Machine Learning - 1100-ML0ENG (Ćwiczenia informatyczne Z-23/24)

Decision tree

```
#dataset = read.csv("sna.csv") #from working directory
dataset=read.csv("http://imul.math.uni.lodz.pl/~bartkiew/ml/data/sna.csv")
```

This sna.csv dataset contains information about users of the social network.

- user id
- gender
- age
- estimated salary
- purchased

```
str(dataset)
```

```
dataset = dataset[2:5] # or you can write dataset=dataset[-1]
#both  commands turn off the first column
```

In the last column, we put information about which users responded positively to the ads and bought the product, and which responded negatively by not buying the product. This target output is numeric, but we need here a factor to build the tree.

```
dataset$Purchased=as.factor(dataset$Purchased)
summary(dataset)
```

We divide the full set into a training set and a test set. We place 70% of the observations in the training set and 30% of all observations in the test set.

We use the **catools** package.

```
# Splitting the dataset into the Training set and Test set
install.packages("caTools")
library(caTools)
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.7)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

sample.split((Y, SplitRatio) function

Split data from vector Y into two sets in predefined ratio while preserving relative ratios of different labels in Y. Used to split the data used during classification into train and test subsets.

We check the distribution of the variable in the set -function table.

```
prop.table(table(dataset$Purchased))
prop.table(table(training_set$Purchased))
#prop.table(table(test_set$Purchased))
```

There are several types of decision tree algorithms in R. We will look at the traditional algorithm, known as rpart, and analyse the sna.csv data.

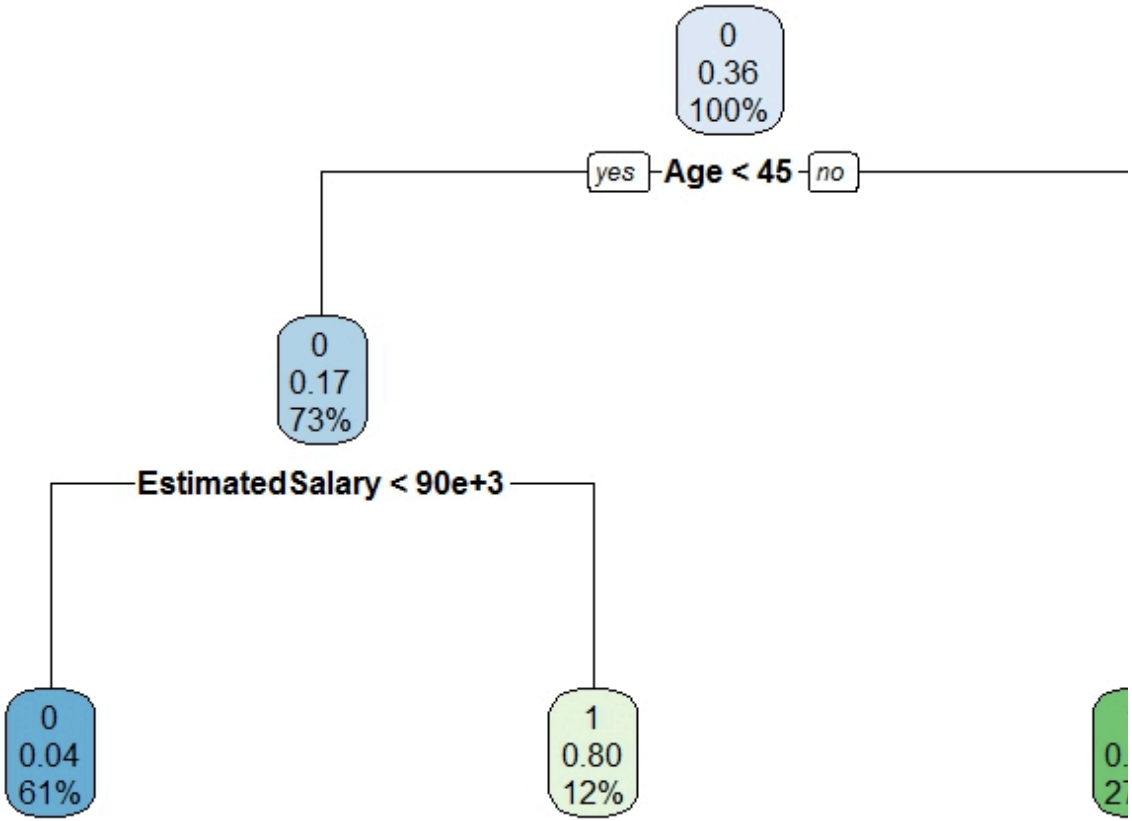
```
# install.packages("rpart")
library(rpart)
model= rpart(formula = Purchased ~ ., data = training_set)
```

The rpart function (from the rpart package) builds a tree.

In the rpart model, we can specify the target variable (Purchased) to the left of the tilde (~) and the predictor variables to the right of the tilde. We will use dot notation (.) to indicate that we will consider all variables.

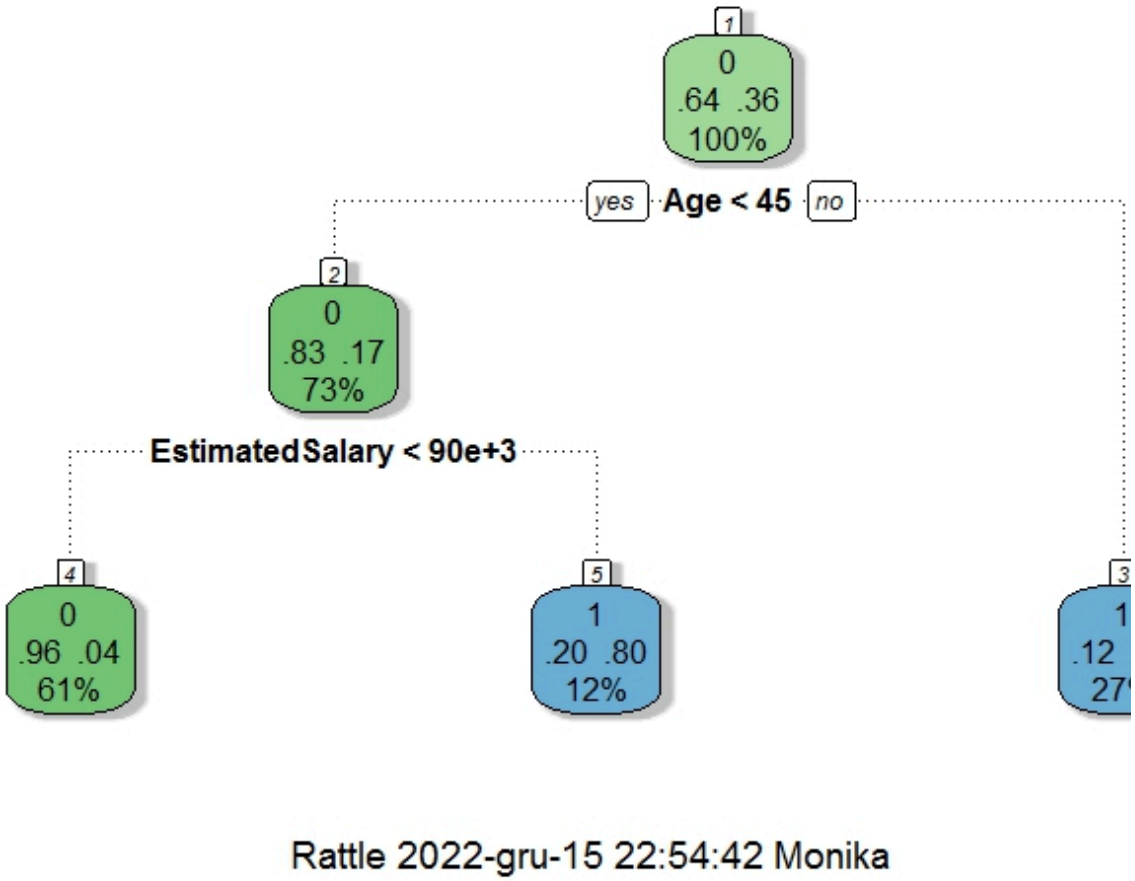
We use the **rpart.plot()** function from the package ratle to visualize the tree

```
library(rpart.plot)
rpart.plot(model)
```



We use the **fancyRpartPlot()** function from the package ratle to visualize the tree

