INFS_SP5_2023
Predictive Analytics
Decision Tree

Enna H

Contents

Objectives

- Load and explore data.
- Build a decision tree.
- Create a fully grown tree.
- Prune the decision tree.
- Make a prediction using the model.

Datasets

In this practical, we will be using a dataset that was derived from the Kaggle Stroke Prediction Dataset (https://www.kaggle.com/fedesoriano/stroke-prediction-dataset). To obtain a dataset for this practical, we applied a function that implements synthetic minority over-sampling technique (SMOTE).

Load and explore data

```r
# Load data
data <- read.csv(url("http://bit.ly/infs5100-stroke-data"))
nrow(data)
```

```
## [1] 964
```

```r
names(data)
```

```
##  [1] "id"                "gender"            "age"
##  [4] "hypertension"      "heart_disease"     "ever_married"
##  [7] "work_type"         "residence_type"    "avg_glucose_level"
## [10] "bmi"               "smoking_status"    "stroke"
```

```r
head(data)
```

```
##         id gender      age hypertension heart_disease ever_married
## 1 62793.00   Male 37.00000            0             0          Yes
## 2 21162.00 Female 78.00000            0             0          Yes
## 3 18053.38   Male 74.08813            0             0          Yes
## 4 28939.00   Male 64.00000            0             0          Yes
## 5 45277.00 Female 74.00000            0             0          Yes
## 6 28309.00 Female 67.00000            0             0          Yes
##       work_type residence_type avg_glucose_level      bmi  smoking_status
## 1       Private          Urban          79.56000 25.20000    never smoked
## 2 Self-employed          Rural          81.68000 23.00000         Unknown
## 3       Private          Urban          97.27607 27.06765    never smoked
## 4 Self-employed          Rural         111.98000       NA formerly smoked
## 5       Private          Rural         231.61000 34.60000 formerly smoked
## 6       Private          Urban          82.09000 14.10000    never smoked
##   stroke
## 1      0
## 2      0
## 3      1
## 4      1
## 5      1
## 6      0
```

```r
data <- data %>%
  mutate(
    across(c(stroke, gender, hypertension, heart_disease, ever_married,
              work_type, residence_type, smoking_status), as.factor),
    bmi = as.numeric(bmi)
  )
summary(data)
```

```
##        id            gender          age          hypertension heart_disease
##  Min.   :  121   Female:571   Min.   : 0.48   0:797        0:866
##  1st Qu.:19680   Male  :393   1st Qu.:38.00   1:167        1: 98
```

```
##  Median :37716                 Median :57.00
##  Mean   :37244                 Mean   :52.60
##  3rd Qu.:54927                 3rd Qu.:72.00
##  Max.   :72918                 Max.   :82.00
##
##  ever_married          work_type    residence_type avg_glucose_level
##  No :254      children      : 89   Rural:457      Min.   : 55.32
##  Yes:710      Govt_job      :123   Urban:507      1st Qu.: 78.28
##                Never_worked :  1                  Median : 96.22
##                Private      :559                  Mean   :116.14
##                Self-employed:192                  3rd Qu.:142.87
##                                                   Max.   :271.74
##
##       bmi                 smoking_status stroke
##  Min.   :11.30   formerly smoked:191    0:572
##  1st Qu.:24.50   never smoked   :361    1:392
##  Median :28.50   smokes         :159
##  Mean   :29.21   Unknown        :253
##  3rd Qu.:32.83
##  Max.   :60.20
##  NA's   :66
```

There are several important things you should notice here. The dataset seems to be very close to being balanced, given that we have 572 cases who did not have a stroke and 392 cases with stroke. What is also interesting is that we have 66 missing data points in the bmi column, that we will impute like we did in the previous practical.

