5) Decision Tree

2023-04-04

Pre-processing the data-set

```
data <- read.csv("Decision_Tree_Dataset.csv", header = TRUE)
processed_data <- na.omit(data)
head(processed_data)
```

```
##   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
```

```
summary(processed_data)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
str(processed_data)
```

```
## '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 ...
```

```
nrow(processed_data)
```

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

Splitting the model

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
indexs = createDataPartition(processed_data$Outcome, times = 1, p = 0.8, list = F)
#times = no. of times to be split
#p = percentage of data to be used for training, here 80% is used of training and 20%
for testing

train = processed_data[indexs, ]
nrow(train)
```

```
## [1] 615
```

```
test = processed_data[-indexs, ]
nrow(test)
```

```
## [1] 153
```

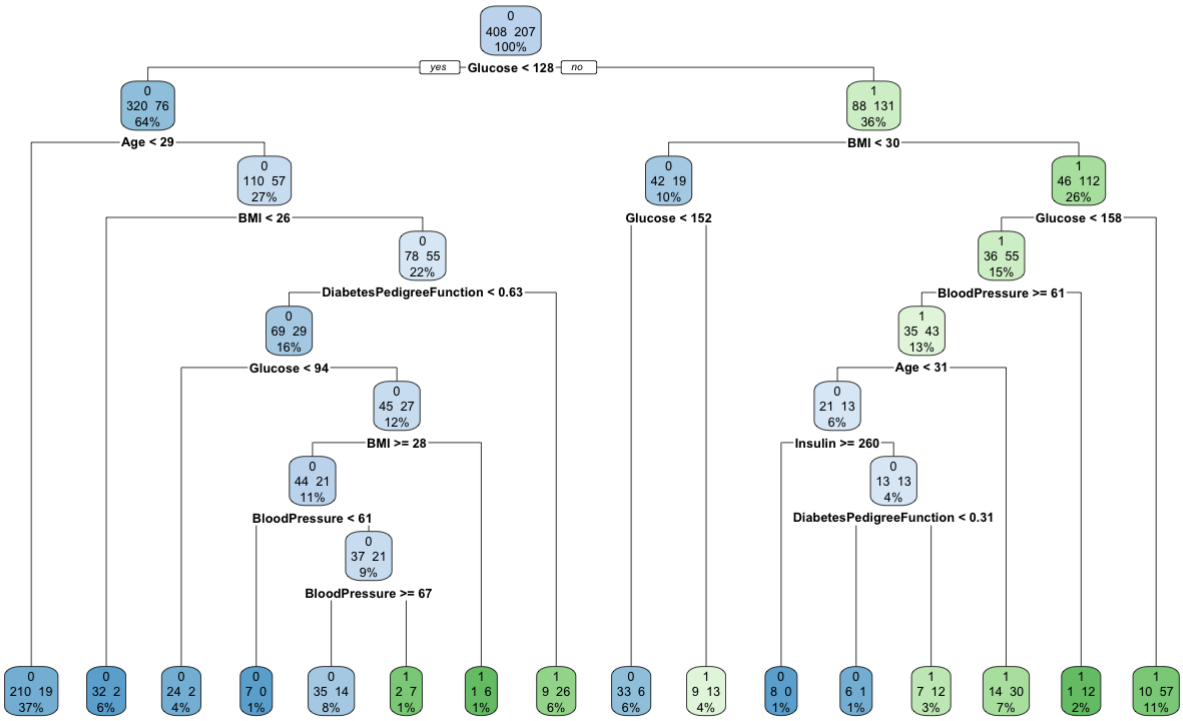
Creating the model - Information Gain and Gini Index

```
# Load the rpart package
library(rpart)