```
# install.packages("rattle")
library(rattle)
fancyRpartPlot(model)
```



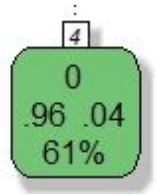
The structure of a decision tree can tell us a lot about our data. For example, the order in which features are evaluated within the tree **is significant**.

Our particular tree begins with a split by **Age** at the root node. This tells us that of the features that we used in our model, **Age** is the most predictive of our final outcome.

The farther away we get from the root node, the less predictive a feature is of the final outcome. This means that after **EstimatedSalary** is the next most predictive feature.

- If Age <45 is no, then we go to the leaf node and we can say that the customer is buying (Purchased=1)
  If Age <45 is yes, we ask if Estimated Salary <90 000 (Estomated Salary - second split).
  - If yes, then the customer does not buy (Purchased=0),
  - if not, the customer buys (Purchased=1)

Besides the order in which features are encountered, the colors and node labels are also useful in understanding our data. Recall that the root node of a tree represents the original dataset before the first split and that each of the subsequent nodes (decision and leaf nodes) represents subpartitions of the original dataset after each previous split. Looking at the labels in each node, we learn something about each of the partitions that they represent.



1. this is leaf node with number 4 - just like in the text version
2. indicates Purchased=0
3. as a result of spliting, 61% of all the observations are in this leaf node
4. among these observations
   - 96% indicates Purchased=0
   - 4% indicates class Purchased=1

When using a decision tree for classification, the nodes and branches of the tree illustrate the logical decision pathway that one can take in classifying previously unclassified data. As new data is encountered, it is evaluated against specific split criteria at each of

the decision nodes, and a pathway is chosen until a terminal node is encountered and a label is assigned. Pathways (or branches) toward the left represent agreement with the split criteria, while pathways toward the right represent disagreement with the split criteria.

```
asRules(model)
```

```
> asRules(model)
```

```
Rule number: 3 [Purchased=1 cover=75 (27%) prob=0.88]
 Age>=44.5
```

```
Rule number: 5 [Purchased=1 cover=35 (12%) prob=0.80]
 Age< 44.5
 EstimatedSalary>=9e+04
```

```
Rule number: 4 [Purchased=0 cover=170 (61%) prob=0.04]
 Age< 44.5
 EstimatedSalary< 9e+04
```

Prediction

```
y_pred = predict(model, newdata = test_set, type = 'class')
```

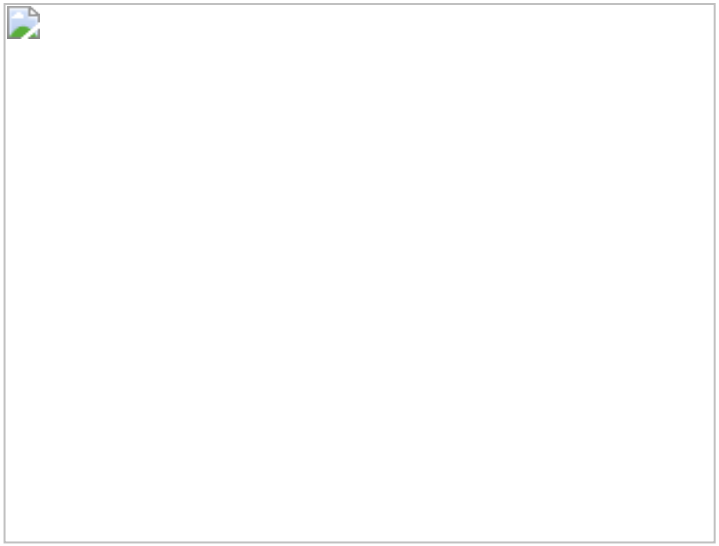
To evaluate the model, we use, for example, a confusion matrix

```
table(test_set$Purchased, y_pred)
```

```
    y_pred
      0  1
  0   67 10
  1    2 41
```

Confusion Matrix

Confusion Matrix is a performance measurement for machine learning classification problem where output can be two or more classes.



- True Positive - you predicted positive and it's true.
- True Negative - you predicted negative and it's true.
- False Positive - you predicted positive and it's false.
- False Negative - you predicted negative and it's false.

Classification Rate/Accuracy

Accuracy is also used as a statistical measure of how well a binary classification test correctly identifies or excludes a condition. That is, the accuracy is the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined. [wiki]

$$ACC = \frac{TP + TN}{TP + FP + FN + TN}$$

Error of our clasifier $error = 1 - ACC = \frac{FN + FP}{TP + FP + FN + TN}$

```
acc<-sum(test_set$Purchased==y_pred)/length(y_pred)
acc
```

```
acc<-function(y1,y2){
 sum(y1==y2)/length(y1)
}
acc(test_set$Purchased, y_pred)
```

Last modified: środa, 13 grudnia 2023, 8:40

Accessibility settings

Przetwarzanie danych osobowych

Platformą administruje Komisja ds. Doskonalenia Dydaktyki wraz z Centrum Informatyki Uniwersytetu Łódzkiego Więcej

Informacje na temat logowania

Na platformie jest wykorzystywana metoda logowania za pośrednictwem Centralnego Systemu Logowania.

Studentów i pracowników Uniwersytetu Łódzkiego obowiązuje nazwa użytkownika i hasło wykorzystywane podczas logowania się do systemu USOSweb.

Deklaracja dostępności