Missing values

```r
# Impute missing values
data.imputed <- mice(data, m=3, maxit = 50, method = 'pmm', seed = 500)
```

```r
summary(data.imputed) # pmm = predictive mean matching
```

```
## Class: mids
## Number of multiple imputations:  3
## Imputation methods:
##                id            gender               age      hypertension
##                ""                ""                ""                ""
##     heart_disease      ever_married         work_type    residence_type
##                ""                ""                ""                ""
## avg_glucose_level               bmi    smoking_status            stroke
##                ""             "pmm"                ""                ""
## PredictorMatrix:
##                id gender age hypertension heart_disease ever_married work_type
## id              0      1   1            1             1            1         1
## gender          1      0   1            1             1            1         1
## age             1      1   0            1             1            1         1
## hypertension    1      1   1            0             1            1         1
## heart_disease   1      1   1            1             0            1         1
```

```
## ever_married   1      1   1            1             1            0          1
##               residence_type avg_glucose_level bmi smoking_status stroke
## id                         1                 1   1              1      1
## gender                     1                 1   1              1      1
## age                        1                 1   1              1      1
## hypertension               1                 1   1              1      1
## heart_disease              1                 1   1              1      1
## ever_married               1                 1   1              1      1
```

```
# Obtain a complete dataset
data.complete <- complete(data.imputed, 1)
head(data.complete)
```

```
##         id gender      age hypertension heart_disease ever_married
## 1 62793.00   Male 37.00000            0             0          Yes
## 2 21162.00 Female 78.00000            0             0          Yes
## 3 18053.38   Male 74.08813            0             0          Yes
## 4 28939.00   Male 64.00000            0             0          Yes
## 5 45277.00 Female 74.00000            0             0          Yes
## 6 28309.00 Female 67.00000            0             0          Yes
##       work_type residence_type avg_glucose_level      bmi  smoking_status
## 1       Private          Urban          79.56000 25.20000    never smoked
## 2 Self-employed          Rural          81.68000 23.00000         Unknown
## 3       Private          Urban          97.27607 27.06765    never smoked
## 4 Self-employed          Rural         111.98000 47.80000 formerly smoked
## 5       Private          Rural         231.61000 34.60000 formerly smoked
## 6       Private          Urban          82.09000 14.10000    never smoked
##   stroke
## 1      0
## 2      0
## 3      1
## 4      1
## 5      1
## 6      0
```

Make sure you run summary as well and compare with the summary you run on the original data. You should notice that there are no more missing values in BMI variable. However, mean and median values should be similar to the original dataset.

```
summary(data.complete)
```

```
##        id           gender          age        hypertension heart_disease
##  Min.   :  121   Female:571   Min.   : 0.48   0:797        0:866
##  1st Qu.:19680   Male  :393   1st Qu.:38.00   1:167        1: 98
##  Median :37716                Median :57.00
##  Mean   :37244                Mean   :52.60
##  3rd Qu.:54927                3rd Qu.:72.00
##  Max.   :72918                Max.   :82.00
##  ever_married         work_type   residence_type avg_glucose_level
##  No :254      children     : 89   Rural:457      Min.   : 55.32
##  Yes:710      Govt_job     :123   Urban:507      1st Qu.: 78.28
##               Never_worked :  1                  Median : 96.22
##               Private      :559                  Mean   :116.14
```

```
##               Self-employed:192                  3rd Qu.:142.87
##                                                  Max.   :271.74
##       bmi                  smoking_status stroke
##  Min.   :11.30   formerly smoked:191    0:572
##  1st Qu.:24.60   never smoked   :361    1:392
##  Median :28.55   smokes         :159
##  Mean   :29.30   Unknown        :253
##  3rd Qu.:32.83
##  Max.   :60.20
```

Data exploration

This part requires a bit of preprocessing as we want to show histograms for age, avg_glucose_level, and bmi on a single plot.

