A Machine Learning approach to classification of companies

Fabio Morea

2022-02-14

Contents

1. Introduction

This notebook is an excercise 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 8 to 10
- Mid: rating 5 to 7
- 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 receive different proposals...).

The issues with this methodology are the cost of financial ratings, and the lack of explainability (financial ratings are based on a proprietary algorithm). Financial ratings can be used for training the ML model, while prediction should be based open data available in the Innovation Intelligence dataset (sector, age, balance sheet data,...).

The **target audience** of the feasibility study is 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 R-notebook presenting a detailed case study, including a ML model and an assessment of its 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 Innofation Intelligence 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.*

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 data manipulation is based on *tidyverse* [https://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).

2. Methodology

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) and explainable (we can inspect each step of the algorithm and assess its importance to the model performance).

The method consists of 3 phases: **pre-processing** in order to select and rescale features to be processed by the classifier, **learning phase**, in which the classifier is trained using derived from the financial rating, and the actual **classification phase**, in which a company is classified as being a Top Performer or not. The Top Performer label is based on financial ratings, while the prediction uses only features from other data sets, such as "balance sheet data" and "employees".

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

$$Accuracy > .85$$

The threshold may seem relatively high compared if compared to a 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 choosen are a tradeoff between complexity and explainability: for the purpose of the feasibility study, explainability and interpretability are to be preferred on complexity, but further development using more complex methods and indicators is outlined in the final section of this notebook.

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 85%, 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

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 (Annex 1).

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 EUEOSTAT and recognised 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)
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(turnover = sum(turnover), balance = sum(balance))

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



# names can be used for debug purposes to check the identiy of each company
# names <- companies %>% select(name,cf,idCompany)
```

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, is.fore,
                              yearsInBusiness,
                              turnover, balance, StockAll,
                              totAssets, totEquity, totIntang,
                              accounts, debts,ammort, deprec,prod,revenues, personnel,
                              valCost,profLoss, valAdded, noi)
```

Labels based on financial ratings

We need to build appropriate labels for a binary classification.

> TODO: improve description: The credit ratings are a commercial dataset, purchased by ModeFinance (a rating agency based in Trieste). TODO: improve description modeFinance Srl is registered as a credit rating agency in accordance with Regulation (EC) No 1060/2009 of the European Parliament ad of the Council of 16 September 2009 (the Credit Rating Agencies Regulation link). MORE Methodology is used by modeFinance also as part of the process of issuance of Credit Ratings in compliance with Regulation (EC) No 1060/2009 of the European Parliament ad of the Council of 16 September 2009 (the Credit Rating Agencies Regulation). TODO improve description: Ratings are evaluated through an innovative methodology called Multi Objective Rating Evaluation which is owned by modeFinance. This innovative methodology studies a corporation as a complex system and deepens the analysis on its different aspects: solvency, debt coverage, liquidity, cash conversion cycle, profitability, fixed asset coverage, compared with the sector which it belongs to and so on.

We filter on a single year (2019) and add "rating" as a new feature. It is a numerical feature ranging from 1 to 10.

```
y <- companies %>%
       mutate(isTop = (rating010 >=9))  %>%
       select(idCompany,isTop)%>%
       mutate(isTop = as.factor(isTop))
```

The dataset will be used as a single tibble "data" and removing idCompany

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

Feature engineering

We need to select most relevant features for prediction from the following: is.sme, is.startup, is.fem, is.young, is.fore, yearsInBusiness, turnover, balance, StockAll, totAssets, totEquity, totIntang, accounts, debts, ammort, deprec, prod, revenues, personnel, valCost, profLoss, valAdded, noi, isTop

Based on domain knowledge, the most relevant atttributes are

- totAssess = totale attivo = total assets
- noi = (ron) reddito operativo netto = (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

others, that may be included:

- totEquity = totale patrimonio netto = total equity

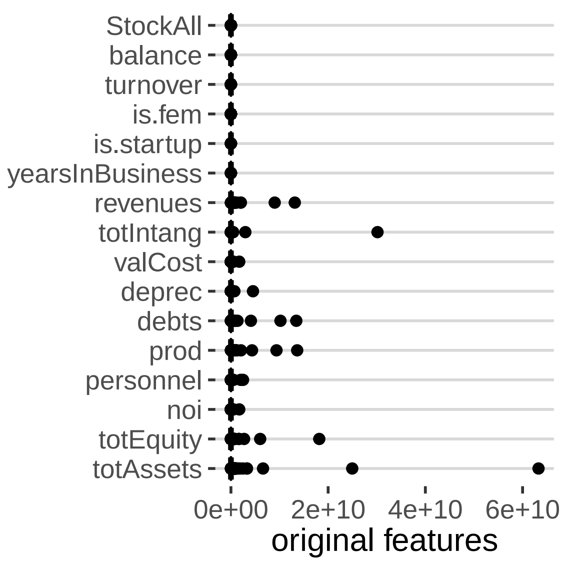
- profLoss = uile/perdita esercizio ultimi = profit / loss for the last financial year
- accounts = crediti esigibili entro l'esercizio successivo = accounts receivables
- totIntang = totale immobilizzazioni immateriali = total intangible fixed assets
- prod = totale valore della produzione = total production 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

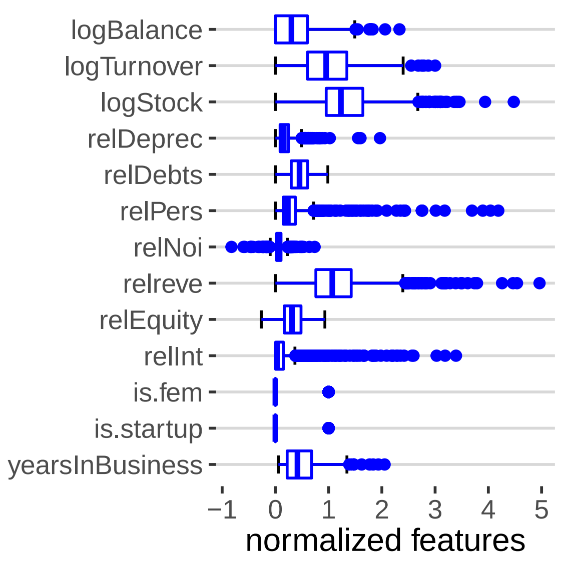
We check for the scale of each variable: variables should be rescaled to the a similar range in order to improve the performance of the learning algorithm

```
data <- data %>% select(totAssets,totEquity, noi,personnel,prod, debts,deprec,valCost,totIntang, revenu

plot1 <- 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()
```

create normalized variables `{r names(data)}` and scaled Questions: are the variable of the same order of magnitude?

```
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/50)  %>%
    mutate(logStock  = log10(StockAll)) %>%
    mutate(logTurnover  = log10(turnover)) %>%
    mutate(logBalance  = log10(balance)) %>%
    select(-totAssets,-totEquity, -noi,-personnel,-debts,-prod,
           -deprec,-revenues,-valCost,-totIntang, -StockAll, -turnover, -balance)%>%
   na.omit()

plot2 <- ggplot(stack(data), 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()
ggarrange(plot1,plot2)
```

