A Machine Learning approach to classification of companies

Fabio Morea

2022-02-18

Contents

1. Introduction

This notebook is an exercise for the course "introduction to machine learning" and, at the same time, a case study for Area Science Park in the frame of Innovation Intelligence FVG project.

Background information

Research, innovation and highly skilled people are considered to be important factors in economic and social development. Economic support policies often include funds to support research (for example with the creation of public research infrastructures), companies (for example with tenders to co-finance innovative projects) and the training of people with the necessary skills.

Area Science Park is a national research institution that manages a science and technology park located in Trieste (Italy) and is engaged in several projects aiming to support innovation at the regional level.

Innovation Intelligence FVG is a project, managed by Area Science Park and supported by Regione Friuli Venezia Giulia, which aims to monitor the performance of companies in terms of economic, employment and innovation results. For more info refer to: (www.innovationintelligence.it).

The core result of Innovation Intelligence FVG is a dataset containing information from several sources such as the chamber of commerce, the Regional Labor Market Observatory, a rating agency, as well as regional databases on innovation projects. Most of the data is open sourced; company information is available trough the chambers of commerce at a cost of 0,30€ per company, while "financial ratings" are the most expansive part, at an average cost of over 1,00€ per company.

The analysis of company performance is often targeted to some specific groups of companies, such as the *tenants of Area Science Park* (about 60 companies that have their premises or research laboratories in the science and technology park), or a *regional cluster of metal and plastic manufacturing* (identified by an association providing a list of its members, or by company size and sector of activity).

Business case and objective

The Innovation Intelligence team at Area Science Park wants to explore the potential use of Machine Learning (ML) techniques to improve the quality of analysis, reduce the impact of labour-intensive tasks and the cost of some data sources.

Specifically, the team requested a *feasibility study* focused on the following business case: define a ML model to identifying "Top performers" among companies working in the sector of metal and plastic manufacturing in Friuli Venezia Giulia.

The classification of "Top performers" is currently based on financial ratings (a numeric variable in range 1 to 10) and assegned as follows:

- Top: rating 9 to 10
- Mid: rating 5 to 8
- Low: rating 1 to 4

The performance class is assigned to each company, and used for further activities (e.g. Top performers are featured in the newsletter, Mid and Low performers are the target of a questionnaire. . .).

The issues with this methodology are the cost of financial ratings and the lack of explainability (financial ratings are based on a proprietary algorithm).

> The **objective** of the project is to assess the feasibility of a machine learnig solutions capable of predicting whether a company belongs to the "top" gropup, wihtout using information on financial rating.

The basic idea to tackle this problem is to use financial ratings as labels for a *supervided ML model*, where the independent variables are retreived from open data sets available in the Innovation Intelligence (sector, age, balance sheet data,...).

The **target audience** of the feasibility study is the Innovation Intelligence team at Area Science Park, a small group of economic analysts, that have a robust domain knowledge but limited experience in data science.

The **expected result** is a report (PDF file) and the supporting R-project (a folder including .Rmd and .Rproj fils) presenting a detailed case study, including a short introduction to model selection, feature engineering and experimental assessment of model performance.

Constraints: training and classification are generally performed on a laptop, twice a year. No specific constraints on time or computation effort (even if it takes hours, it's ok). The number of companies involved in each target group ranges generally between 200 and 2000.

Data management plan

The original data available from Innovation Intelligence and fulfills the following requirements:

- encoded in UTF-8, cleaned from non-printable characters
- table columns are attributes (features, independent variables), renamed to be human- and machine-readable
- table rows are observations If you have multiple tables, they should include a column in the table that allows them to be linked
- splitted into several tables, created unique identifiers to connect the tables
- saved each table to separate .csv file with a hunam-readable name.

No attributes were removed or summarized during pre-processing. Pre-processing is described in a separate notebook, providing details on all the attributes available in the raw data, and the transformations used to produce a smaller, cleaner data set ready for further analysis. Tidy data is saved in local folder *data/tidy*.

```
pathTidyData = './../../_data/tidy/'
```

Original and tidy data are updated on a monthly basis; the current version in based on June 2021 version and does not provide automatic updating.

The notebook has been written using *R-Studio*; data manipulation is based on *tidyverse* [www.tidyverse.org/], a data science library that includes *magrittr* (pipe operator %>%), *dplyr* (select, summarize...), *tibble* (a tidier version of the data.frame) and *ggplot2* (visualizations).

```
require(tidyverse)
require(ggthemes)
require(patchwork)
library(corrplot)

theme_set( theme_hc(base_size = 12))
pathTidyData = './../../_data/tidy/'
```

2. Methodology

Formal statement of the problem

Let a company be represented by a vector X in a multidimensional space, and associated with a binary label y representing whether the company belongs to a group of "top performers" or not.

The objective is to show that a binary classification tree can predict y with an accuracy of at least 90%, under the following conditions:

- dataset composed of at least 100 observations, homogeneous by sector and type
- computation time $<$ 1 hour on a state-of-the art personal computer.

Indexes for measuring the solutions

The solution will be assessed by a single index: *accuracy* i.e. the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined.

The **target value** is set to

$$accuracy > 0.90$$

.

The threshold may seem relatively high compared if compared to safety-critical applications, but is sufficient for the business case and in line with similar cases of binary classification of company performance found in scientific literature.

The proposed solution and the indicator are a tradeoff between complexity and explainability: for the purpose of this feasibility study, explainability and interpretability are to be preferred on complexity, but more complex methods may be used as outlined in the final section of this notebook.

Proposed solution

We consider a supervised binary classification approach, in which a company has to be classified as Top Performer using a *binary decision tree*.

In the context of a feasibility study, the binary decision tree has two key advantages over other solutions: models are *interpretable* (we can easily follow the path from input data to predictions, printing the tree structure in a textual form or plotting a graphical representation of the tree) and *explainable* (each step of the algorithm can b einspected and assess its importance to the model performance).

The model will be generated in R using a recursive partitioning algorithm available in the `rpart` library using tho followig functions:

- `rpart(formula, data, method, control)` for learning the model
- `predict(model, data, ...)` for predicting

For the purpose of this research, we will use only two of the hyperparameter available in the library, namely `minsplit` (the minimum number of observations that must exist in a node in order for a split to be attempted) and `cp` (a complexity parameter that limits the creation of *small* branches: any split that does not decrease the overall lack of fit by a factor of cp is not attempted). The standard set of parameters used in this simulation is: `parms = rpart.control(split = "gini", cp = .01 )`

The graphical representation of decision trees will be generated using `rpart.plot` library.

```
require(rpart)
require(rpart.plot)
```

The proposed solution consists of 3 phases:

1) **pre-processing**: select and rescale features to be processed (matrix X) and generate the labels to be predicted (3 different labels will be generated, based on financial rating of the companies: vectors $y7$, $y8$ and $y9$).
2) **learning phase**: generate a first instance of the decision tree, visualize the structure and evaluate its performance
3) **optimization**: tune the tree parameters to achieve optimal performance on test data.

Pre-processing: data exploration and feature engineering

The data available from Innovation Intelligence needs to be pre-processed in order to obtain a *tidy* dataset suitable for ML. An extended explanation of all the features available in the original dataset is available in the resources.

Relevant datasets

The first task is feature selection, based on domanin knowledge. The features that may have a predictive power on financial rating are "financial indicators" from the official balance sheet of each company, as well as some categorical attributes (is a startup, an "innovative SME", a "young" of "women-led companies")

The relevant datasets are

- *cmp.csv* and *codes.csv*: company information from the Italian Business Registry. Each observation is a company, there are p = 41 attributes. The study will be focused on a subset filter companies that belong to a specific sector () and of a specific type.
- *bsd.csv*. Each observation is a summary of balance sheet data (bsd) of a company (identified by *cf*) for a given year. Column labels need some improvement to remove whitespaces and possibly short english names.
- *rating.csv*. The financial rating of each company.
- *employees*. Stock and flows of employees. *empl-flow.csv* and *empl-stock.csv*.

Data is loaded in separate data structures (tibbles)

```
companies  <- read_csv( paste0(pathTidyData,"cmp.csv"),       show_col_types = FALSE )
bsd        <- read_csv( paste0(pathTidyData,"bsd.csv"),       show_col_types = FALSE )
rating     <- read_csv( paste0(pathTidyData,"rating.csv"),    show_col_types = FALSE )
codes      <- read_csv( paste0(pathTidyData,"nace.csv"),      show_col_types = FALSE )
empl.flows <- read_csv( paste0(pathTidyData,"empl_flows.csv"),show_col_types = FALSE )
empl.stock <- read_csv( paste0(pathTidyData,"empl_stock.csv"),show_col_types = FALSE )
```

Sample selection

The sample is selected according to **NACE codes** and **company type**.