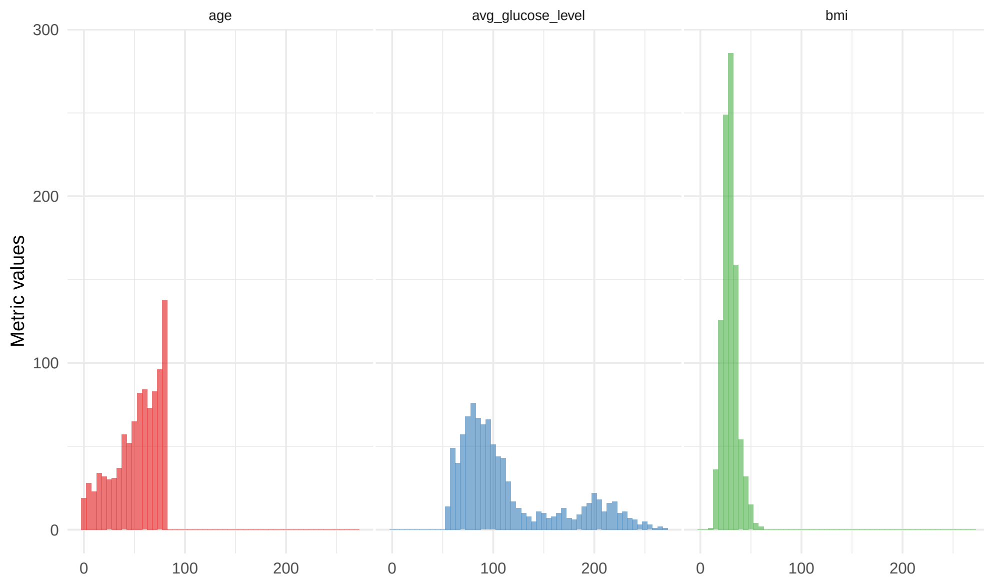
The easiest way to do that is to convert our data from wide to long format.

```
data.num.long <- gather(data.complete[, c(1, 3, 9, 10)], metric, value,
age:bmi, factor_key=TRUE)
View(data.num.long)
```

What happened here? • We selected required columns, id, age, avg_glucose_level, and bmi from the original dataset • The new dataset - data.num.long, will have two new columns, metric and value. • The metric column will have values "age", "avg_glucose_level", and "bmi". Whereas value will contain values for the given row.

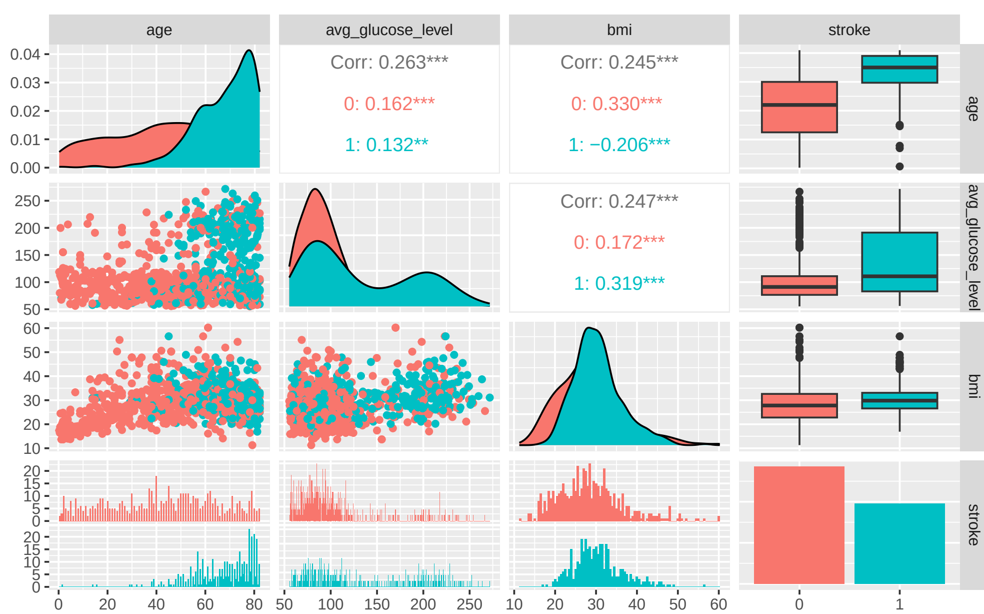
```
library(ggplot2)
library(dplyr)
library(tidyr)

# Your plotting code
num.plot <- data.num.long %>%
  ggplot(aes(x=value, fill=metric)) +
  geom_histogram(alpha=0.6, binwidth = 5) +
  scale_fill_brewer(palette="Set1") +
  theme_minimal() +
  theme(
    legend.position="none",
    panel.spacing = unit(0.1, "lines"),
    strip.text.x = element_text(size = 8)
  ) +
  xlab("") +
  ylab("Metric values") +
  facet_wrap(~metric)
num.plot
```



pacman::p_load(devtools)

```
ggpairs(data.complete, columns = c(3, 9, 10, 12),
  ggplot2::aes(colour=stroke), progress = FALSE,
  lower=list(combo=wrap("facethist", binwidth=0.5)))
```



- Age: Age of the patient
- Average Glucose Level: Average level of glucose in the blood
- BMI: Body Mass Index

Upon examining the plot, we can analyze the relationships and correlations among the numerical variables (age, average glucose level, and BMI) with respect to the target variable (stroke).