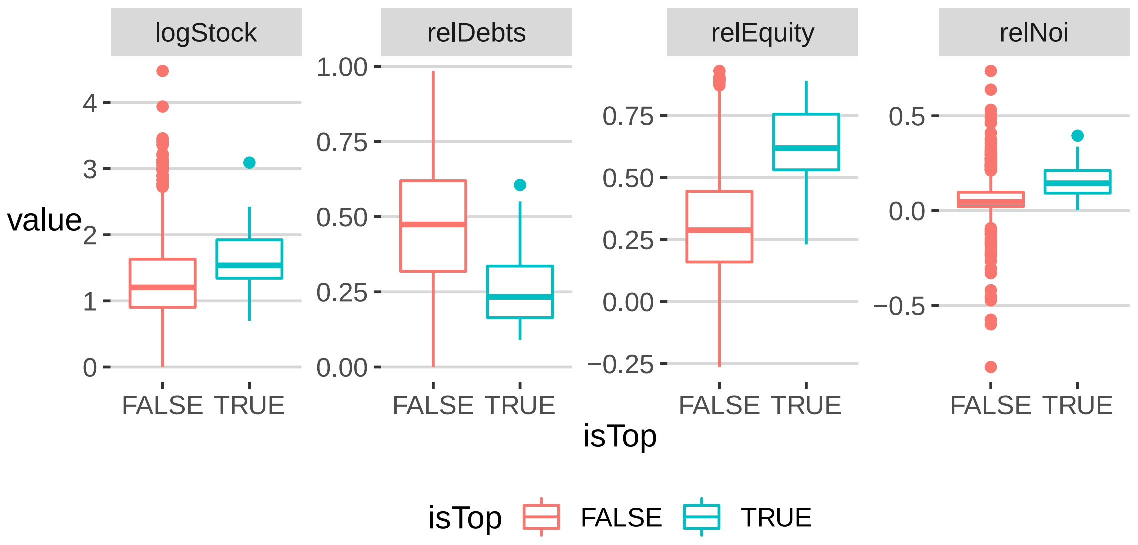




Questions: are the variables useful to predict the given label? are the distributions different acocrding to the label? Answet: bowplot by laber

```
data.to.plot <- data %>% select(relEquity, relNoi, logStock, relDebts,isTop)

plot <- data.to.plot %>% pivot_longer(cols=!isTop) %>% ggplot(aes(x=isTop, y=value, color = isTop)) + g
plot + facet_wrap(~name, ncol = 5,    scales="free")
```

```
## Warning: Removed 8 rows containing non-finite values (stat_boxplot).
```



Learning: Binary Classification Trees

The proposed solution is to apply “Binary Classification Trees” and check if the model can be tuned to achieve at least 85% performance

What are “Binary Classification Trees” [short intro to tree learning methods] We use “tree” that provides: - a function for doing the learning `tree()` - a function for doing the prediction, usually named `predict()`

The learning function in tree requires a dataframe and an indication of the dependency, using a data type, peculiar of R, known as **formula** as in the examples `a ~ b+c` (variable `a` depends on `b` and `c`) or `a ~ .` (variable `a` depends on all the other variables).

How to optimize the performance of “Binary Classification Trees”: hyperparameter minsplit

The following are 2 examples of trees (a) with a limited complexity, minsplit = … and (b) high complexity, minsplit = …

```
names(data)
```

```
##  [1] "yearsInBusiness" "is.startup"      "is.fem"          "isTop"
##  [5] "relInt"          "relEquity"       "relreve"         "relNoi"
##  [9] "relPers"         "relDebts"        "relDeprec"       "logStock"
## [13] "logTurnover"     "logBalance"
```

```
require(rpart)
```

```
## Caricamento del pacchetto richiesto: rpart
```

```
## Warning: il pacchetto 'rpart' è stato creato con R versione 4.1.2
```

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

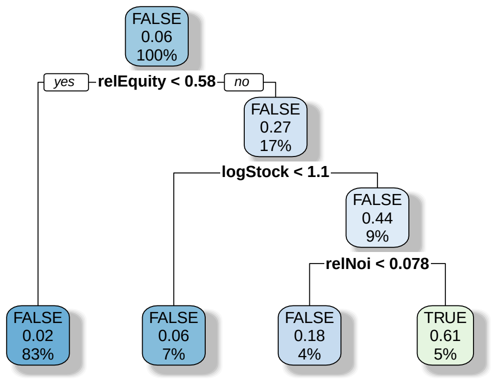
```
## Caricamento del pacchetto richiesto: rpart.plot
```

```
## Warning: il pacchetto 'rpart.plot' è stato creato con R versione 4.1.2
```