The first selection on company type : we select all types that have a duty of disclosure of financial information, and therefore are suitable for the analysis, namely SU (società a responsabilità limitata con unico socio), SR (società a responsabilità limitata), SP (società per azioni), SD (società europea), RS (società a responsabilità limitata semplificata), RR (società a responsabilità limitata a capitale ridotto), AU (società per azioni con socio unico), AA (società in accomandita per azioni.

```
selectedNg = c("SU", "SR", "SP", "SD", "RS", "RR", "AU", "AA")
companies <- companies %>% filter(ng2 %in% selectedNg)
```

A further selection is based on **NACE codes**. The acronym NACE stands for *Nomenclature of Economic Activities*, a standard classification for classifying business activities managed by EUROSTAT and recognized by national statistic offices at European level. NACE codes provide a framework for the collection and presentation of a wide range of statistics in economic fields such as production, employment, national accounts, and others. The statistics produced on the basis of NACE codes are comparable at the European level and more generally at the global level. The use of NACE codes is compulsory within the European statistics system. (see _data/ino/ for a complete list of codes https://ec.europa.eu/eurostat/web/products-manuals-and-guidelines/-/ks-ra-07-015

The NACE code is subdivided into a hierarchical structure with four levels:

Level 1: 21 sections identified by alphabetical letters A to U; Level 2: 88 divisions identified by two-digit numerical codes (01 to 99); Level 3: 272 groups identified by three-digit numerical codes (01.1 to 99.0); Level 4: 615 classes identified by four-digit numerical codes (01.11 to 99.00).

Each company can be associated with one or more NACE codes, that may be different for each local unit, and are identified as main (I), “primary” (P), or “Ancillary” (S).

The selected sample is composed of companies that have at least one NACE code in one of the following Divisions: 22 (Manufacture of rubber and plastic products), 23 (Manufacture of other non-metallic mineral products), 24 (Manufacture of basic metals), 25 (Manufacture of fabricated metal products, except machinery and equipment), 26 (Manufacture of computer, electronic and optical products), 27 (Manufacture of electrical equipment) and 28 (Manufacture of machinery and equipment).

```
divs = c( 22,23,24,25,26,27,28, 29, 30, 31)
#divs = c( 22,23,24,25)


selectedCf <- codes %>% filter(division %in% divs) %>% select(cf)
#semi_join() return all rows from first table with a match in second table
companies  <- companies %>% semi_join(selectedCf)
bsd        <- bsd       %>% semi_join(selectedCf) %>% filter(year == 2019)
rating     <- rating    %>% semi_join(selectedCf) %>% filter(year == 2019)

empl.flows <- empl.flows%>% semi_join(selectedCf) %>% filter(year == 2019) %>%
  group_by(cf)%>% summarise(staffTurnover = sum(turnover), staffBalance = sum(balance))

empl.stock <- empl.stock%>% semi_join(selectedCf) %>%
  group_by(cf)%>% summarise(StockAll=max(StockAll)) %>%
  filter(StockAll > 0) %>% filter(StockAll < 1000)
```

Now we have a sample of 20 companies. We can create the dataset X with the relevant features, filter.

```
companies <- companies %>%
       inner_join(bsd, by = "cf") %>%
       inner_join(rating, by = "cf") %>%
       inner_join(empl.flows, by = "cf") %>%
       inner_join(empl.stock, by = "cf")

X <- companies %>% select(idCompany, is.sme, is.startup, is.fem, is.young,
  yearsInBusiness,staffTurnover, staffBalance, StockAll,
  totAssets, totEquity, totIntang,
  accounts, debts,ammort, deprec,prod,revenues, personnel,
  valCost,profLoss, valAdded, noi) %>%
  mutate(is.sme = factor(is.sme)) %>%
  mutate(is.startup = factor(is.startup)) %>%
  mutate(is.fem  = factor(is.fem)) %>%
  mutate(is.young = factor(is.young)) %>%
  filter(yearsInBusiness > 1) %>%
  na.omit()

X %>% write.csv(file = "X.csv",row.names = FALSE) #save data for future applications
```

Labels based on financial ratings

In order to assess the performance of a decision tree under a variety of conditions, we need to build different y labels, appropriate for a binary classification (in the form of a factor variable in R) representative of the actual use in context (the top performers are generally a fraction of 5% to 15% of the selected sample of companies).

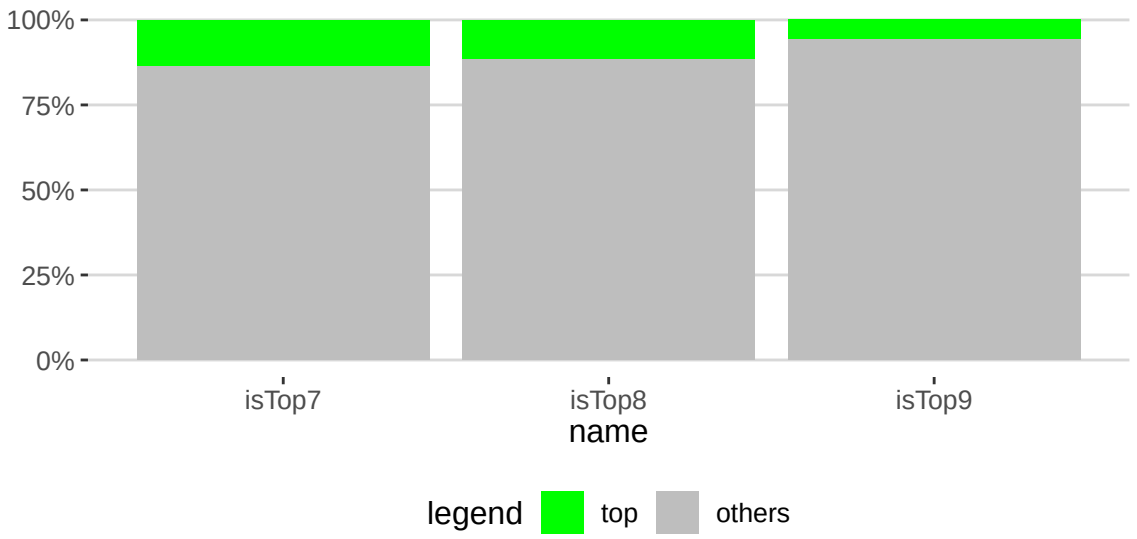
Financial rating is a numerical variable ranging from 1 to 10, where low values denote an insufficient capability to meet financial obligations, and high values denote very good or excellent reliability. Ratings are available for years 2018, 2019 and 2020; for the purpose of this study a single year (2019) will be selected.

```
ys <-companies %>%
  mutate(isTop9 = (rating010 >=9))  %>%
  mutate(isTop8 = (rating010 >=8)&(staffBalance >0))  %>%
  mutate(isTop7 = (rating010 >=7)&(staffBalance >0)&(profLoss>100000))  %>%
  select(idCompany,isTop9, isTop8, isTop7)

ys %>%  write.csv(file = "y.csv",row.names = FALSE) #save data for future applications

proportions <- ys %>% pivot_longer(cols=c(isTop9, isTop8, isTop7)) %>%
  group_by(name) %>%
  summarise(top = round(mean(value),3)) %>%
  mutate(others = 1 - top)
proportions %>% print()
```

```
## # A tibble: 3 x 3
##   name     top others
##   <chr>  <dbl>  <dbl>
## 1 isTop7 0.136  0.864
## 2 isTop8 0.117  0.883
## 3 isTop9 0.059  0.941
```

The three labels are TRUE for a suitable proportion of the dataset (6.7% to 14.6%).

```
proportions %>% pivot_longer(-c(name), names_to = "type", values_to = "value") %>%
  ggplot(aes(x = name, y =value, fill=type))+
  geom_bar(, stat = "identity", position = position_fill(reverse = TRUE)) +
  scale_fill_manual("legend", values = c( "top" = "green","others" = "gray")) +
  ylab("") + scale_y_continuous(labels = scales::percent)
```



Feature engineering

The objective of feature engineering is to build a dataset *data* that contains all relevant variables for learning and prediction, appropriately scaled, in the form of a matrix. In this phase we will focus on the label isTop9 (other labels will be used in the optimization phase).

```
y <- ys %>% mutate(isTop = as.factor(isTop9)) %>%
  select(idCompany,isTop)
```