Analysis of Numerical Variables

Age

- Observing the relationship between age and stroke, we may notice a pattern indicating that the probability of having a stroke increases with age.
- This variable is likely to be a significant predictor.

Average Glucose Level

- The relationship between the average glucose level and stroke might show some correlation, but it may not be as strong as age.
- It's common in medical studies to consider glucose levels as a potential risk factor for various health conditions, including stroke.
- It might be reasonable to keep this variable in the model.

BMI (Body Mass Index)

- The correlation between BMI and stroke may be less clear from the plot.
- BMI is often considered in health studies as it can be related to overall health, but its direct connection to stroke may not be strongly evident in this data.
- Further statistical tests might be required to ascertain the significance of this variable.

Conclusion

Based on the visual analysis: - The variables "age" and "average glucose level" should likely be retained in the model, as they appear to have some correlation with the occurrence of a stroke. - The variable "BMI" might require further investigation. Depending on the context of the study, statistical tests, and considering domain knowledge, a decision could be made to either include or exclude it.

It is essential to consider the correlation analysis as a preliminary step. Further statistical analysis, model testing, and validation would provide more robust insights into the significance and relevance of these variables. Conducting feature importance analysis during model training could also aid in making the final decision on variable inclusion or exclusion.

Analysis of the Categorical Variables

```r
ggpairs(data.complete, columns = c(2, 4, 5, 6), ggplot2::aes(colour=stroke),
progress = FALSE)
```

Plot 1:

- Gender
- Hypertension
- Heart Disease
- Ever Married

Upon examining the first plot, we can analyze the relationships and associations among the categorical variables (gender, hypertension, heart disease, ever married) with respect to the target variable (stroke).

Gender

- The distribution of strokes may differ slightly between genders. Understanding if there's a significant difference might require statistical testing.

Hypertension

- The presence of hypertension (value 1) seems to be associated with a higher occurrence of strokes. This indicates that hypertension might be a significant factor in predicting strokes.

Heart Disease

- Similar to hypertension, having heart disease (value 1) could be correlated with a higher likelihood of strokes. Heart disease is a known risk factor for stroke and might be an essential variable in the model.

Ever Married

- The relationship between marital status and stroke might not be as apparent. Further exploration and domain knowledge might be required to interpret this association.

```r
ggpairs(data.complete, columns = c(7, 8, 11), ggplot2::aes(colour=stroke),
progress = FALSE)
```

Plot 2:

- Work Type
- Residence Type
- Smoking Status

Upon examining the second plot, we can analyze the relationships and associations among the categorical variables (work type, residence type, smoking status) with respect to the target variable (stroke).

Work Type

- Different work types may show varying degrees of association with stroke occurrence. For example, the "Self-employed" or "Govt_job" categories might have different implications for stroke risk, depending on lifestyle factors.
- Interpretation of this variable might require domain expertise related to occupation and health.

Residence Type

- The plot may not reveal a clear distinction between "Rural" and "Urban" residence types in terms of stroke occurrence. The relevance of this variable might need further investigation.

Smoking Status

- Smoking status can be a significant health-related variable. The categories "formerly smoked" and "smokes" might show higher occurrences of strokes compared to "never smoked."
- "Unknown" category may require special consideration, as missing data might affect the interpretation.

Conclusion

The visual analysis of categorical variables provides insights into potential associations with stroke occurrence: - Variables like hypertension, heart disease, and smoking status appear to have a meaningful association with stroke and should likely be considered in the predictive model. - Other variables like gender, work type, and residence type may require further analysis, domain knowledge, and possibly statistical testing to ascertain their relevance.

Visualizations provide valuable preliminary insights, but further modeling, statistical analysis, and validation are essential to fully understand the relationships and make informed decisions about variable inclusion or exclusion in the predictive model.