# Use Information Gain as the splitting criterion
df_tree_info_gain <- rpart(Outcome ~ ., data = train,
                           method = "class", parms = list(split = "information"))
# Use Gini Index as the splitting criterion
df_tree_gini_index <- rpart(Outcome ~ ., data = train,
                            method = "class", parms = list(split = "gini"))
```

Creating the Decision Tree of the model

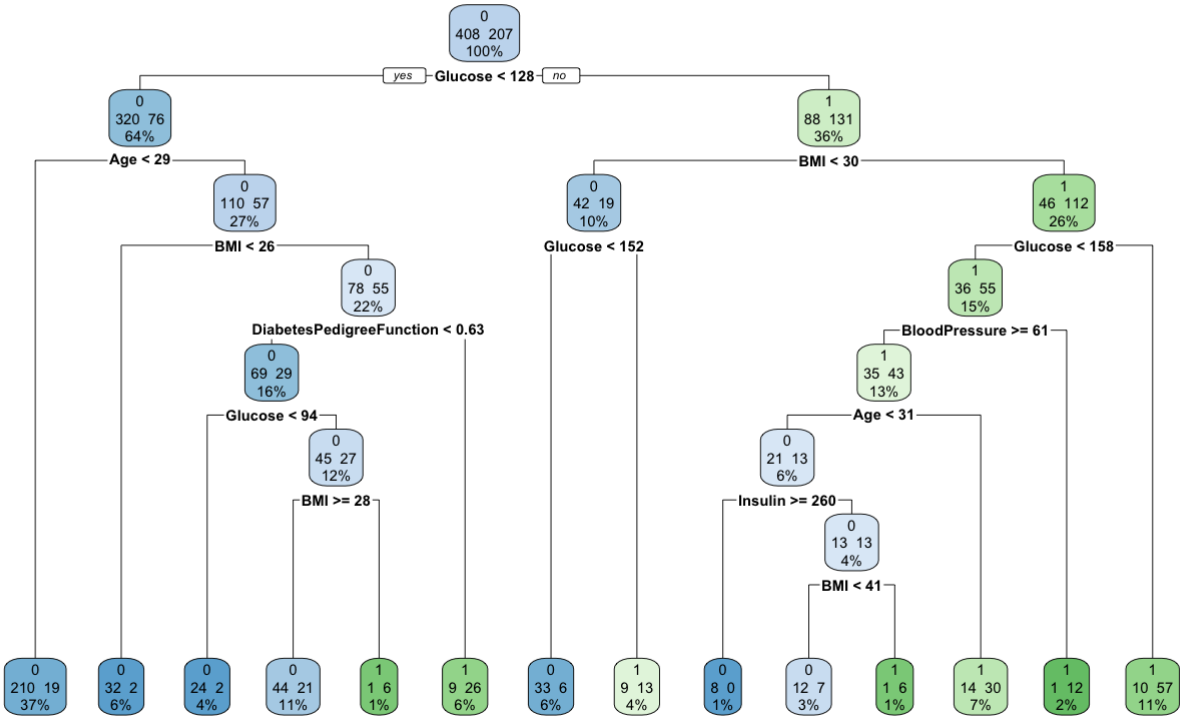
```
# Load the rpart.plot package
library(rpart.plot)

# Visualize the decision tree using Information Gain as the splitting criterion
rpart.plot(df_tree_info_gain,
           main = "Decision Tree - Information Gain", type = 2, extra = 101)
```

Decision Tree - Information Gain




```
# Visualize the decision tree using Gini Index as the splitting criterion
rpart.plot(df_tree_gini_index,
           main = "Decision Tree - Gini Index", type = 2, extra = 101)
```

Decision Tree - Gini Index



Predicting the values using the model and the confusion matrix

```
predicted = predict(df_tree_info_gain, test, type = "class")
predicted
```

```
##   5  14  16  17  22  25  28  30  35  39  42  53  57  58  73  78  83  87  92  93
##   1   1   0   0   0   1   0   0   1   0   1   0   1   1   0   0   1   0   0   0
##  95  96 105 111 112 115 122 127 129 131 133 142 143 145 148 166 168 171 176 177
##   0   1   0   1   1   1   0   0   0   1   1   0   0   0   1   1   1   0   1   0
## 180 181 186 188 195 203 212 228 232 233 236 239 245 256 261 274 275 293 295 297
##   1   0   1   1   0   1   1   1   1   0   1   1   1   0   1   0   0   1   1   0
## 319 325 330 339 357 366 369 371 381 383 393 396 408 425 426 431 437 439 441 445
##   0   0   0   1   0   0   0   1   0   0   0   0   0   1   1   0   1   0   1   1
## 446 451 455 457 463 469 471 478 480 481 493 494 496 504 507 508 517 523 527 536
##   1   0   0   0   1   0   1   0   0   1   0   1   1   1   1   0   1   0   0   1
## 538 548 551 564 568 571 572 574 576 577 578 586 600 604 615 625 627 632 636 643
##   0   0   0   1   0   0   0   0   0   0   0   0   0   1   1   0   0   0   0   0
## 652 655 657 664 665 675 676 681 683 685 689 691 693 698 701 711 715 716 719 723
##   0   0   0   1   0   0   1   0   0   0   0   0   0   0   0   1   0   1   0   0
## 728 731 733 740 744 749 752 754 758 760 762 764 767
##   1   0   1   0   1   1   0   1   0   1   1   0   0
## Levels: 0 1
```