The rpart library requires a dataset in the form of a single table, that can be obtained joining X and y. The company identifier can be removed at this stage, since it carries no information on the company and may only lead to overfitting.

```
data <- X %>% inner_join(y) %>% select(-idCompany)
```

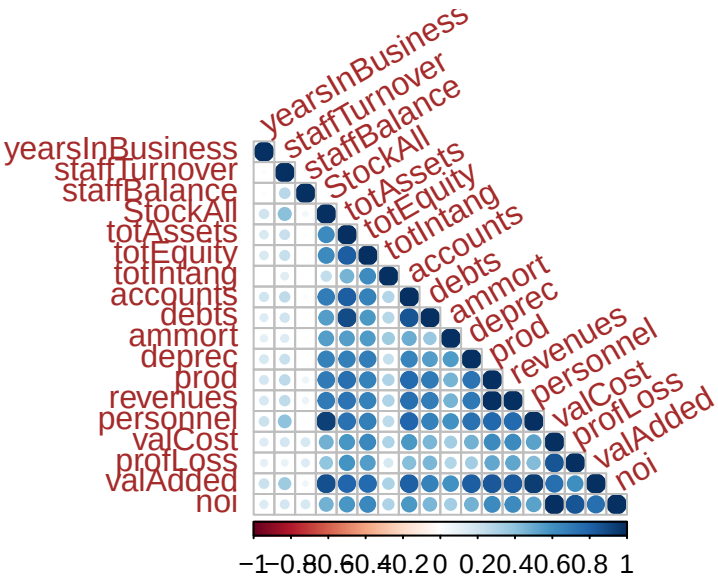
The dataset is composed of n = 2008 observations and p = 23 features (namely: is.sme, is.startup, is.fem, is.young, yearsInBusiness, staffTurnover, staffBalance, StockAll, totAssets, totEquity, totIntang, accounts, debts, ammort, deprec, prod, revenues, personnel, valCost, profLoss, valAdded, noi, isTop). There meaning of variables is self explanatory in some cases, but economic features need to be explicitly mentioned:

- totAssess: totale attivo = total assets
- noi: RON Reddito Operativo Nnetto = NOI Net Operating Income
- personnel: totale costi del personale = total personnel costs
- debts: debiti esigibili entro l'esercizio successivo = debts due within the following financial year
- totEquity: totale patrimonio netto = total equity
- profLoss: utile/perdita esercizio ultimi = profit / loss for the last financial year

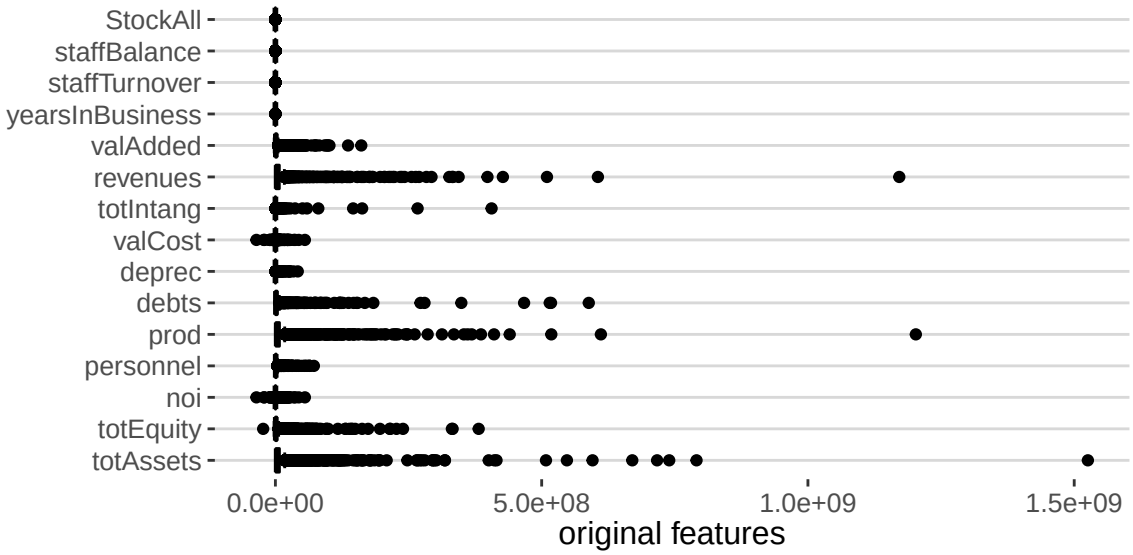
- accounts: crediti esigibili entro l'esercizio successivo = accounts receivables
- totIntang: totale immobilizzazioni immateriali = total intangible fixed assets
- prod: totale valore della produzione = total production value
- revenues: ricavi delle vendite = revenues from sales
- valCost: differenza tra valore e costi della produzione = difference between production value and production costs
- ammort: ammortamento immobilizzazione immateriali = amortisation
- valAdded: valore aggiunto = value added
- deprec: tot.aam.acc.svalutazioni = total amortisation, depreciation and write-downs

We can check multycollinearity issues by plotting a correlation plot. Several features have a high correlation that should be avoided.

```
cm <- X %>% select(where(is.numeric), -idCompany) %>% cor()
cm %>% corrplot( tl.col = "brown", tl.srt = 30, bg = "White",title = "",type = "lower")
```



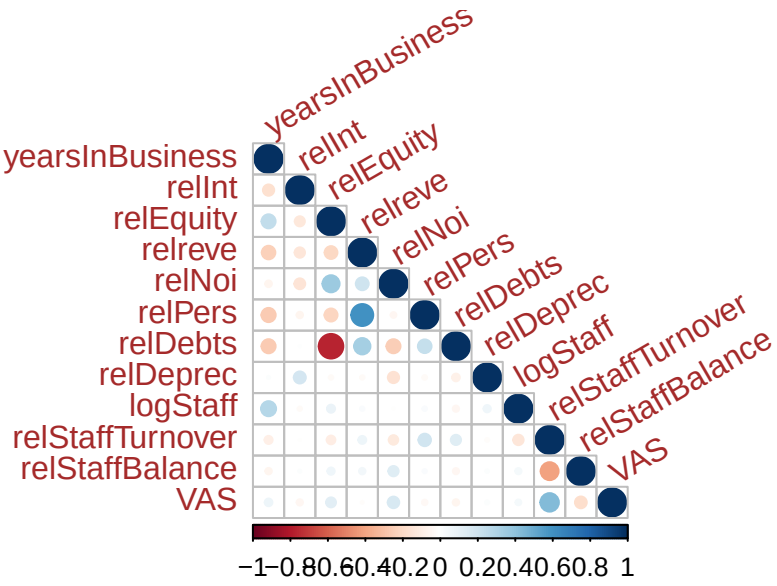

```
data <- data %>% select(totAssets,totEquity, noi,personnel,prod, debts,deprec,valCost,totIntang, revenu
ggplot(stack(data), aes(x = ind, y = values)) +
 stat_boxplot(geom = "errorbar", width = 0.5) +
 labs(x="", y="original features") +
 geom_boxplot(fill = "white", colour = "black") + coord_flip()
```



Since the size of company varies greatly within the dataset, performance can be compared using economic indicators scaled to the total assets (totAssets) or the total number of employees (StockAll) of each company.

```
data <- data %>%
    mutate(relInt =   totIntang/totAssets*5) %>%
    mutate(relEquity = totEquity/totAssets) %>%
    mutate(relreve   = revenues/totAssets) %>%
    mutate(relNoi    = noi/totAssets) %>%
    mutate(relPers   = personnel/totAssets) %>%
    mutate(relDebts  = debts/totAssets) %>%
    mutate(relDeprec = deprec/totAssets*5) %>%
    mutate(yearsInBusiness = yearsInBusiness)  %>%
    mutate(logStaff  = log10(StockAll)) %>%
    mutate(relStaffTurnover = staffTurnover/StockAll) %>%
    mutate(relStaffBalance  = staffBalance/StockAll) %>%
    mutate(VAS  = valAdded/StockAll) %>%
   select(-totAssets,-totEquity, -noi,-personnel,-debts,-prod,
          -deprec,-revenues,-valCost,-totIntang,-valAdded,
          -StockAll, -staffTurnover, -staffBalance)%>%
   na.omit()
```

```
cm <- data %>% select(where(is.numeric)) %>% cor()
cm %>% corrplot( tl.col = "brown", tl.srt = 30, bg = "White",
         title = "", type = "lower")
```




```
#value of the minimun correlation coefficient: red dot in the plot
cm %>% as_tibble %>% select(relDebts) %>% min()
```

```
## [1] -0.7654333
```

```
data <-data %>% select(-relDebts)
```