Decision Tree

we will delete id column, as we will not be using it in predictions. Additionally, we will again set seed as partitioning dataset is a random process and we want to be able to replicate our results.

```
data.class <- data.complete[, c(-1)]
set.seed(1000)
```

```
train_index <- sample(1:nrow(data.class), 0.8 * nrow(data.class))
test_index <- setdiff(1:nrow(data.class), train_index)
train <- data.class[train_index,]
test <- data.class[test_index,]
list(train = summary(train), test = summary(test))
```

```
## $train
##     gender         age        hypertension heart_disease ever_married
##  Female:454   Min.   : 0.72   0:632        0:694         No :200
##  Male  :317   1st Qu.:38.00   1:139        1: 77         Yes:571
##               Median :57.00
##               Mean   :52.69
##               3rd Qu.:71.50
##               Max.   :82.00
##          work_type   residence_type avg_glucose_level      bmi
##  children     : 69   Rural:368      Min.   : 55.32    Min.   :11.30
##  Govt_job     :102   Urban:403      1st Qu.: 78.80    1st Qu.:24.70
##  Never_worked :  1                  Median : 97.49    Median :28.55
##  Private      :440                  Mean   :117.37    Mean   :29.27
##  Self-employed:159                  3rd Qu.:147.51    3rd Qu.:32.70
##                                     Max.   :271.74    Max.   :60.20
##          smoking_status stroke
##  formerly smoked:148    0:455
##  never smoked   :281    1:316
##  smokes         :138
##  Unknown        :204
##
##
##
## $test
##     gender         age        hypertension heart_disease ever_married
##  Female:117   Min.   : 0.48   0:165        0:172         No : 54
```

```
##  Male  : 76   1st Qu.:38.00   1: 28        1: 21         Yes:139
##               Median :58.00
##               Mean   :52.25
##               3rd Qu.:73.00
##               Max.   :82.00
##          work_type   residence_type avg_glucose_level      bmi
##  children     : 20   Rural: 89      Min.   : 56.47    Min.   :13.80
##  Govt_job     : 21   Urban:104      1st Qu.: 76.34    1st Qu.:24.40
##  Never_worked :  0                  Median : 92.14    Median :28.69
##  Private      :119                  Mean   :111.21    Mean   :29.42
##  Self-employed: 33                  3rd Qu.:124.50    3rd Qu.:34.00
##                                     Max.   :252.72    Max.   :56.60
##          smoking_status stroke
##  formerly smoked:43     0:117
##  never smoked   :80     1: 76
##  smokes         :21
##  Unknown        :49
##
##
```

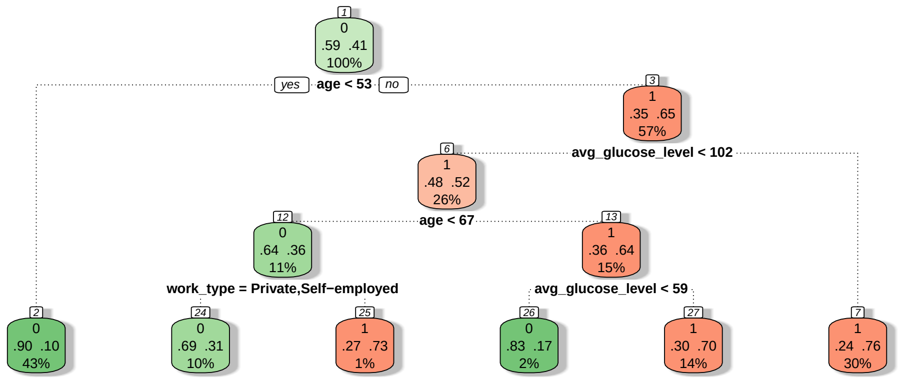
```
# Fitting a decision tree
c.tree <- rpart(stroke ~ ., train, method = "class")
```