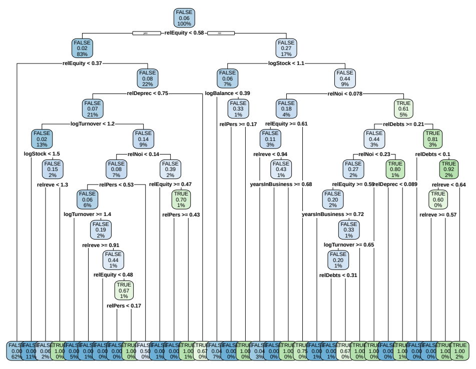
```
figures <- par(mfrow=c(1,2)) # more trees in the same figure

parms = rpart.control(split = "gini", minsplit = 95, cp = .01 )
simple.tree <- rpart(isTop~., data, method = "class", control=parms)
simple.tree %>% rpart.plot(  main="A simple classification tree", shadow.col = "gray")

parms = rpart.control(split = "gini",  minsplit = 5, cp=.01 )
complex.tree <- rpart(isTop~., data, method = "class",control=parms)
complex.tree %>% rpart.plot(  main="A complex classification tree")
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

A simple classification tree



A complex classification tree

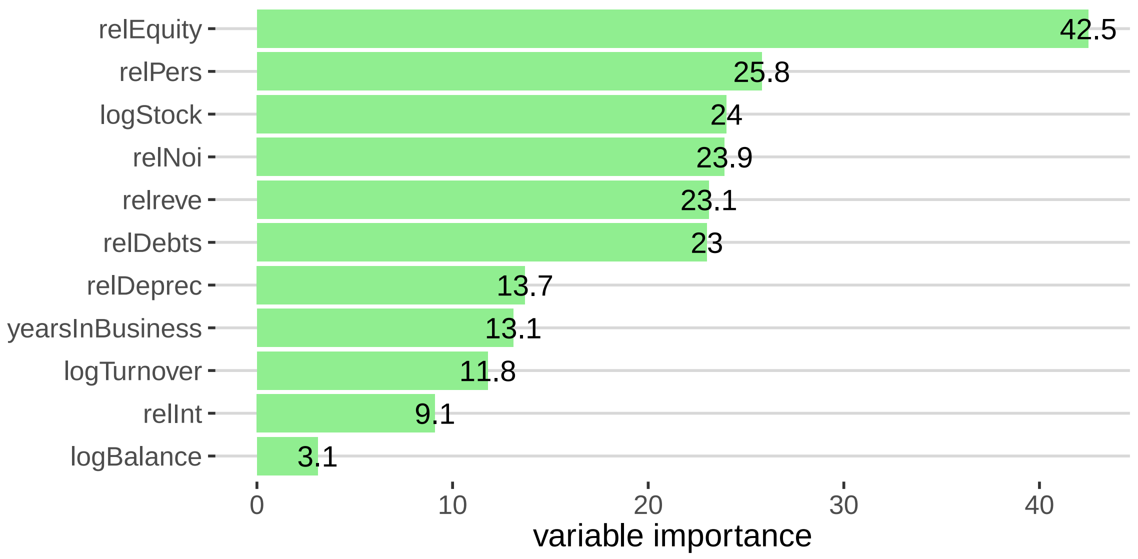


```
par(figures)
```

The models exibith different complexity, accuracy and explainability.

variable importance

```
tibble(features=names(complex.tree$variable.importance) , imp=complex.tree$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))
```



3. Experimental evaluation

Error rate of a single tree

[explain]

- error rate
- accuracy
- confusion matrix

Another way is to "compute" the **confusion matrix** and then obtaining the error from that. The confusion matrix shows the number misclassifications, class by class:

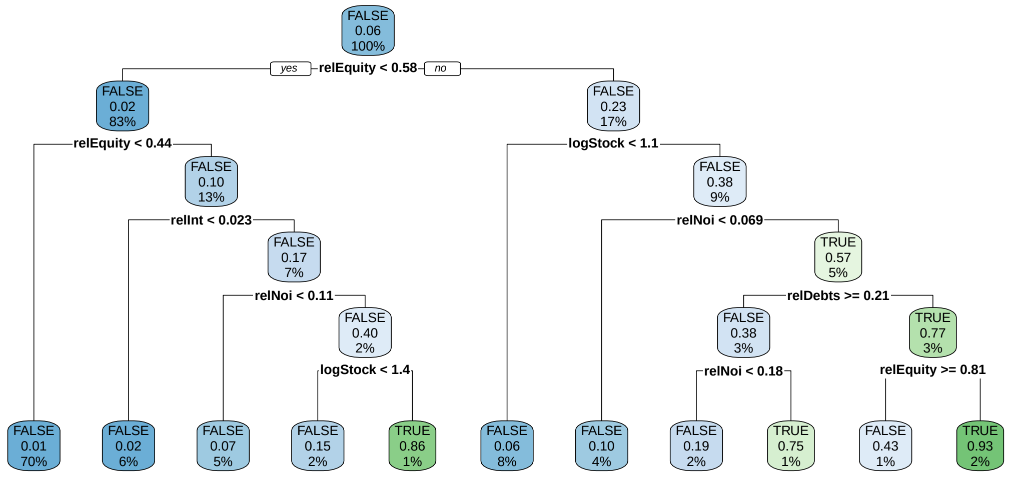
```r
computeConfusionMatrix = function(model, data) {
  predicted.y = predict(model, data, type="class")
  errors <- data %>%
    mutate(err = (categorical.y.name = predicted.y)) %>%
    filter(err == TRUE) %>% nrow()
  return(table(predicted.y, data$isTop))
}
```

we might build a function that computes the accuracy:

```r
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

```r
#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,]

parms = rpart.control(split = "gini",  minsplit = 20, cp=.01 )
tree <- rpart(isTop~.,
              data.learn,
              method = "class",control=parms)
rpart.plot(tree)
```



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

```
##
## predicted.y FALSE TRUE
##       FALSE   809   23
##       TRUE      4   26
```

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

```
##
## predicted.y FALSE TRUE
##       FALSE   192   12
##       TRUE      5    7
```

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

```
## [1] 0.96868
```

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

```
## [1] 0.9213
```

TODO: error rates are stocastic: training the model with the same parameters will get different results. So we need to perform the procedure several times, and return the average result.

Overfitting

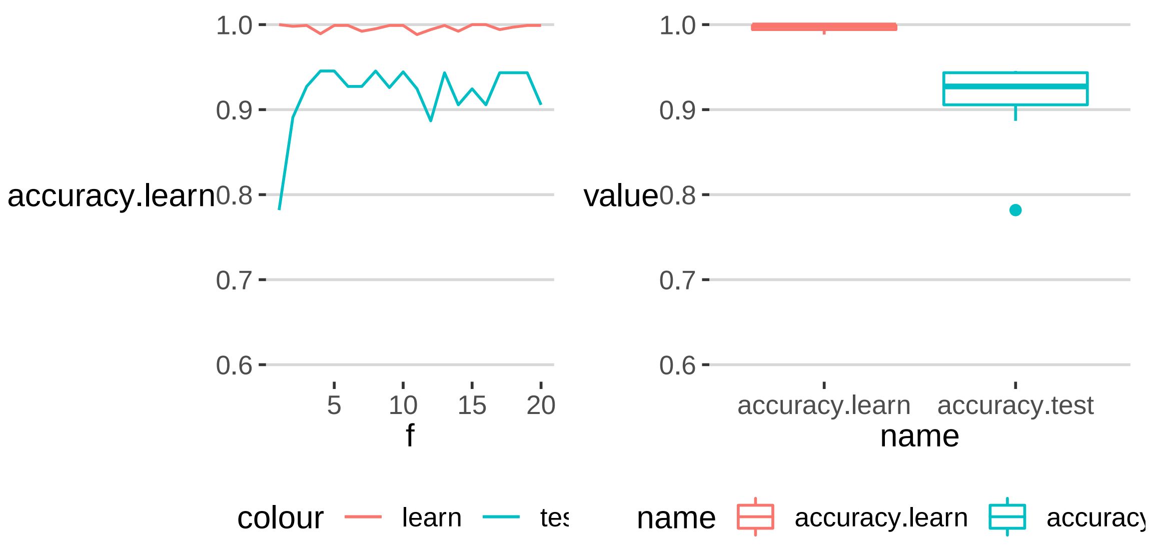
How will themodel perform on "unseen data"? can the learner generalize beyond available data? The classification error is an appropriate indicator. But the error depends on a random choice of learn and test data. We need to have a more stable value, so we introduce k-fold cross validation on the whole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

  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 ,
        accuracy.learn = computeAccuracy(model, data.other.folds ),
        accuracy.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=.001)
plot1 <- ggplot(data = folds.results)+geom_line( aes(x=f, y=accuracy.learn,color = "learn"))+
  geom_line( aes(x=f, y=accuracy.test, color = "test")) + ylim(0.6,1.0)


plot2 <- folds.results %>% pivot_longer(cols=c(accuracy.learn, accuracy.test)) %>%
  ggplot(aes(x=name, y=value, color = name))+geom_boxplot()+ ylim(0.6,1.0)