Keeping only the new features, the dataset consists of n = 2008 observations and p = 14 features (namely: yearsInBusiness, is.startup, is.fem, isTop, relInt, relEquity, relreve, relNoi, relPers, relDeprec, logStaff, relStaffTurnover, relStaffBalance, VAS). The next step is to normalize (center and rescale) numeric features to the a similar range in order to improve the performance of the learning algorithm.

```
data <- data %>% mutate(across(is.numeric, ~ as.numeric(scale(.))))
```

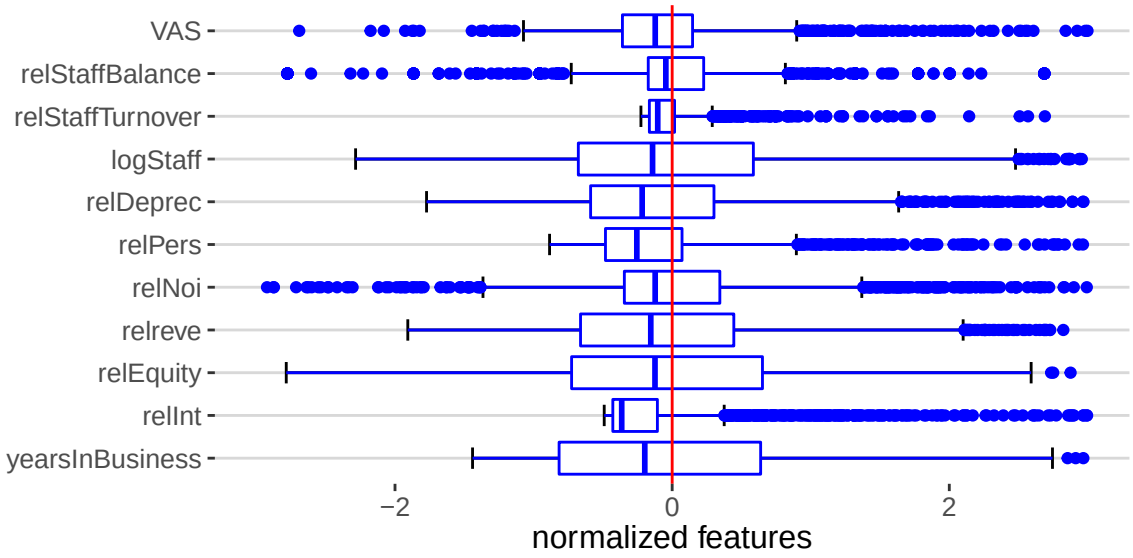
```
## Warning: Predicate functions must be wrapped in `where()`.
##
##   # Bad
##   data %>% select(is.numeric)
##
##   # Good
##   data %>% select(where(is.numeric))
##
## i Please update your code.
## This message is displayed once per session.
```

```
data %>% stack() %>% ggplot(aes(x = ind, y = values)) +
 stat_boxplot(geom = "errorbar", width = 0.5,outlier.size=.1) +
 labs(x="", y="normalized features") +
 geom_boxplot(fill = "white", colour = "blue") + coord_flip()  +  scale_y_continuous(  limit = c(-3, 3))
```

```
## Warning in stack.data.frame(.): non-vector columns will be ignored
```

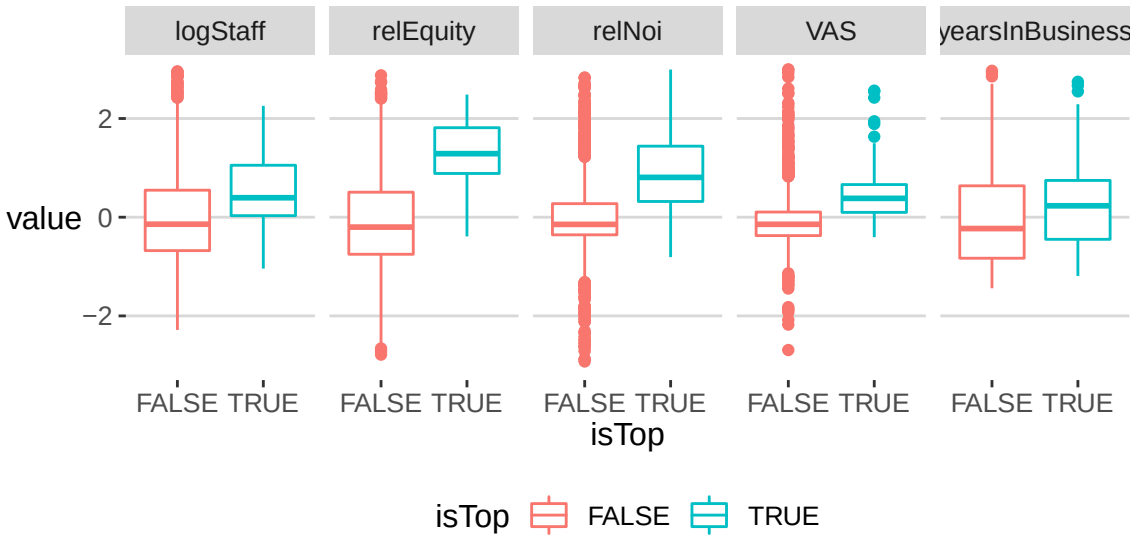
```
## Warning: Ignoring unknown parameters: outlier.size
```

```
## Warning: Removed 266 rows containing non-finite values (stat_boxplot).

## Warning: Removed 266 rows containing non-finite values (stat_boxplot).
```



All features are of the same order of magnitude, but have not the same predictive power on y label. The decision tree will provide a detailed estimate of each variable, but at this stage we can have a basic idea of the predictive power of each variable by examining the distributions of observations according to the label

```
data %>%
  select(relEquity, relNoi, logStaff, VAS,yearsInBusiness,isTop)%>%
  pivot_longer(cols=!isTop) %>% ggplot(aes(x=isTop, y=value, color = isTop)) +
  geom_boxplot() + facet_wrap(~name, ncol = 6) +
  scale_y_continuous(  limit = c(-3, 3))
```



Some features, namely *relNoi* (net operative income scaled to total assets), relEquity (total equity scaled to total assets) and VAS (Value Added on total Staff), are clearly separate according to label y9 and can therefore be expected to be good predictors.

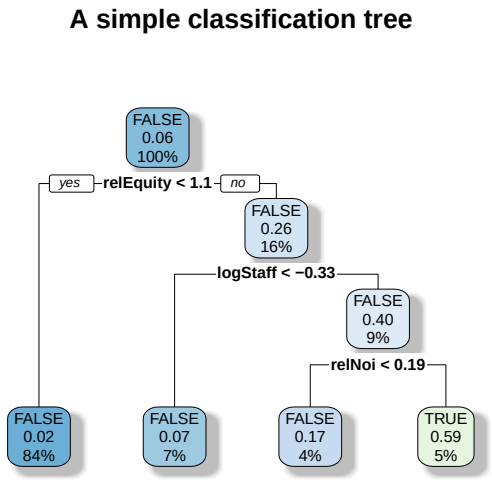
Learning: Binary Classification Trees

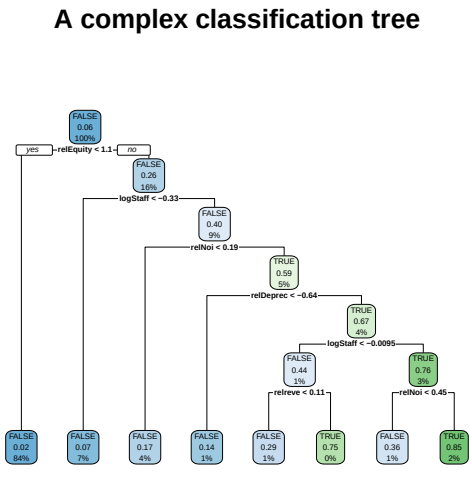
The proposed solution is to apply "Binary Classification Trees" and check if the model can be tuned to achieve the expcted performance. The solution can be implemented using the two usual blocks: - a function for learning the model `rpart()` - a function for prediction `predict()`

The learning function in `rpart()` requires data in the form of a matrix (a data.frame or a tibble) and instructions on which features should be predicted, expressed as a *formula*. In our case the formula used for all trees is `isTop ~ .` which stands for "*feature* ***`isTop`*** *depends on all the other variables*".