Here is the outline of the arguments we used: • stroke ~ . tells the method to build a decision tree that will predict stroke using all the available features ("."). You can also specify a subset of features using the following notation stroke ~ age + bmi. • train is our training set. • method = "class" is used because we are predicting categorical variable (stroke). Other options would be "anova", "poisson", or "exp". If method is missing then the routine tries to make an intelligent guess. If y is a survival object, then method = "exp" is assumed, if y has 2 columns then method = "poisson" is assumed, if y is a factor then, as in our case, method = "class" is assumed, otherwise method = "anova" is assumed. It is good practice to specify the method directly, especially as more criteria may added to the function in future. • Recall that we discussed three measures for selecting the best split. Default measure for rpart is Gini. We can also set Entropy (Please refer to the documentation for this method).

```
# Printing the tree
print(c.tree)
```

```
## n= 771
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 771 316 0 (0.59014267 0.40985733)
##    2) age< 53.40585 334  32 0 (0.90419162 0.09580838) *
##    3) age>=53.40585 437 153 1 (0.35011442 0.64988558)
##      6) avg_glucose_level< 102.475 203  97 1 (0.47783251 0.52216749)
##       12) age< 67.208 86  31 0 (0.63953488 0.36046512)
##         24) work_type=Private,Self-employed 75  23 0 (0.69333333 0.30666667) *
##         25) work_type=Govt_job 11   3 1 (0.27272727 0.72727273) *
##       13) age>=67.208 117  42 1 (0.35897436 0.64102564)
##         26) avg_glucose_level< 58.99 12   2 0 (0.83333333 0.16666667) *
##         27) avg_glucose_level>=58.99 105  32 1 (0.30476190 0.69523810) *
##      7) avg_glucose_level>=102.475 234  56 1 (0.23931624 0.76068376) *
```

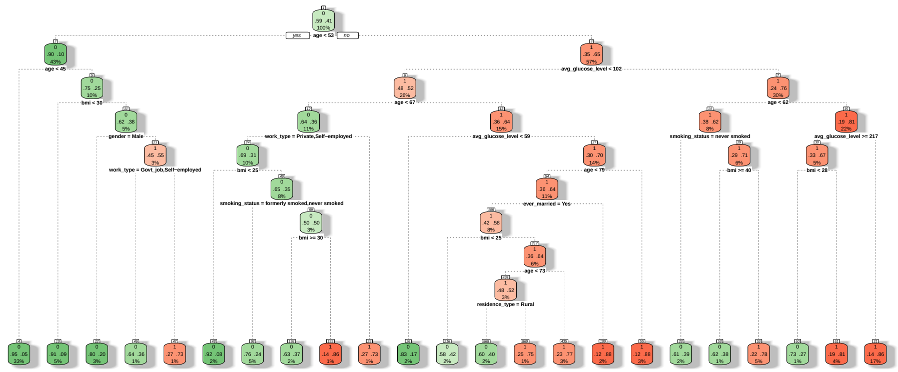
```r
# Plot the tree
fancyRpartPlot(c.tree, palettes = c("Greens", "Reds"), sub = "")
```



things to note here: • The splitting rules are created starting with the variables that has the highest association with the response variable - avg_glucose_level in this case. • Node purity - each node has two proportions written left and right. The leftmost leaf has .90 and .10, meaning that 90% of the node belongs to the predicted class - 0, i.e., no stroke. • Sample proportion - each node also has a proportion of the sample. For the leftmost node - 43% of the sample belongs to this node. • Predicted class - finally, each node also has a predicted class (0 - no stroke for the leftmost node).

As we discussed in the lecture, there are exponentially many ways to build a decision tree. The tree we have above is not a fully grown decision tree. If we look at the documentation for rpart, we will see that the default value for the complexity parameter (i.e., cp) is 0.05. This ensures that decision tree does not include any split that does not decrease the overall lack of fit by a factor of 5%.

```r
# Fit the fully grown tree
c.tree.full <- rpart(stroke ~ ., train, method = "class", cp=0)
fancyRpartPlot(c.tree.full, palettes = c("Greens", "Reds"), sub = "")
```



The fully grown tree adds two all the predictors to the model. Although there is no rule of thumb how to set the complexity parameter, there are two things we should be aware of: • A large tree is likely to overfit the data. • A small tree might miss important parameters and thus might lead to a model with high bias.

