711 712 714 716 717 719 722 723 724 727 731 733 734 737 739 741
##   2   2   1   2   1   2   1   1   2   2   2   2   2   2   2   2   2   2   2   2
## 742 743 745 746 748 749 752 754 756 761 764 766
##   2   2   2   2   2   2   2   1   2   2   2   2
```

```
table(Outcome=D$Outcome, cluster=fitted(km, "classes"))
```

```
##        cluster
## Outcome   1   2
##       0  27 235
##       1  29 101
```

```
table(D$Outcome)
```

```
##
##   0   1
## 262 130
```

```
# Hierarchical
hc.complete = hclust(dist(X), method = "complete")
hc.complete
```

```
##
## Call:
```

```
## hclust(d = dist(X), method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 392
```

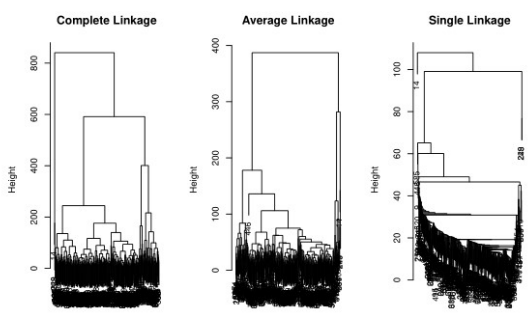
```
hc.single = hclust(dist(X), method = "single")
hc.single
```

```
##
## Call:
## hclust(d = dist(X), method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 392
```

```
hc.average = hclust(dist(X), method = "average")
hc.average
```

```
##
## Call:
## hclust(d = dist(X), method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 392
```

```
par(mfrow=c(1,3))
plot(hc.complete, main = "Complete Linkage", xlab = "", sub = "", cex = 0.9)
plot(hc.average, main = "Average Linkage", xlab = "", sub = "", cex = 0.9)
plot(hc.single, main = "Single Linkage", xlab = "", sub = "", cex = 0.9)
```




```
table(cutree(hc.complete, 2),D$Outcome)
```

```
##
##       0   1
##   1 260 129
##   2   2   1
```

```
table(cutree(hc.average, 2),D$Outcome)
```

```
##
##       0   1
##   1 252 117
##   2  10  13
```

```
table(cutree(hc.single, 2), D$Outcome)
```

```
##
##       0   1
##   1 262 129
##   2   0   1
```

```
#Purity
ClusterPurity <- function(clusters, classes) {
  sum(apply(table(classes, clusters), 2, max)) / length(clusters)
}
ClusterPurity(fitted(km, "classes"),D$Outcome)   #K-means
```

```
## [1] 0.6734694
```

```
ClusterPurity(cutree(hc.complete, 2),D$Outcome)  #Complete
```

```
## [1] 0.6683673
```

```
ClusterPurity(cutree(hc.average, 2),D$Outcome)   #Average
```

```
## [1] 0.6760204
```

```
ClusterPurity(cutree(hc.single, 2),D$Outcome)    #Single
```

```
## [1] 0.6709184
```

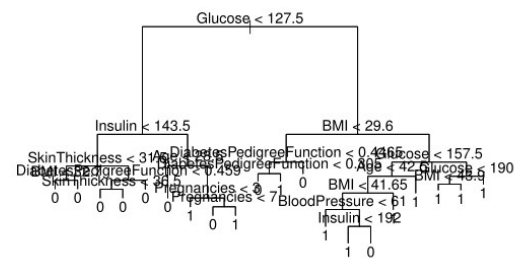
```
#Split data into train and test dataset
set.seed(20289300)
train = sample(1:nrow(D), round(nrow(D)*0.65))

#Decision Tree model
D$Outcome = as.factor(D$Outcome)
str(D)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ Pregnancies             : int  1 0 3 2 1 5 0 1 1 3 ...
##  $ Glucose                 : int  89 137 78 197 189 166 118 103 115 126 ...
##  $ BloodPressure           : int  66 40 50 70 60 72 84 30 70 88 ...
##  $ SkinThickness           : int  23 35 32 45 23 19 47 38 30 41 ...
##  $ Insulin                 : int  94 168 88 543 846 175 230 83 96 235 ...
##  $ BMI                     : num  28.1 43.1 31 30.5 30.1 25.8 45.8 43.3 34.6 39.3 ...
##  $ DiabetesPedigreeFunction: num  0.167 2.288 0.248 0.158 0.398 ...
##  $ Age                     : int  21 33 26 53 59 51 31 33 32 27 ...
##  $ Outcome                 : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 1 2 1 ...
```