```
confusionMatrix(factor(test$Outcome), factor(predicted))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 72 20
##          1 19 42
##
##                Accuracy : 0.7451
##                  95% CI : (0.6684, 0.812)
##     No Information Rate : 0.5948
##     P-Value [Acc > NIR] : 7.08e-05
##
##                   Kappa : 0.4698
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7912
##             Specificity : 0.6774
##          Pos Pred Value : 0.7826
##          Neg Pred Value : 0.6885
##              Prevalence : 0.5948
##          Detection Rate : 0.4706
##    Detection Prevalence : 0.6013
##       Balanced Accuracy : 0.7343
##
##        'Positive' Class : 0
##
```

Another method of creating a decision tree model

```
library(caret)
library(party)
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

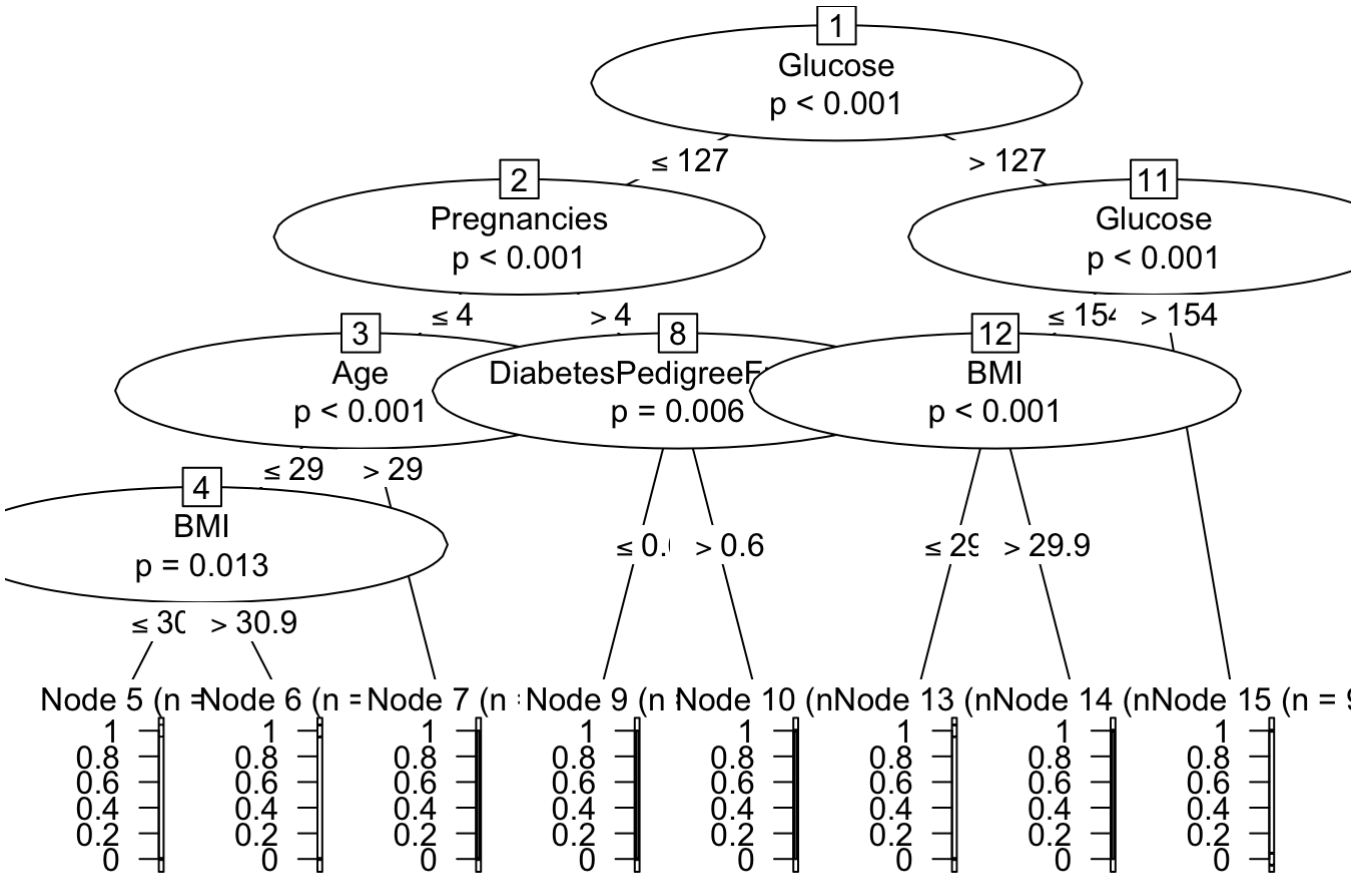
```
library(partykit)
```

```
## Loading required package: libcoin
```

```
##
## Attaching package: 'partykit'
```

```
## The following objects are masked from 'package:party':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner, node_surv,
##     node_terminal, varimp
```

```
model_using_ctree <- ctree(Outcome ~ ., data = train)
plot(model_using_ctree)
```




```
predicted_using_ctree = predict(model_using_ctree, test)
predicted_using_ctree
```

```
##           5          14          16          17          22          25
## 0.600000000 0.784946237 0.268817204 0.285714286 0.268817204 0.600000000
##          28          30          35          39          42          53
## 0.008064516 0.268817204 0.268817204 0.170000000 0.600000000 0.268817204
##          57          58          73          78          83          87
## 0.784946237 0.285714286 0.268817204 0.268817204 0.633333333 0.268817204
##          92          93          95          96         105         111
## 0.285714286 0.268817204 0.170731707 0.600000000 0.170000000 0.784946237
##         112         115         122         127         129         131
## 0.784946237 0.784946237 0.268817204 0.285714286 0.285714286 0.784946237
##         133         142         143         145         148         166
## 0.784946237 0.268817204 0.170000000 0.600000000 0.285714286 0.633333333
##         168         171         176         177         180         181
## 0.285714286 0.268817204 0.784946237 0.268817204 0.600000000 0.268817204
##         186         188         195         203         212         228
## 0.784946237 0.600000000 0.268817204 0.285714286 0.600000000 0.784946237
##         232         233         236         239         245         256
## 0.600000000 0.008064516 0.784946237 0.784946237 0.600000000 0.170000000
##         261         274         275         293         295         297
## 0.784946237 0.170000000 0.268817204 0.600000000 0.784946237 0.170731707
##         319         325         330         339         357         366
## 0.170000000 0.170000000 0.268817204 0.600000000 0.170000000 0.268817204
##         369         371         381         383         393         396
## 0.008064516 0.784946237 0.008064516 0.008064516 0.170731707 0.008064516
##         408         425         426         431         437         439
## 0.008064516 0.600000000 0.784946237 0.008064516 0.600000000 0.008064516
##         441         445         446         451         455         457
## 0.784946237 0.285714286 0.784946237 0.008064516 0.170000000 0.170731707
##         