Pruning the decision tree

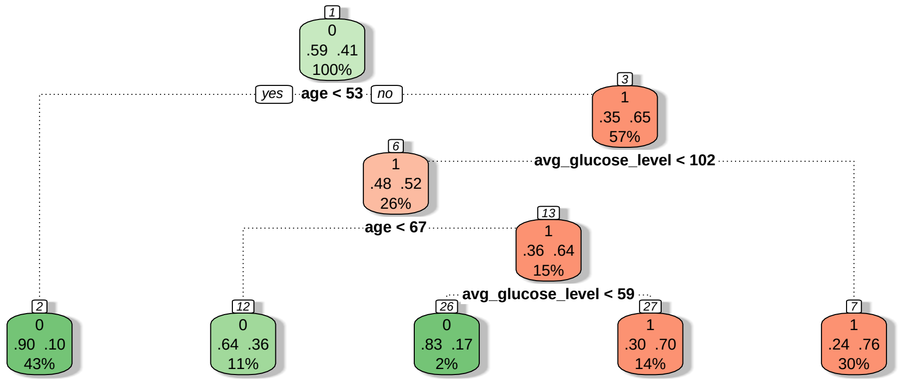
An optimal tree size can be selected adaptively from the training data. What we usually do is to build a fully-grown decision tree and then extract a nested sub-tree (prune it) in a way that gives us the tree that has the minimal node impurities.

```
# Select the best complexity parameter
min.cp <-
c.tree.full$cptable[which.min(c.tree.full$cptable[,"xerror"]),"CP"]
# print the best cp
min.cp
```

```
## [1] 0.01582278
```

prune the fully grown decision tree to find the optimal tree for the selected parameter.

```
# Prune the tree fully grown tree
p.tree.full<- prune(c.tree.full, cp=
c.tree.full$cptable[which.min(c.tree.full$cptable[,"xerror"]),"CP"])
fancyRpartPlot(p.tree.full, palettes = c("Greens", "Reds"), sub = "")
```



Make a prediction

```r
# Make a prediction
stroke.predict <- predict(p.tree.full, test, type = "class")
stroke.predicted.data <- cbind(test, stroke.predict)
head(stroke.predicted.data)
```

```
##    gender      age hypertension heart_disease ever_married     work_type
## 11 Female 44.00000            0             0          Yes       Private
## 14 Female 70.00000            0             0          Yes Self-employed
## 16 Female 53.00000            0             0          Yes Self-employed
## 17   Male 62.93439            0             0          Yes      Govt_job
## 18 Female 53.00000            0             0          Yes       Private
## 19   Male 79.84995            0             0          Yes       Private
##    residence_type avg_glucose_level      bmi  smoking_status stroke
## 11          Rural          87.71000 34.00000 formerly smoked      0
## 14          Rural          76.34000 24.40000 formerly smoked      1
## 16          Urban          81.51000 28.50000         Unknown      0
## 17          Rural         174.21767 39.14095 formerly smoked      1
## 18          Rural          94.14000 27.70000          smokes      0
## 19          Rural          65.68391 42.50917    never smoked      1
##    stroke.predict
## 11              0
## 14              1
## 16              0
## 17              1
## 18              0
## 19              1
```

Try to compare predicted labels with the ones that were already assigned in test dataset For this practical, you can calculate average agreement between assigned and predicted labels for our test set.

```
mean(stroke.predict == test$stroke)
```

```
## [1] 0.7668394
```

Feature Selection

```
# Load the necessary libraries
pacman::p_load(mlr3, mlr3learners, mlr3viz, mlr3filters, FSelectorRcpp, mlr3pipelines)

# Assuming that 'data.complete' is the name of your dataset and 'stroke' is the target variable
stroke.task <- TaskClassif$new(id = "stroke", backend = data.complete, target = "stroke")

# Create a filter using the information gain method
filter.importance = flt("information_gain")

# Calculate the feature importance
stroke.feature.importance <- filter.importance$calculate(stroke.task)

# Create a pipeline operation to filter the top 3 features
po = po("filter", filter.importance, filter.nfeat=3)

# Apply the pipeline to the task
filtered.task = po$train(list(stroke.task))[[1]]

# Subset the data with selected features
stroke.filtered = subset(data.complete, select = filtered.task$feature_names)
head(stroke.filtered)
```

```
##        age avg_glucose_level ever_married
## 1 37.00000          79.56000          Yes
## 2 78.00000          81.68000          Yes
## 3 74.08813          97.27607          Yes
## 4 64.00000         111.98000          Yes
## 5 74.00000         231.61000          Yes
## 6 67.00000          82.09000          Yes
```

```
# Adding the target variable 'stroke' back to the filtered dataset
stroke.filtered$stroke <- data.complete$stroke
```

```
train_index <- sample(1:nrow(stroke.filtered), 0.8 * nrow(stroke.filtered))
test_index <- setdiff(1:nrow(stroke.filtered), train_index)
train <- stroke.filtered[train_index,]
test <- stroke.filtered[test_index,]
list(train = summary(train), test = summary(test))
```

```
## $train
##       age         avg_glucose_level ever_married stroke
##  Min.   : 0.48   Min.   : 55.84    No :209      0:463
##  1st Qu.:38.00   1st Qu.: 77.82    Yes:562      1:308
```