The model complexity and performance will be controlled by a single parameter, minsplit, i.e. the minimum number of observations that must exist in a node in order for a split to be attempted. A high value of minsplit produces simple trees with large terminal nodes (example on the left) while low values of minsplit produces complex trees (example on the right). Optimal performance can be achieved selecting an appropriate value of minsplit that produces the best results on test data. Such value will be determined experimentally in the next section of the notebook.

```
figures <- par(mfrow=c(1,2)) # more trees in the same figure

#building a simple tree
parms = rpart.control(split = "gini", minsplit = 100, cp = .02 )
simple.tree <- rpart(isTop~., data, method = "class", control=parms)
simple.tree %>% rpart.plot(  main="A simple classification tree", shadow.col = "gray")

#building a complex tree
parms = rpart.control(split = "gini",  minsplit = 20, cp=.01 )
complex.tree <- rpart(isTop~., data, method = "class",control=parms)
complex.tree %>% rpart.plot(  main="A complex classification tree")
```





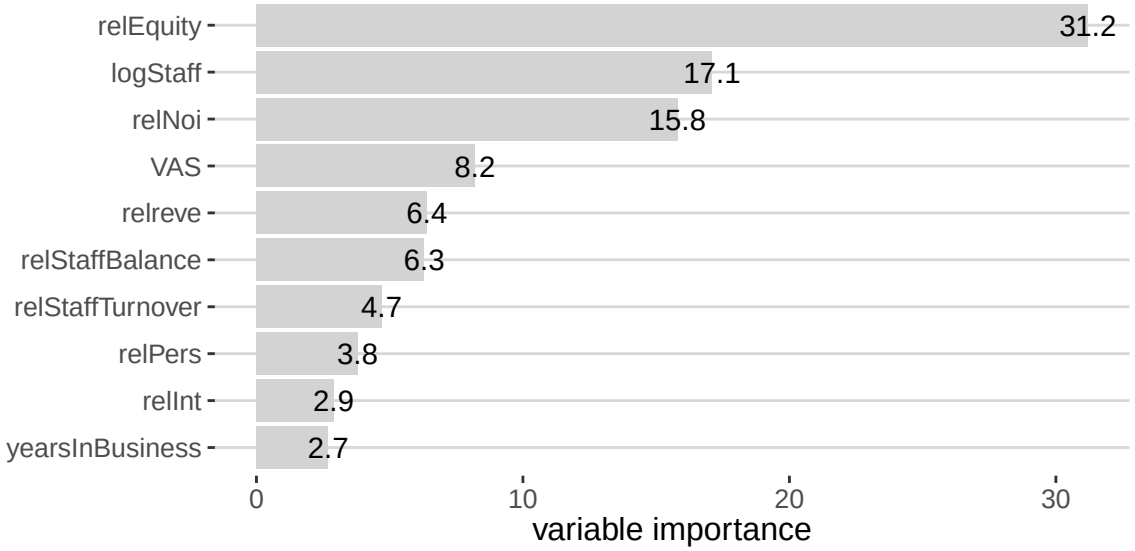

```
par(figures)
```

The models exhibit different complexity, accuracy and explainability.

variable importance

The recursive partitioning algorithm calculates the relative importance of each variable to achieve the final predictions. Values vary with data and hyperparameters.

```
tibble(features=names(simple.tree$variable.importance) , imp=simple.tree$variable.importance) %>%
 mutate(imp = round(imp,1))%>%
 ggplot(aes(reorder(features, imp, sum), imp))  + coord_flip()+
 labs(x="", y="variable importance") +
 geom_bar(stat="identity", fill="light gray")+
 geom_text(aes(label=imp))
```



In the case of simple.tree model, the most relevant variables are relEquity, logStaff and relNoi. Selecting the most predictive variables improves the performance of the algorithm. Selecting predictors with low predictive power can lead, in fact, to overfitting or low model performance.

3. Performance assessment and optimization

Quality of prediction of a single tree

[explain]

- error rate
- accuracy
- confusion matrix **confusion matrix** is a table (2x2 in the case of binary classifiers) that shows the number of values classified correctly (on the main diagonal) or misclassifications.

```
computeConfusionMatrix = function(model, data) {
  predicted.y = predict(model, data, type="class")
  errors <- data %>%
    mutate(err = (isTop = predicted.y)) %>%
    filter(err == TRUE) %>% nrow()
  return(table(predicted.y, data$isTop))
}
```

we might build a function that computes the accuracy:

```
computeAccuracy =  function(model, data) {
  conf.matrix <- computeConfusionMatrix (model, data)
  accuracy <- sum(diag(conf.matrix))/sum(conf.matrix)
  return(round(accuracy,5))
}
```

we can use the function as follows TODO: apply to simple and complex tree above (not to another tree!)

```
#split data into learn and test
p.learn = .8
indexes.learning = sample(c(1:nrow(data)))[1:(nrow(data)*p.learn)]
data.learn = data[ indexes.learning,]
data.test =  data[-indexes.learning,]

print("Confusion matrix and accurcy of simple tree on learning data:")
```

```
## [1] "Confusion matrix and accurcy of simple tree on learning data:"
```

```
print( computeConfusionMatrix(simple.tree, data.learn) )
```

```
##
## predicted.y FALSE TRUE
##       FALSE  1474   56
##       TRUE     33   43
```

```
print( computeAccuracy(simple.tree, data.learn) )
```

```
## [1] 0.94458
```

```
print("Confusion matrix and accurcy of simple tree on test data:")
```

```
## [1] "Confusion matrix and accurcy of simple tree on test data:"
```

```
print( computeConfusionMatrix(simple.tree, data.test ) )
```

```
##
## predicted.y FALSE TRUE
##       FALSE   376    4
##       TRUE      7   15
```

```
print( computeAccuracy(simple.tree, data.test)  )
```

```
## [1] 0.97264
```

The learning and test datasets are definied as a random sample of the data, therefore they are different every time the program is executed. The model (and its performance) vary accordingly. In order to achieve a stable performance assessment we need to repeat the procedure several times, and return the average result.

Overfitting

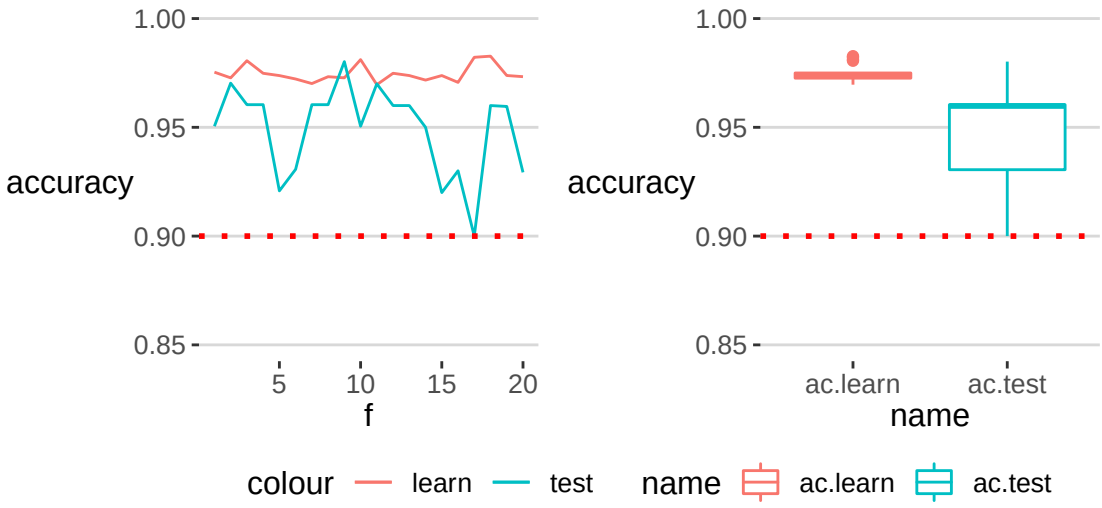
How will the model perform on "unseen data"? can the learner generalize beyond available data? The classification error is an appropriate indicator. But the error depends on a random choice of learn and test data. We need to have a more stable value, so we introduce k-fold cross validation on the whole dataset.