ggarrange(plot1, plot2)
```



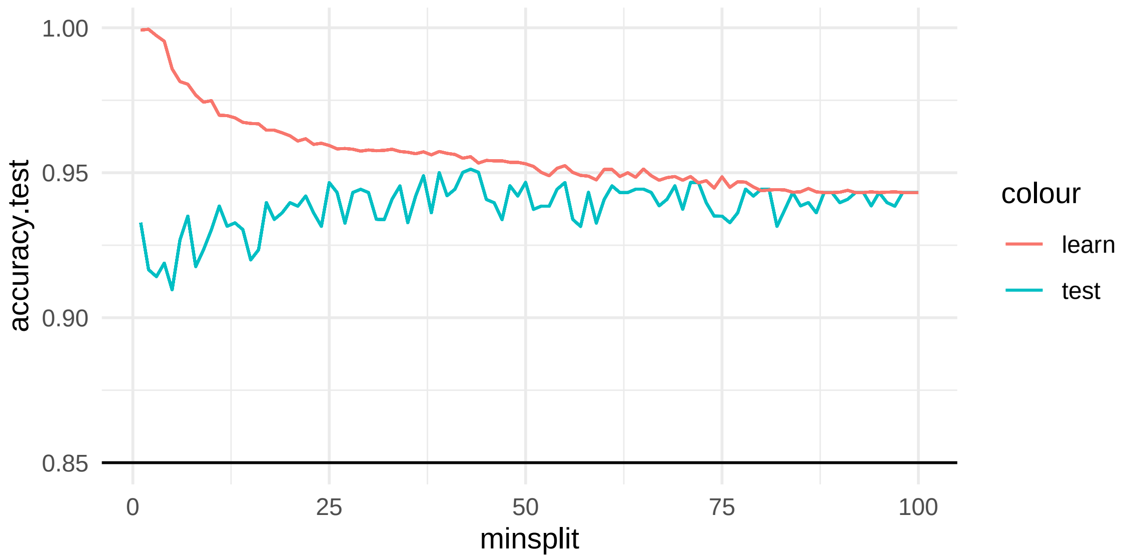
Tuning tree with k-fold cross validation

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.

Performance assessment (nested k-fold cross validation)

```
# a sample of tuning and plot / overfitting
#
#

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=.001 ))
}
tuning.results %>% group_by(minsplit)%>%
  summarise(accuracy.test=mean(accuracy.test), accuracy.learn=mean(accuracy.learn), f=f)%>%
  group_by(f)%>%
  ggplot() + theme_minimal()+
  geom_line(aes(x=minsplit, y = accuracy.test, group = f, colour = 'test')) +
  geom_line(aes(x=minsplit, y = accuracy.learn,group = f, colour = 'learn'))+
  geom_hline(yintercept = .85)
```

```
## `summarise()` has grouped output by 'minsplit'. You can override using the `.groups` argument.
```



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

  testing.results <- tibble(i=NA , acc.test=NA, acc.learn=NA,number.of.nodes=NA, minsplit=NA) %>% head(
  plots   <- tibble()
  maxms=max(minsplit_candidates)

  #initialize empty tibble
  tuning.results = tibble(f = NA, accuracy.learn = NA,
                          accuracy.test = NA,  n = NA, minsplit = NA) %>% head(0)

# loop to calculate errors for all minsplit candidates
  for (mspl in minsplit_candidates){
    tuning.results <- tuning.results %>%
      bind_rows(compute.Kfold.accuracy(data.learn,  k_folds=k_folds_tuning, minsplit=mspl, cp=.001 ))
  }

  best = tuning.results %>%
      group_by(minsplit) %>%
      summarise(accuracy.learn = mean(accuracy.learn),
                accuracy.test = mean(accuracy.test), n = n) %>%
      arrange(desc(accuracy.test)) %>% head(1)
  mspl.star = best$minsplit[1]
  print(mspl.star)

  #tune model with optimal value
  opt.model <- rpart(  isTop~.,data.learn, method = "class", minsplit = mspl.star, cp = .001)
  testing.results <- compute.Kfold.accuracy(data, k_folds=k_folds_testing, minsplit=mspl.star,  cp = .0

  return(testing.results)
}
```