```
table(D$Outcome)
```

```
##
##   0   1
## 262 130
```

```
traintree = D[train,]
testtree = D[-train,]
treemodel1 = tree(Outcome~., traintree)
par(mfrow=c(1,1))
plot(treemodel1)
text(treemodel1,pretty=0)
```




```
summary(treemodel1)
```

```
##
## Classification tree:
## tree(formula = Outcome ~ ., data = traintree)
## Number of terminal nodes:  20
## Residual mean deviance:  0.3994 = 93.85 / 235
## Misclassification error rate: 0.08627 = 22 / 255
```

```
tree_pred1 = predict(treemodel1, testtree, type="class")
mis = table(tree_pred1,testtree$Outcome)
table(tree_pred1,testtree$Outcome)
```

```
##
## tree_pred1  0  1
##          0 79 15
##          1 17 26
```

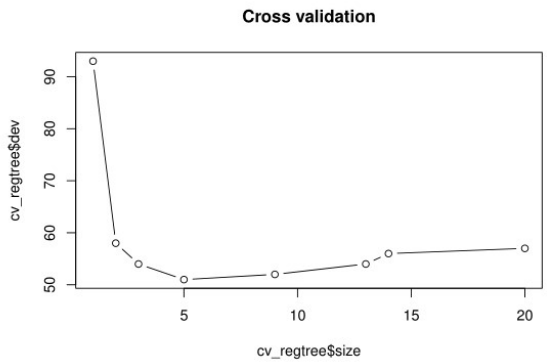
```
(mis[1,2]+mis[2,1])/sum(mis)
```

```
## [1] 0.2335766
```

```r
#cross validation
set.seed(20289300)
cv_regtree <- cv.tree(treemodel1, FUN=prune.misclass)
cv_regtree
```

```
## $size
## [1] 20 14 13  9  5  3  2  1
##
## $dev
## [1] 57 56 54 52 51 54 58 93
##
## $k
## [1] -Inf  0.0  1.0  1.5  2.5  4.5 10.0 31.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
plot(cv_regtree$size, cv_regtree$dev, type="b", main="Cross validation")
```




```r
treemodel2 = prune.misclass(treemodel1, best=5)
plot(treemodel2, main = "Classification tree model with 5 terminal nodes")
text(treemodel2,pretty=0)
```

```
## 160   4.2657020  1.3082122     0.1069722     1.1272455 -0.3538836  1.1118603
## 51   -0.7165108 -0.6360031     0.7471724    -1.7254352 -0.6231494 -1.9474798
##     DiabetesPedigreeFunction        Age Outcome
## 608               -0.11880561 -0.5749362       0
## 383                1.22711651 -0.9670632       0
## 455               -0.07249431 -0.6729680       0
## 215               -0.76137488  0.5034131       1
## 160                0.85083721  1.5817623       1
## 51                -0.09275551 -0.8690315       0
```