1. split learning data (X and y) in k equal slices (each of n k observations)
2. for each split (i.e., each i {1, . . . , k}) 2.1 learn on all but k-th slice 2.2 compute classification error on unseen k-th slice
3. average the k classification errors

```
compute.Kfold.accuracy <- function(data,  k_folds=10, minsplit, cp) {
  data_f <- data %>% sample_frac(1) %>%
  group_by(isTop) %>%
  mutate(fold=rep(1:k_folds, length.out=n())) %>% ungroup

  folds.results = tibble(f = NA, ac.learn = NA, ac.test = NA,  n = NA, minsplit = NA) %>% head(0)

  for(i in 1:k_folds){
    data.kth.fold <- data_f %>% filter(fold==i)
    data.other.folds <- data_f %>% filter(fold!=i)

    # learn on  data.other.folds
    model <- rpart(  isTop~.,data.other.folds,method = "class", minsplit = minsplit, cp = cp)

    #save results
    folds.results <-  folds.results %>%
    add_row(f = i ,
        ac.learn = computeAccuracy(model, data.other.folds ),
        ac.test =  computeAccuracy(model, data.kth.fold    ),
        n = nrow(model$frame),
        minsplit = minsplit)
  }

  return(folds.results)
}
```

Now we can use the compute.Kfold.accuracy() function to show how accuracy varies in different tests. Accuracy on learning data is higher and more stable, while accuracy on test shows larger variations.

```
folds.results = compute.Kfold.accuracy(data,  k_folds=20, minsplit=1, cp=.01)

p01 <- ggplot(data = folds.results)+geom_line( aes(x=f, y=ac.learn,color = "learn"))+
  geom_line( aes(x=f, y=ac.test, color = "test")) + ylim(0.85,1.0)+
  geom_hline(yintercept = .90, linetype = 'dotted', col = 'red', size=1)+ylab("accuracy")


p02 <- folds.results %>% pivot_longer(cols=c(ac.learn, ac.test)) %>%
  ggplot(aes(x=name, y=value, color = name))+geom_boxplot()+ ylim(0.85,1.0)+
  geom_hline(yintercept = .90, linetype = 'dotted', col = 'red', size=1)+ylab("accuracy")

wrap_plots(p01,p02, guides = 'collect')
```

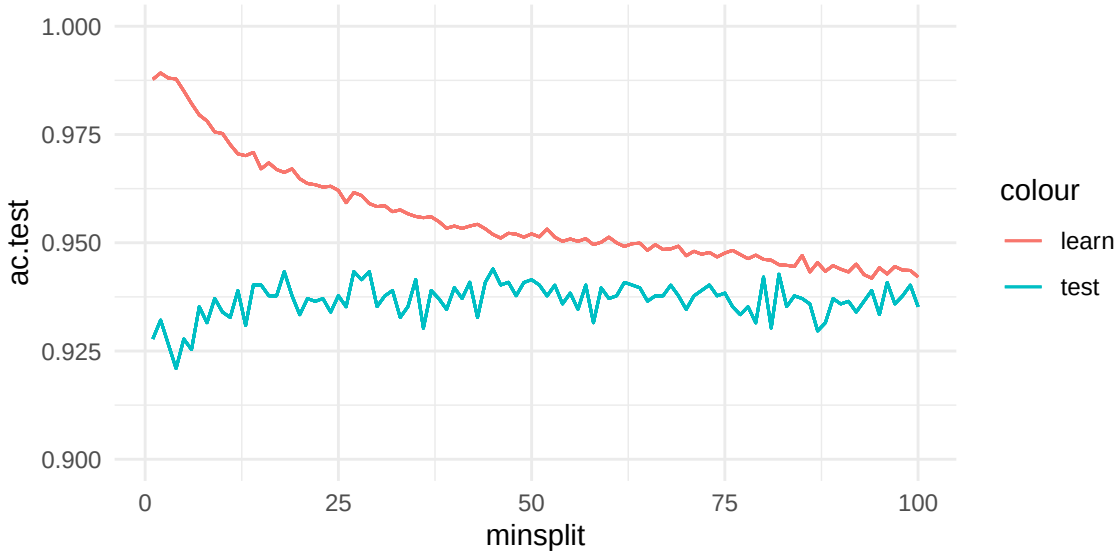


Optimization (nested k-fold cross validation)

Moreover, performance varies with tree complexity. a complex tree will perfectly fit the data, but will overfit the model. We can spot this behaviour "overfitting" by plotting accuracy on test and learn data vs a complexity parameter.

Here we demonstrate a tuning procedure: the graph shows the mean accuracy for each value of minsplit

```
tuning.results = tibble()

for (mspl in seq(1,100,1)){
    tuning.results <- tuning.results %>%
      bind_rows(compute.Kfold.accuracy(data.learn,  k_folds=10, minsplit=mspl, cp=.01 ))
}

tuning.results %>% group_by(minsplit)%>%
  summarise(ac.test=mean(ac.test), ac.learn=mean(ac.learn), f=f)%>%
  group_by(f)%>%
  ggplot() + theme_minimal()+
  geom_line(aes(x=minsplit, y = ac.test, group = f, colour = 'test')) +
  geom_line(aes(x=minsplit, y = ac.learn,group = f, colour = 'learn'))+
  ylim(0.90,1.0)#+geom_hline(yintercept = .90, linetype = 'dotted', col = 'red', size=1)
```



We build a procedure based on 2 nested k-fold cross validation

the outer loop is performance assessment the inner loop is tuning

```
AutoTuneTree <- function(data.learn, data.test, k_folds_testing, k_folds_tuning, minsplit_candidates) {

  #split data into folds
  data_f <- data %>% sample_frac(1) %>% group_by(isTop) %>%
    mutate(fold=rep(1:k_folds_testing, length.out=n())) %>% ungroup
  folds.results = tibble(f=NA,ac.learn=NA,ac.test=NA,n=NA,minsplit=NA)%>% head(0)

  #outer cycle: testing
  for(i in 1:k_folds_testing){
    #select the fold to be used for testing
    data.kth.fold <- data_f %>% filter(fold==i)
    data.other.folds <- data_f %>% filter(fold!=i)
    tuning.results = tibble()

    # inner cycle: tuning
    for (mspl in minsplit_candidates){
      tuning.results <- tuning.results %>%
        bind_rows(compute.Kfold.accuracy(data.learn,  k_folds=k_folds_tuning, minsplit=mspl, cp=.01 ))
    }

    #select optimal value for minsplit
    mspl.star = tuning.results %>%
      group_by(minsplit) %>%
      summarise(ac.learn = mean(ac.learn), ac.test = mean(ac.test), n = n) %>%
      arrange(desc(ac.test)) %>%
      head(1) %>% pull

    model <- rpart(  isTop~.,data.other.folds,method = "class", minsplit = mspl.star, cp = .01)
    #save results
    folds.results <-  folds.results %>%
      add_row(f = i ,
              ac.learn = computeAccuracy(model, data.other.folds ),
```

```
              ac.test =  computeAccuracy(model, data.kth.fold    ),
              n = nrow(model$frame),
              minsplit = mspl.star)
    print(mspl.star)
  }
  # sort fold results on descending values of ac.test and minsplit
  # fold.results[1] is the highest minsplit that produces the highest accuracy
  return(folds.results %>%arrange(desc(ac.test, minsplit)))
}
```

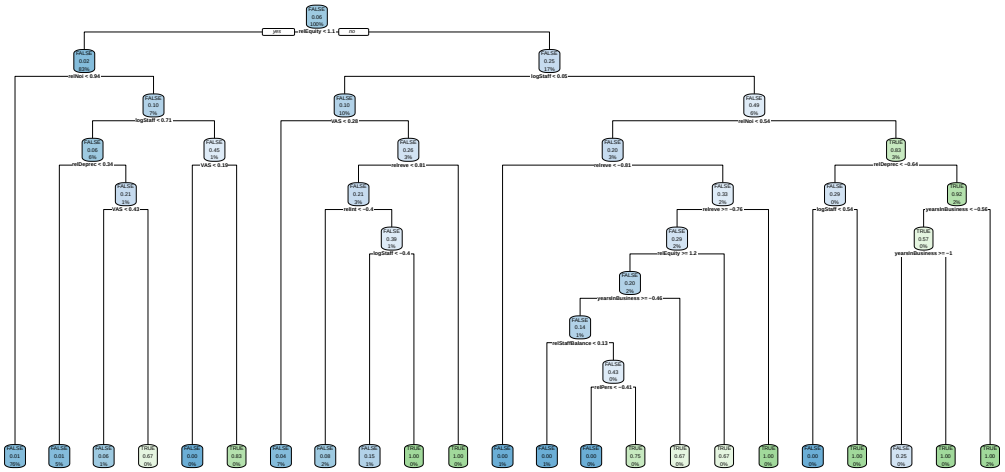
Now we can apply the function

```
#split train and test data
fraction = .8
indexes.learning = sample(c(1:nrow(data)))[1:(nrow(data)*fraction)]
data.learn <- data %>% slice( indexes.learning)
data.test  <- data %>% slice(-indexes.learning)

# define the range for minsplit parameter
minsplit_candidates = seq(1,100,1)

#define k-fold validation (computational cost vs quality)
k_folds_tuning=3
k_folds_testing=3

optimal.params <- AutoTuneTree(data.learn, data.test, k_folds_testing, k_folds_tuning, minsplit_candida
```