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

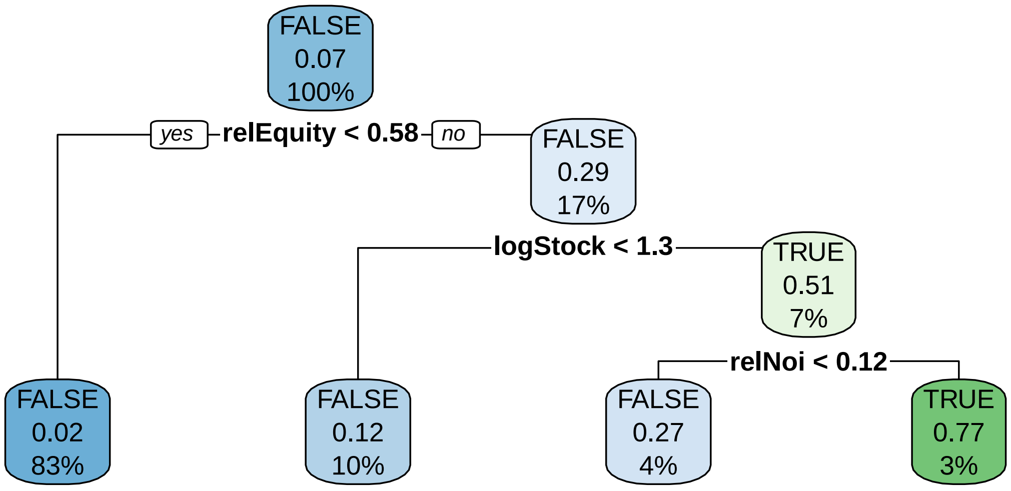
```r
data.test <- data %>% slice(-indexes.learning)

# define the range for complexity parameter
nn<- 100#nrow(data)/4 # reduce to save on computetion effort
minsplit_candidates = seq(1,nn,round(nn/100,0))

#define k-fold validation (computational cost vs quality)
k_folds_tuning=5
k_folds_testing=5

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

```
## ‘summarise()‘ has grouped output by ’minsplit’. You can override using the ‘.groups‘ argument.

## [1] 38
```

```r
optimal.params <- optimal.params %>% arrange(desc(accuracy.test, minsplit))
optimal.minsplit <- optimal.params %>% head(1) %>% select(minsplit) %>% pull
optimal.acc <-      optimal.params %>% head(1) %>% select(accuracy.test) %>% pull
```

final result

```r
final.tree <- rpart(  isTop~.,data.learn, method = "class",
                      minsplit = optimal.minsplit, cp = .001)
rpart.plot(final.tree)
```



```r
print( computeConfusionMatrix(final.tree, data.learn) )
```

```
##
## predicted.y FALSE TRUE
##       FALSE   798   34
##       TRUE      7   23
```

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

```
##
## predicted.y FALSE TRUE
##       FALSE   204   10
##       TRUE      1    1
```

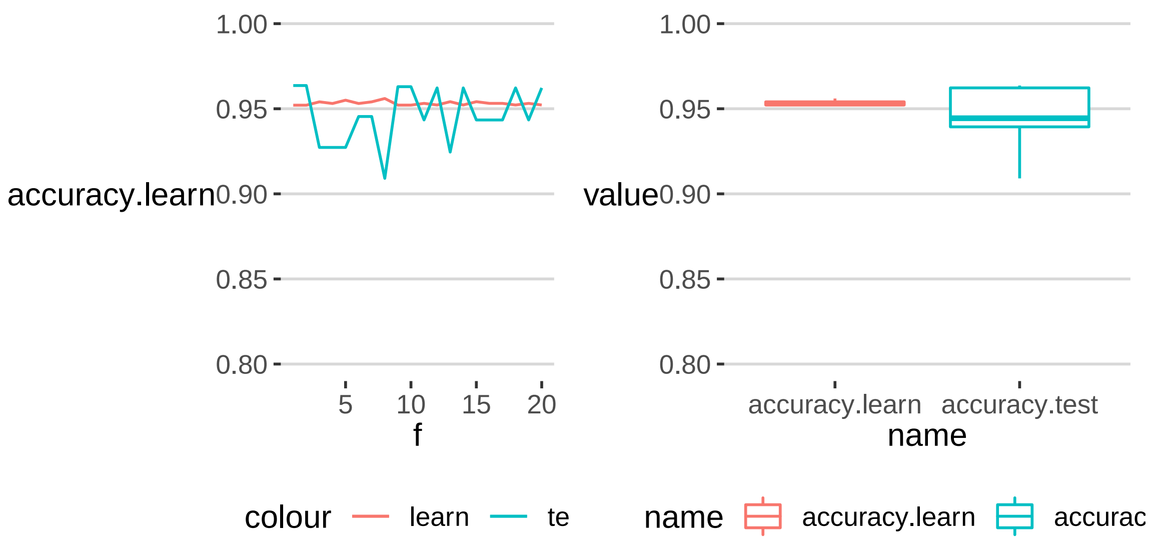
```
print( computeAccuracy(final.tree, data.learn) )
```