```r
dim(logtrain)
```

```
## [1] 255   9
```

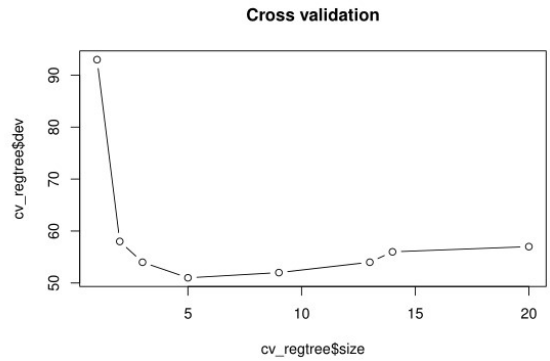
```r
#Backward selection
LogLoss <- function(pred, res){
  (-1/length(pred)) * sum (res * log(pred) + (1-res)*log(1-pred))
}
logmodel = glm(Outcome~., data=logtrain, family=binomial)
summary(logmodel)
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = binomial, data = logtrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3873  -0.6871  -0.3447   0.6497   2.4344
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -0.93535    0.17669  -5.294 1.20e-07 ***
## Pregnancies               0.30840    0.22789   1.353   0.1760
## Glucose                   1.09336    0.21597   5.063 4.14e-07 ***
## BloodPressure             0.02043    0.17318   0.118   0.9061
## SkinThickness             0.15665    0.21686   0.722   0.4701
## Insulin                  -0.03398    0.19445  -0.175   0.8613
## BMI                       0.56958    0.22423   2.540   0.0111 *
## DiabetesPedigreeFunction  0.14342    0.16744   0.857   0.3917
## Age                       0.22996    0.23555   0.976   0.3289
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 329.89  on 254  degrees of freedom
## Residual deviance: 231.23  on 246  degrees of freedom
## AIC: 249.23
##
## Number of Fisher Scoring iterations: 5
```


```r
#cross validation
set.seed(20289300)
cv_regtree <- cv.tree(treemodel1, FUN=prune.misclass)
cv_regtree
```

```
## $size
## [1] 20 14 13  9  5  3  2  1
##
## $dev
## [1] 57 56 54 52 51 54 58 93
##
## $k
## [1] -Inf  0.0  1.0  1.5  2.5  4.5 10.0 31.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
plot(cv_regtree$size, cv_regtree$dev, type="b", main="Cross validation")
```


```r
treemodel2 = prune.misclass(treemodel1, best=5)
plot(treemodel2, main = "Classification tree model with 5 terminal nodes")
text(treemodel2,pretty=0)
```


```r
#Standardize data
str(D)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ Pregnancies             : int  1 0 3 2 1 5 0 1 1 3 ...
##  $ Glucose                 : int  89 137 78 197 189 166 118 103 115 126 ...
##  $ BloodPressure           : int  66 40 50 70 60 72 84 30 70 88 ...
##  $ SkinThickness           : int  23 35 32 45 23 19 47 38 30 41 ...
##  $ Insulin                 : int  94 168 88 543 846 175 230 83 96 235 ...
##  $ BMI                     : num  28.1 43.1 31 30.5 30.1 25.8 45.8 43.3 34.6 39.3 ...
##  $ DiabetesPedigreeFunction: num  0.167 2.288 0.248 0.158 0.398 ...
##  $ Age                     : int  21 33 26 53 59 51 31 33 32 27 ...
##  $ Outcome                 : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 1 2 1 ...
```

```r
head(D)
```

```
##    Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 4            1      89            66            23      94 28.1
## 5            0     137            40            35     168 43.1
## 7            3      78            50            32      88 31.0
## 9            2     197            70            45     543 30.5
## 14           1     189            60            23     846 30.1
## 15           5     166            72            19     175 25.8
##    DiabetesPedigreeFunction Age Outcome
## 4                     0.167  21       0
## 5                     2.288  33       1
## 7                     0.248  26       1
## 9                     0.158  53       1
## 14                    0.398  59       1
## 15                    0.587  51       1
```

```r
mu=c()
s=c()
for (i in (1:8)){
  mu[i] = mean(D[,i])
  s[i] = sd(D[,i])
}
for (i in (1:dim(D)[1])){
  for (j in (1:8)) {
    D[i,j] = (D[i,j] - mu[j])/s[j]
  }
}

logtrain = D[train,]
logtest = D[-train,]
head(logtrain)
```

```
##     Pregnancies    Glucose BloodPressure SkinThickness    Insulin        BMI
## 608  -0.7165108 -0.9924425    -0.6932780    -0.3941842 -0.9681461 -1.9332503
## 383  -0.7165108 -0.4415815    -0.8533280    -2.0107033  0.2183062 -1.0937105
## 455  -0.4051225 -0.7332138    -1.3334782    -0.1089161 -0.4296146  0.6707462
## 215   1.7745956 -0.3443708     0.9072224     0.2714413  0.1594043  0.1584846
```

```
pred = predict(logmodel, logtest, type="response")
LogLoss(pred,as.numeric(logtest$Outcome)-1)
```