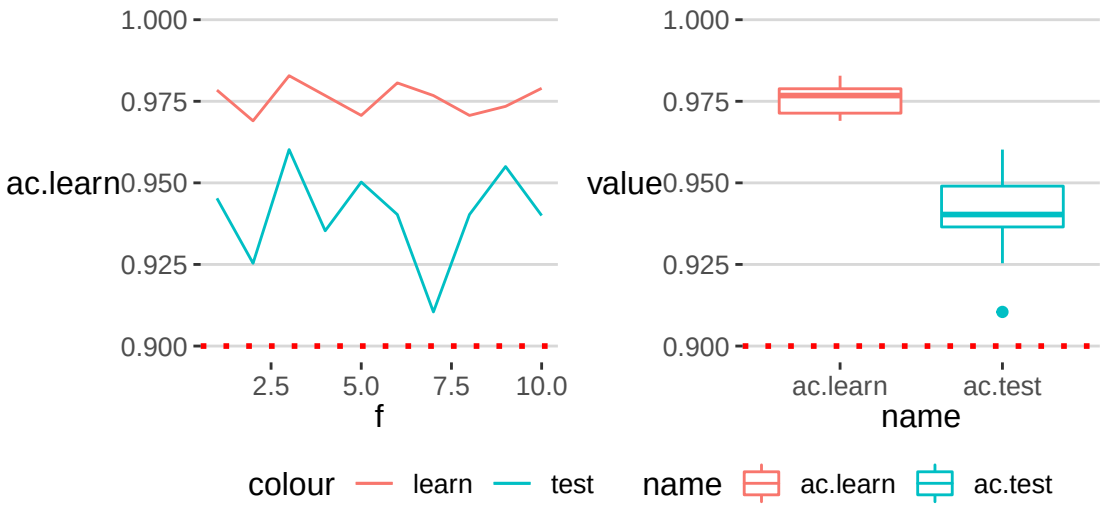
```
## [1] 7
## [1] 7
## [1] 7
```

final result

```
optimal.minsplit  <- optimal.params %>% head(1) %>% select(minsplit) %>% pull
optimal.acc.learn <- optimal.params %>% head(1) %>% select(ac.learn) %>% pull
optimal.acc.test  <- optimal.params %>% head(1) %>% select(ac.test) %>% pull

optimal.tree <- rpart(  isTop~.,data.learn,
                        method = "class",
                        minsplit = optimal.minsplit,
                        cp = .01)
rpart.plot(optimal.tree)
```

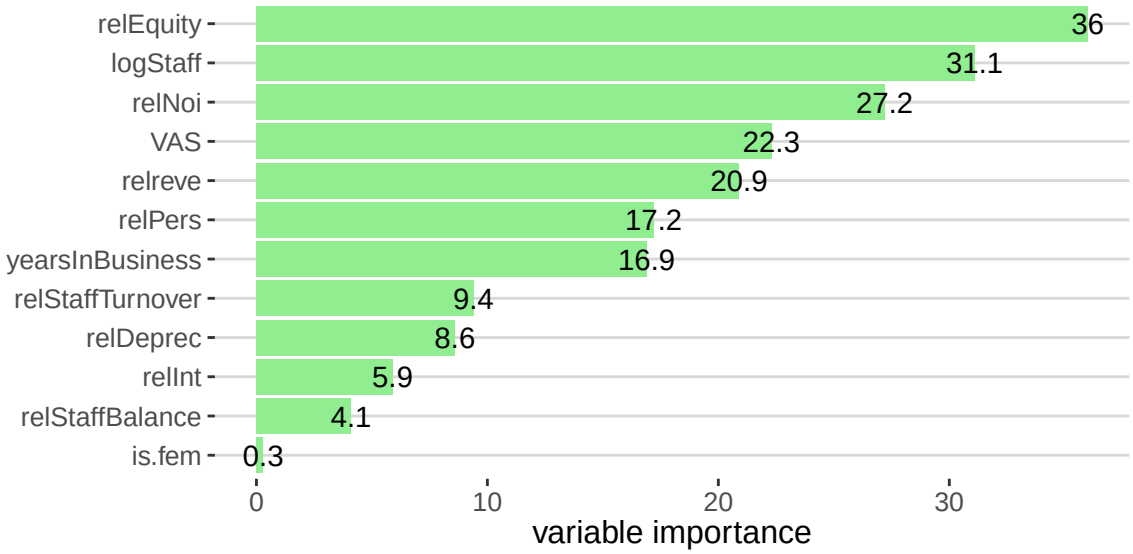


```
folds.results = compute.Kfold.accuracy(data,  k_folds=10, minsplit=optimal.minsplit, cp=.01)

p_9 <- ggplot(data = folds.results)+geom_line( aes(x=f, y=ac.learn,color = "learn"))+
  geom_line( aes(x=f, y=ac.test, color = "test")) + ylim(0.90,1.0)+
  geom_hline(yintercept = .90, linetype = 'dotted', col = 'red', size=1)

plot9 <- folds.results %>% pivot_longer(cols=c(ac.learn, ac.test)) %>%
  ggplot(aes(x=name, y=value, color = name))+geom_boxplot()+ ylim(0.90,1.0)+
  geom_hline(yintercept = .90, linetype = 'dotted', col = 'red', size=1)

wrap_plots(p_9,plot9, guides = 'collect')
```




```
tibble(features=names(optimal.tree$variable.importance) , imp=optimal.tree$variable.importance) %>%
  mutate(imp = round(imp,1))%>%
  ggplot(aes(reorder(features, imp, sum), imp))  + coord_flip()+
```

```
  labs(x="", y="variable importance") +
  geom_bar(stat="identity", fill="light green")+
  geom_text(aes(label=imp))
```



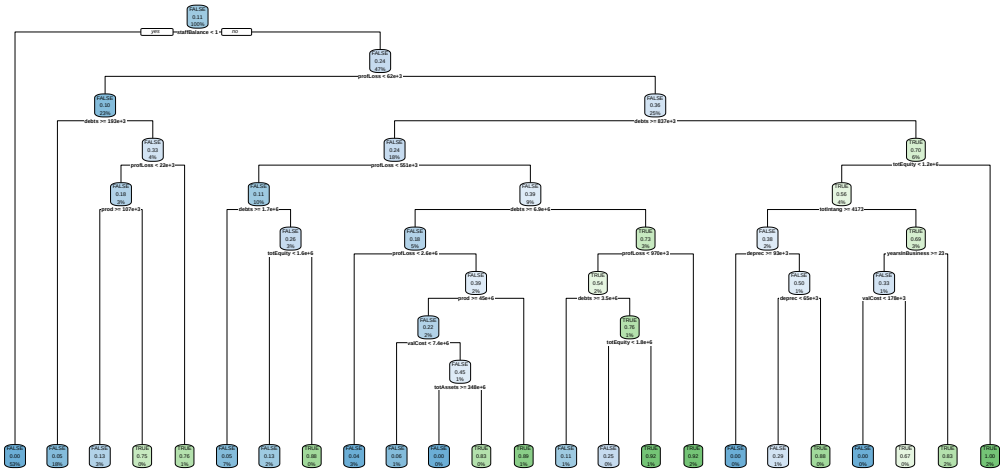
4. Robustness of the solution against different definitions of y

Robustness refers to the effectiveness of the algorithm when tested on the new independent (but similar) datasets. In the other words, the robust algorithm is the one, the testing error of which is close to the training error.

```
y8 <- ys %>% mutate(isTop = as.factor(isTop8)) %>%  select(idCompany,isTop)
data8 <- X %>% inner_join(y8) %>% select(-idCompany)
data.learn <- data8 %>% slice( indexes.learning)
data.test  <- data8 %>% slice(-indexes.learning)

optimal.params8 <- AutoTuneTree(data.learn, data.test, k_folds_testing, k_folds_tuning, minsplit_candida
```

```
## [1] 27
## [1] 15
## [1] 37
```