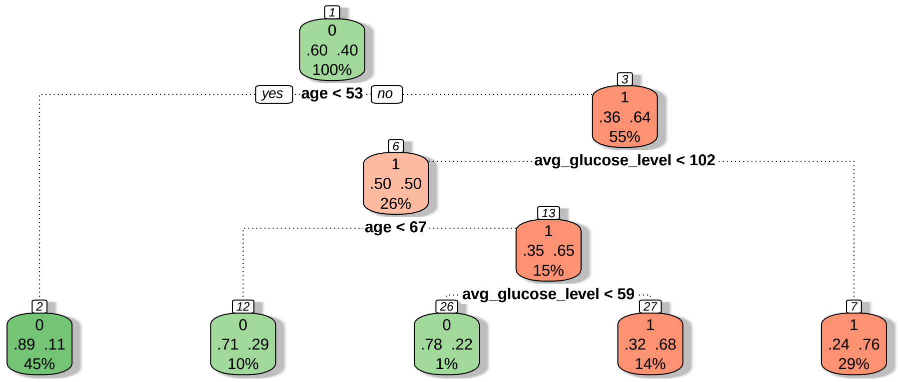
```
##  Median :56.00   Median : 96.52
##  Mean   :52.34   Mean   :116.78
##  3rd Qu.:72.00   3rd Qu.:147.51
##  Max.   :82.00   Max.   :271.74
##
## $test
##       age        avg_glucose_level ever_married stroke
##  Min.   : 0.72   Min.   : 55.32    No : 45      0:109
##  1st Qu.:40.00   1st Qu.: 81.25    Yes:148      1: 84
##  Median :59.00   Median : 96.02
##  Mean   :53.64   Mean   :113.58
##  3rd Qu.:72.00   3rd Qu.:133.19
##  Max.   :82.00   Max.   :252.72
```

```
# Fitting a decision tree
c.tree <- rpart(stroke ~ ., train, method = "class")
```

```
# Printing the tree
print(c.tree)
```

```
## n= 771
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 771 308 0 (0.6005188 0.3994812)
##    2) age< 53.40585 348  38 0 (0.8908046 0.1091954) *
##    3) age>=53.40585 423 153 1 (0.3617021 0.6382979)
##      6) avg_glucose_level< 102.475 199  99 1 (0.4974874 0.5025126)
##       12) age< 67.48474 80  23 0 (0.7125000 0.2875000) *
##       13) age>=67.48474 119  42 1 (0.3529412 0.6470588)
##         26) avg_glucose_level< 58.99 9   2 0 (0.7777778 0.2222222) *
##         27) avg_glucose_level>=58.99 110  35 1 (0.3181818 0.6818182) *
##      7) avg_glucose_level>=102.475 224  54 1 (0.2410714 0.7589286) *
```

```
# Plot the tree
fancyRpartPlot(c.tree, palettes = c("Greens", "Reds"), sub = "")
```



Make a prediction

```r
# Make a prediction
stroke.predict <- predict(c.tree, test, type = "class")
stroke.predicted.data <- cbind(test, stroke.predict)
head(stroke.predicted.data)
```

```
##          age avg_glucose_level ever_married stroke stroke.predict
## 3  74.08813          97.27607          Yes      1              1
## 6  67.00000          82.09000          Yes      0              0
## 8  57.00000          82.62000          Yes      0              0
## 14 70.00000          76.34000          Yes      1              1
## 27 37.00000          91.45000          Yes      0              0
## 28 78.00000         133.19000          Yes      1              1
```

```r
mean(stroke.predict == test$stroke)
```

```
## [1] 0.7564767
```