```
## [1] 0.95244
```

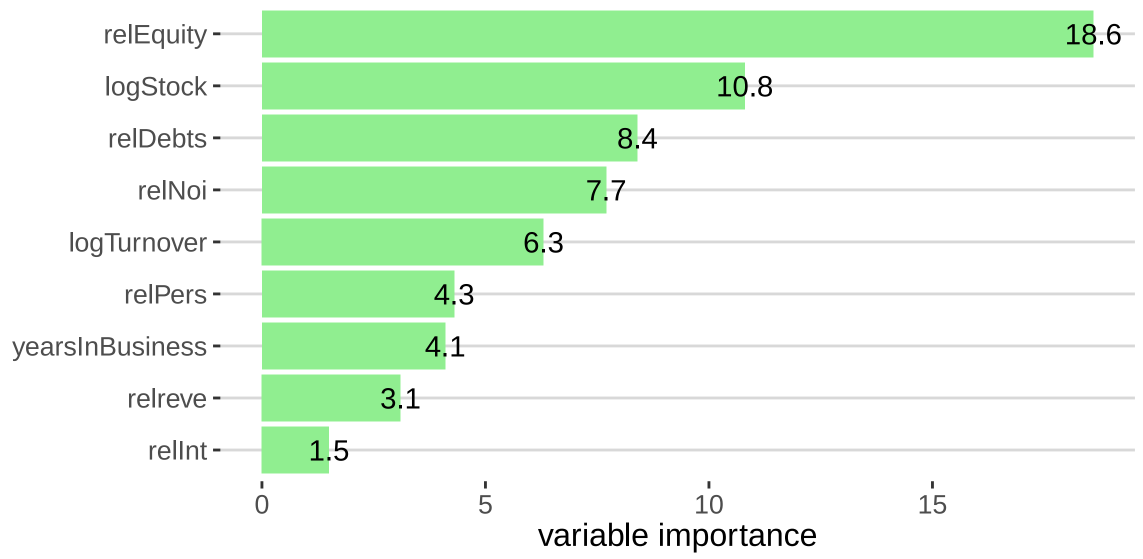
```
print( computeAccuracy(final.tree, data.test)  )
```

```
## [1] 0.94907
```

```
folds.results = compute.Kfold.accuracy(data,  k_folds=20, minsplit=optimal.minsplit, cp=.001)
plot1 <- ggplot(data = folds.results)+geom_line( aes(x=f, y=accuracy.learn,color = "learn"))+
  geom_line( aes(x=f, y=accuracy.test, color = "test")) + ylim(0.8,1.0)


plot2 <- folds.results %>% pivot_longer(cols=c(accuracy.learn, accuracy.test)) %>%
  ggplot(aes(x=name, y=value, color = name))+geom_boxplot()+ ylim(0.8,1.0)

ggarrange(plot1, plot2)
```



```
tibble(features=names(final.tree$variable.importance) , imp=final.tree$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. Concluding remarks and future work

Feasibility

the paper demonstrates demonstrate feasibility by training a simple model with accuracy > 85% Results are specific for the given subset (2300 companies in a range of sectors, focus on year 2019)

- the feasibility is demonstrated by 3 examples

1) a simple, tree only 5 nodes, few variables, 85% explainability
2) including staff turnover, balance and stock definitely improve the quality of predictions
3) robust to label and sector (try a subsector and a larger label)

remark: better performance ==> tree should be learned for specific subsets and specific definitions of label isTop.

All samples respect the constraint: computation time is of the order of minutes, athe model is interpretable and explainable.

Future work

Preliminary results suggest further investigations:

- explore in depth under different conditions: itTop defined as rating >=7 or >=9
- performance with other supevised models: multiclass decision trees, random forrest, svm, naive bayes, knn,
- unsupervised: PCA on all variables (promising, since only 2 variables give good resulta); 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