```
## [1] 0.4291069
```

```
confusionMatrix(table(predict(logmodel, logtest, type="response") >= 0.5, logtest$Outcome == 1))
```

```
## Confusion Matrix and Statistics
##
##
##         FALSE TRUE
##   FALSE    86   17
##   TRUE     10   24
##
##                Accuracy : 0.8029
##                  95% CI : (0.7264, 0.8659)
##     No Information Rate : 0.7007
##     P-Value [Acc > NIR] : 0.00463
##
##                   Kappa : 0.5059
##
##  Mcnemar's Test P-Value : 0.24821
##
##             Sensitivity : 0.8958
##             Specificity : 0.5854
##          Pos Pred Value : 0.8350
##          Neg Pred Value : 0.7059
##              Prevalence : 0.7007
##          Detection Rate : 0.6277
##    Detection Prevalence : 0.7518
##       Balanced Accuracy : 0.7406
##
##        'Positive' Class : FALSE
##
```

```
g <- roc(logtest$Outcome ~ pred, data = logtest)
```

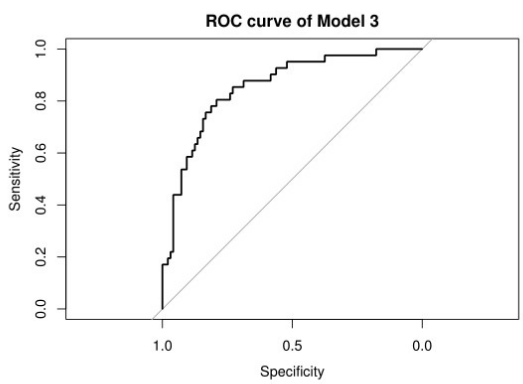
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
g$auc
```

```
## Area under the curve: 0.8554
```

```
plot(g, main = "ROC curve of Model 3")
```




```
anova(logmodel)
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Outcome
##
## Terms added sequentially (first to last)
##
##
##                          Df Deviance Resid. Df Resid. Dev
## NULL                                       254     329.89
## Pregnancies               1   16.526       253     313.36
## Glucose                   1   62.102       252     251.26
## BloodPressure             1    2.236       251     249.02
## SkinThickness             1    9.025       250     240.00
## Insulin                   1    0.014       249     239.98
## BMI                       1    6.784       248     233.20
## DiabetesPedigreeFunction  1    1.003       247     232.20
## Age                       1    0.967       246     231.23
```

```
logmodel2 = glm(Outcome~Glucose + BMI, data=logtrain, family=binomial)
summary(logmodel2)
```

```
##
## Call:
## glm(formula = Outcome ~ Glucose + BMI, family = binomial, data = logtrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2176  -0.7420  -0.3949   0.7212   2.3936
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.8694     0.1671  -5.204 1.95e-07 ***
## Glucose       1.2161     0.1822   6.673 2.51e-11 ***
## BMI           0.5992     0.1695   3.534 0.000409 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 329.89  on 254  degrees of freedom
## Residual deviance: 243.22  on 252  degrees of freedom
## AIC: 249.22
##
## Number of Fisher Scoring iterations: 5
```

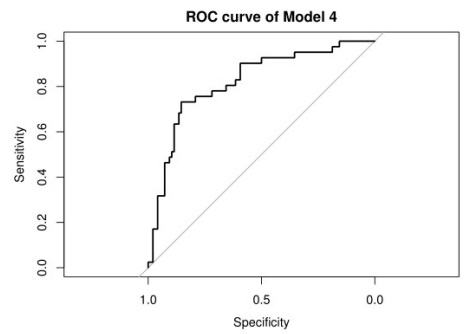
```
pred2 = predict(logmodel2, logtest, type="response")
LogLoss(pred2,as.numeric(logtest$Outcome)-1)
```

```
## [1] 0.4735107
```

```
c = confusionMatrix(table(predict(logmodel2, logtest, type="response") >= 0.5, logtest$Outcome == 1))
g <- roc(logtest$Outcome ~ pred2, data = logtest)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(g, main = "ROC curve of Model 4")
```




```
c$table
```

```
##
##         FALSE TRUE
##   FALSE    85   17
##   TRUE     11   24
```

```
#Polynomial
accuracy=c()
precision=c()
Aucc = c()
accuracy[1] = (c$table[1,1]+c$table[2,2])/sum(c$table)
precision[1] = (c$table[2,2])/(c$table[1,2]+c$table[2,2])
Aucc[1] = g$auc[1]

for (i in (2:6)) {
  polymodel = glm(Outcome~poly(Glucose,i) + poly(BMI,i), data=logtrain, family=binomial)
  c = confusionMatrix(table(predict(polymodel, logtest, type="response") >= 0.5, logtest$Outcome == 1))
  polypred = predict(polymodel, logtest, type="response")
  polyroc <- roc(logtest$Outcome ~ polypred, data = logtest)
  Aucc[i] = polyroc$auc[1]
  accuracy[i] = (c$table[1,1]+c$table[2,2])/sum(c$table)
  precision[i] = (c$table[2,2])/(c$table[1,2]+c$table[2,2])

}
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
accuracy
```

```
## [1] 0.7956204 0.7883212 0.7810219 0.7737226 0.7591241 0.7591241
```