463         469         471         478         480         481
## 0.633333333 0.268817204 0.600000000 0.268817204 0.170731707 0.784946237
##         493         494         496         504         507         508
## 0.285714286 0.285714286 0.784946237 0.633333333 0.784946237 0.170731707
##         517         523         527         536         538         548
## 0.600000000 0.268817204 0.008064516 0.600000000 0.285714286 0.600000000
##         551         564         568         571         572         574
## 0.008064516 0.268817204 0.268817204 0.285714286 0.170731707 0.170000000
##         576         577         578         586         600         604
## 0.170000000 0.633333333 0.170000000 0.008064516 0.008064516 0.600000000
##         615         625         627         632         636         643
## 0.600000000 0.008064516 0.008064516 0.170000000 0.268817204 0.170731707
##         652         655         657         664         665         675
## 0.170000000 0.170000000 0.008064516 0.600000000 0.268817204 0.268817204
##         676         681         683         685         689         691
## 0.784946237 0.008064516 0.170000000 0.170731707 0.170731707 0.633333333
##         693         698         701         711         715         716
## 0.170000000 0.008064516 0.170000000 0.784946237 0.285714286 0.784946237
##         719         723         728         731         733         740
## 0.170000000 0.170731707 0.600000000 0.170731707 0.784946237 0.285714286
##         744         749         752         754         758         760
## 0.600000000 0.784946237 0.170000000 0.784946237 0.285714286 0.784946237
##         762         764         767
## 0.784946237 0.268817204 0.285714286
```

```
tb<-table(test$Outcome, predict(model_using_ctree, test))
tb
```

```
##
##     0.00806451612903226 0.17 0.170731707317073 0.268817204301075
##   0                  19   16                 8                21
##   1                   0    4                 4                 6
##
##     0.285714285714286 0.6 0.633333333333333 0.78494623655914
##   0                10   9                 5                4
##   1                 7  14                 1               25
```

Conclusion: The accuracy of the model is, 77.98% which can be regarded as an acceptable solution for the dataset. In conclusion, the Decision Tree algorithm is a powerful tool for classification and regression tasks. It is a widely used algorithm in machine learning, with applications in various fields such as finance, healthcare, and marketing. During the course of this lab report, we have implemented and evaluated the Decision Tree algorithm on a given dataset. We have seen how the algorithm works and how to tune its parameters for better performance. We have also discussed some of the limitations of Decision Trees, such as the tendency to overfit and the sensitivity to small changes in the data. Overall, Decision Trees are a useful algorithm to have in your machine learning toolbox. They are easy to interpret and can handle both categorical and numerical data. However, it is important to be aware of their limitations and to use them in combination with other algorithms or techniques, such as ensemble methods, to achieve better performance. In conclusion, the Decision Tree algorithm is a valuable tool for data analysis and prediction, and its flexibility and interpretability make it a popular choice in many real-world applications.