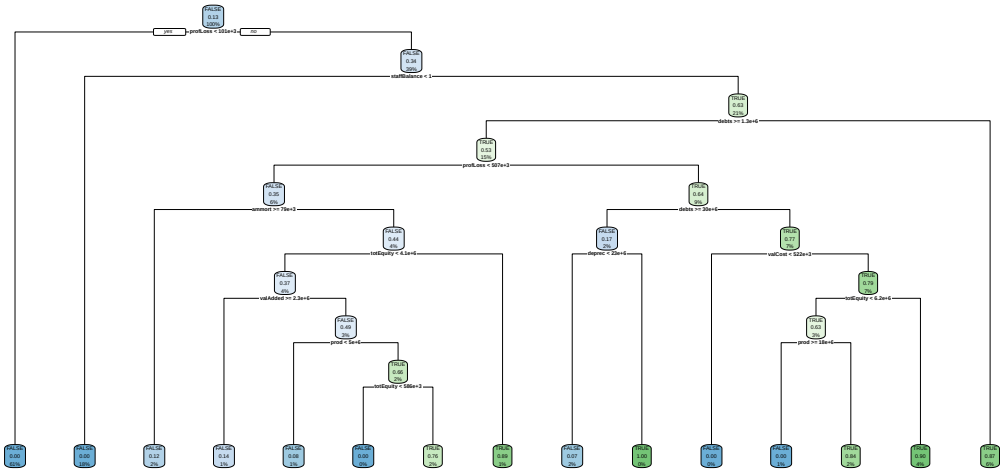
```
optimal.minsplit8 <-  optimal.params %>% head(1) %>% select(minsplit) %>% pull
optimal.tree8 <- rpart( isTop~.,
                        data.learn,
                        method = "class",
                        minsplit = optimal.minsplit8,
                        cp = .01)
rpart.plot(optimal.tree8)
```



```
y7 <- ys %>% mutate(isTop = as.factor(isTop7)) %>%  select(idCompany,isTop)
data7 <- X %>% inner_join(y7) %>% select(-idCompany)
data.learn <- data7 %>% slice( indexes.learning)
data.test  <- data7 %>% slice(-indexes.learning)

optimal.params7 <- AutoTuneTree(data.learn, data.test, k_folds_testing, k_folds_tuning, minsplit_candida
```

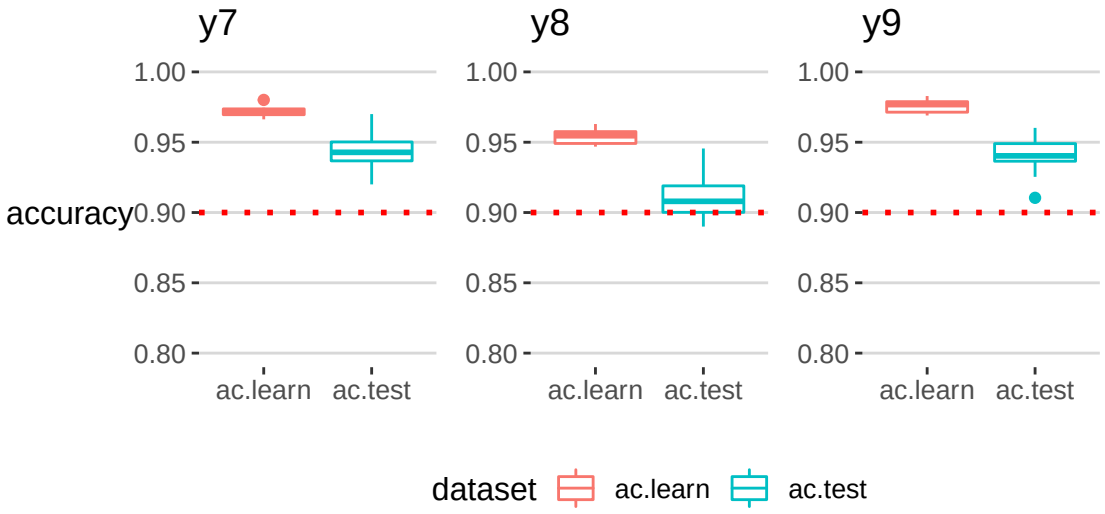
```
## [1] 17
## [1] 11
## [1] 19
```

```
optimal.minsplit7 <-  optimal.params %>% head(1) %>% select(minsplit) %>% pull
optimal.tree7 <- rpart( isTop~.,
                        data.learn,
                        method = "class",
                        minsplit = optimal.minsplit7,
                        cp = .01)
rpart.plot(optimal.tree7)
```



...

```
folds.results7 <- compute.Kfold.accuracy(data7,  k_folds=10, minsplit=optimal.minsplit7, cp=.01)
plot7 <- folds.results7 %>% pivot_longer(cols=c(ac.learn, ac.test)) %>%
  mutate(dataset =name, accuracy = value)%>%
  ggplot(aes(x=dataset, y=accuracy, color = dataset))+geom_boxplot()+ ylim(0.8,1.0)+
  geom_hline(yintercept = .90, linetype = 'dotted', col = 'red', size=1) +
  labs(title = "y7") + xlab("")

folds.results8 <- compute.Kfold.accuracy(data8,  k_folds=10, minsplit=optimal.minsplit8, cp=.01)
plot8 <- folds.results8 %>% pivot_longer(cols=c(ac.learn, ac.test)) %>%
  mutate(dataset =name, accuracy = value)%>%
  ggplot(aes(x=dataset, y=accuracy, color = dataset))+geom_boxplot()+ ylim(0.8,1.0)+
  geom_hline(yintercept = .90, linetype = 'dotted', col = 'red', size=1) +
  labs(title = "y8")+ xlab("")+ ylab("")

plot9 <- folds.results %>% pivot_longer(cols=c(ac.learn, ac.test)) %>%
  mutate(dataset =name, accuracy = value)%>%
  ggplot(aes(x=dataset, y=accuracy, color = dataset))+geom_boxplot()+ ylim(.8,1.0)+
  geom_hline(yintercept = .90, linetype = 'dotted', col = 'red', size=1)   +
  labs(title = "y9")+ xlab("") + ylab("")


wrap_plots(plot7,plot8,plot9, guides = 'collect')
```



A wider definition of the label y results in a lower quality of predictions, still above the selected threshold accuracy > 0.90, with a much more complex mode.

5. Concluding remarks

Feasibility

This notebook demonstrates that a machine learning algorithm can predict whether a company belongs to the group of "top performers", given a sufficiently narrow definition of the label and sufficiently large dataset. The feasibility has been demonstrated for a binary classification algorithm (namely a single tree, generated by the recursive partitioning algorithm of r-part library. The dataset used for learning and validation is composed of n = 2008 observations and n = 14 features, obtained from a larger dataset of regional companies, filtered on NACE sectors of activity and on year 2019.

THree case studies have been examined in detailed, using different definitions of the y label (isTop9, a narrow definition addressing 6% of companies, isTop8 and isTop7 providing boader definitions). The model has been tuned to maximize accuracy on test data using a nested k-fold cross validation loops for tuning and assessing experimentally the performance,

in both cases computation time is of the order of minutes, the model is interpretable and explainable and the accuracy of predictions has been measued experimetnally. In 2 cases is above the treshod, and in another case is below the threshold $accuracy > 90$.

The quality of predictions depends on the selection of companies included in the dataset: we may expect that smaller, homogeneous sets may result in highre accuracy (e.g. selecting only companies of a given size, or of a single NACE sector) and the exact definition of the label. In future applications the model *should be learned for specific subsets and specific definitions of label*, and accuracy (or quality of predictions) has to be evaluated for each case.

Future work

Preliminary results suggest further investigations:

- how data on employment (staff turnover, balance and stock) influence the quality of predictions. Such data is not available for companies outside FVG.

- robustness with respect to differet definition of label y isTop. If we use a different definition to rating =8 or ==7...
- performance with other supevised models: multiclass decision trees, random forrest, svm, naive bayes, knn
- unsupervised: PCA on all variables, intrinsic dimension and k-means clustering
- explore time-dependent analysis: eg how data from 2018 can predict perfomance in 2019, and mode in depth in 2020 (where a disruptive change in economy took place).

References

- innovation intelligence
- nace
- ISL
- papers

Acknowlegments: Marinella Favot for suggesting new indicators, and specifically VAS, that resulted a very effective predictor of y9.