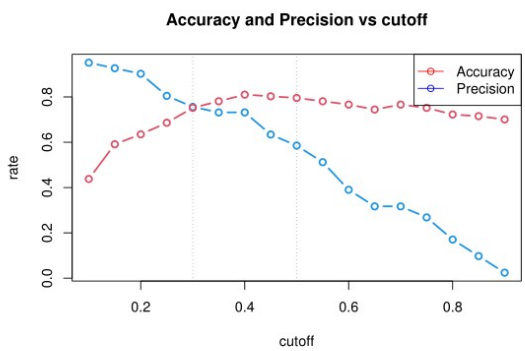
```
precision
```

```
## [1] 0.5853659 0.5609756 0.5365854 0.4878049 0.4634146 0.4634146
```

```
Aucc
```

```
## [1] 0.8198679 0.8211382 0.8043699 0.7802337 0.7782012 0.7738821
```

```
# Cut-off value
accuracy=c()
precision=c()
cutoff = seq(from=0.1, to=0.9, by=0.05)
j = 0

for (i in cutoff) {
  j = j+1
  c = confusionMatrix(table(predict(logmodel2, logtest, type="response") >= i, logtest$Outcome == 1))
  accuracy[j] = (c$table[1,1]+c$table[2,2])/sum(c$table)
  precision[j] = (c$table[2,2])/(c$table[1,2]+c$table[2,2])
}

plot(precision-cutoff, col=4, type="b", lwd=2,main = "Accuracy and Precision vs cutoff", ylab = "rate")
points(accuracy-cutoff, type="b", col=2, lwd=2)
abline(v=0.5, lty=3)
abline(v=0.3, lty=3)
legend("topright", c("Accuracy", "Precision"), col = c("red","blue"), lty = c(1, 1), pch=c(1,1))
```




```
logmodel2 = glm(Outcome-Glucose + BMI, data=logtrain, family=binomial)
summary(logmodel2)
```

```
##
## Call:
## glm(formula = Outcome - Glucose + BMI, family = binomial, data = logtrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2176  -0.7420  -0.3949   0.7212   2.3936
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.8694     0.1671  -5.204 1.95e-07 ***
## Glucose       1.2161     0.1822   6.673 2.51e-11 ***
## BMI           0.5992     0.1695   3.534 0.000409 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 329.89  on 254  degrees of freedom
## Residual deviance: 243.22  on 252  degrees of freedom
## AIC: 249.22
##
## Number of Fisher Scoring iterations: 5
```

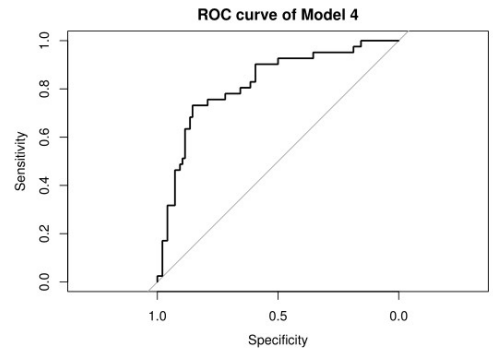
```
pred2 = predict(logmodel2, logtest, type="response")
LogLoss(pred2,as.numeric(logtest$Outcome)-1)
```

```
## [1] 0.4735107
```

```
c = confusionMatrix(table(predict(logmodel2, logtest, type="response") >= 0.3, logtest$Outcome == 1))
g <- roc(logtest$Outcome - pred2, data = logtest)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(g, main = "ROC curve of Model 4")
```




```
c$table
```

```
##
##          FALSE TRUE
##   FALSE    72   10
##   TRUE     24   31
```

```
g$auc
```

```
## Area under the curve: 0.8199
```

```
# Supporting vector machine
set.seed(20289300)
svmtrain = D[train,]
svmtest = D[-train,]
head(svmtrain)
```

```
##     Pregnancies    Glucose BloodPressure SkinThickness    Insulin        BMI
## 608  -0.7165108 -0.9924425    -0.6932780    -0.3941842 -0.9681461 -1.9332503
## 383  -0.7165108 -0.4415815    -0.8533280    -2.0107033  0.2183062 -1.0937105
## 455  -0.4051225 -0.7332138    -1.3334782    -0.1089161 -0.4296146  0.6707462
## 215   1.7745956 -0.3443708     0.9072224     0.2714413  0.1594043  0.1584846
## 160   4.2657020  1.3082122     0.1069722     1.1272455 -0.3538836  1.1118603
## 51   -0.7165108 -0.6360031     0.7471724    -1.7254352 -0.6231494 -1.9474798
##     DiabetesPedigreeFunction        Age Outcome
## 608               -0.11880561 -0.5749362       0
## 383                1.22711651 -0.9670632       0
## 455               -0.07249431 -0.6729680       0
## 215               -0.76137488  0.5034131       1
## 160                0.85083721  1.5817623       1
## 51                -0.09275551 -0.8690315       0
```

```
dim(svmtrain)
```

```
## [1] 255   9
```

```
tune_out = tune(svm, Outcome - ., data = svmtrain, kernel = "linear", ranges = list(cost = c(0.001, 0.0
summary(tune_out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.2355385
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.3493846 0.08408834
## 2 1e-02 0.2355385 0.05618346
## 3 1e-01 0.2392308 0.06209259
## 4 1e+00 0.2472308 0.06715032
## 5 1e+01 0.2472308 0.06715032
## 6 1e+02 0.2472308 0.06715032
## 7 1e+03 0.2472308 0.06715032
```

```
bestmodel1 = tune_out$best.model
summary(bestmodel1)
```

```
##
## Call:
## best.tune(method = svm, train.x = Outcome ~ ., data = svmtrain, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 10, 100, 1000)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  168
##
##  ( 85 83 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
# polynomial
set.seed(20289300)
tune_out2 = tune(svm, Outcome ~ ., data = svmtrain, kernel = "polynomial",
                 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100, 1000), d = c(2:5)))
summary(tune_out2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost d
##    10 5
##
## - best performance: 0.2553846
##
## - Detailed performance results:
##     cost d     error dispersion
## 1  1e-03 2 0.3493846 0.08408834
## 2  1e-02 2 0.3493846 0.08408834
## 3  1e-01 2 0.3572308 0.08431742
## 4  1e+00 2 0.3140000 0.05409928
## 5  1e+01 2 0.3024615 0.07734993
## 6  1e+02 2 0.2983077 0.09078936
## 7  1e+03 2 0.2946154 0.10107184
## 8  1e-03 3 0.3493846 0.08408834
## 9  1e-02 3 0.3495385 0.09464227
## 10 1e-01 3 0.3021538 0.12314998
## 11 1e+00 3 0.2632308 0.07057368
## 12 1e+01 3 0.2672308 0.07137994
## 13 1e+02 3 0.2709231 0.10347150
## 14 1e+03 3 0.3175385 0.07465089
## 15 1e-03 4 0.3533846 0.09352769
## 16 1e-02 4 0.3495385 0.09464227
## 17 1e-01 4 0.3258462 0.10224772
## 18 1e+00 4 0.2786154 0.11130793
## 19 1e+01 4 0.3023077 0.08433754
## 20 1e+02 4 0.3804615 0.09952431
## 21 1e+03 4 0.3880000 0.12979081
## 22 1e-03 5 0.3495385 0.09464227
## 23 1e-02 5 0.3340000 0.09643603
## 24 1e-01 5 0.3178462 0.11511467
## 25 1e+00 5 0.2707692 0.08487141
## 26 1e+01 5 0.2553846 0.08314021
## 27 1e+02 5 0.2675385 0.08614606
## 28 1e+03 5 0.2749231 0.09858713
```

```
best_model2 = tune_out2$best.model
summary(best_model2)
```

```
##
## Call:
## best.tune(method = svm, train.x = Outcome ~ ., data = svmtrain, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 10, 100, 1000), d = c(2:5)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  5
##      coef.0:  0
##
## Number of Support Vectors:  130
##
##  ( 72 58 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#radial
set.seed(20289300)
tune_out3 = tune(svm, Outcome ~ ., data = svmtrain, kernel = "radial",
                 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100, 1000), gamma = c(0.5, 1, 2, 3,4))
summary(tune_out3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 0.2396923
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-03   0.5 0.3493846 0.08408834
## 2  1e-02   0.5 0.3493846 0.08408834
## 3  1e-01   0.5 0.3493846 0.08408834
## 4  1e+00   0.5 0.2396923 0.08863580
## 5  1e+01   0.5 0.3060000 0.06089601
## 6  1e+02   0.5 0.3021538 0.06948368
## 7  1e+03   0.5 0.3021538 0.06948368
## 8  1e-03   1.0 0.3493846 0.08408834
## 9  1e-02   1.0 0.3493846 0.08408834
## 10 1e-01   1.0 0.3493846 0.08408834
## 11 1e+00   1.0 0.3141538 0.07818722
## 12 1e+01   1.0 0.3209231 0.05658806
## 13 1e+02   1.0 0.3209231 0.05658806
## 14 1e+03   1.0 0.3209231 0.05658806
## 15 1e-03   2.0 0.3493846 0.08408834
## 16 1e-02   2.0 0.3493846 0.08408834
## 17 1e-01   2.0 0.3493846 0.08408834
## 18 1e+00   2.0 0.3573846 0.08048195
## 19 1e+01   2.0 0.3532308 0.07393940
## 20 1e+02   2.0 0.3532308 0.07393940
## 21 1e+03   2.0 0.3532308 0.07393940
## 22 1e-03   3.0 0.3493846 0.08408834
## 23 1e-02   3.0 0.3493846 0.08408834
## 24 1e-01   3.0 0.3493846 0.08408834
## 25 1e+00   3.0 0.3533846 0.07911094
## 26 1e+01   3.0 0.3533846 0.07911094
## 27 1e+02   3.0 0.3533846 0.07911094
## 28 1e+03   3.0 0.3533846 0.07911094
## 29 1e-03   4.0 0.3493846 0.08408834
## 30 1e-02   4.0 0.3493846 0.08408834
## 31 1e-01   4.0 0.3493846 0.08408834
## 32 1e+00   4.0 0.3493846 0.08408834
## 33 1e+01   4.0 0.3533846 0.07911094
## 34 1e+02   4.0 0.3533846 0.07911094
## 35 1e+03   4.0 0.3533846 0.07911094
```

```
best_model3 = tune_out3$best.model
summary(best_model3)
```

```
##
## Call:
## best.tune(method = svm, train.x = Outcome ~ ., data = svmtrain, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 10, 100, 1000), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
```



```
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  200
##
##  ( 111 89 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
svmbest = svm(Outcome~., data=svmtrain, kernel="radial", cost=1, gamma=0.5)
svmpredict = predict(svmbest, svmtest)
table(svmpredict,svmtest$Outcome)
```

```
##
## svmpredict  0  1
##          0 78 16
##          1 18 25
```

```
rocplot = function(pred, truth, ...){
  predob = prediction(pred, truth)
  perf = performance (predob, "tpr", "fpr")
  plot(perf, ...)
}
fitted = attributes(predict(svmbest, svmtest,
                            decision.values = TRUE))$decision.values
rocplot(fitted, svmtest$Outcome, main="ROC curve for radial kernel function with cost=1, gamma=0.5")
```