Tree-based Models

Microeconometrics and Application: Report 5

Simrit Rattan (27720199656249)
Elena Riccarda Ziege (27720199656247)

December 22, 2019

1 Statistical Background

The aim of this work is to predict if a person is unemployed or not, which we are doing with three different approaches. First, we start with an simple model, the logistic regression and then we will move to more complex models, a classification tree and boosting. We will compare the performance of each model on a test data set, by computing the classification error, AUC and plotting the ROC curves.

The AUC is the area under the ROC (receiver operating characteristic) curve, which plots the true positive rate on the y-axis and the false positive rate on the x-axis (cf. Fawcett, 2006). The true positive rate, also called sensitivity, is the percentage of correctly identified unemployed cases and the specificity is the percentage of correctly identified cases that have never been unemployed (cf. James et al., 2013). The false positive rate is the percentage of incorrectly specified non-unemployed cases (cf. Fawcett, 2006).

1.1 Logistic Regression

Our outcome variable, unemployment, is a dichotomous variable, which is why as a simple approach, the logistic regression is briefly introduced. The logit model looks at the log odds of the outcome, where the odds are the ratio of the probability of our outcome taking up the value 1 and the probability of our outcome taking up the value 0. The log odds of the outcome is modelled as a linear combination of the predictors. We are interested in investigating how the log odds is changed by the independent variables. To estimate the parameters the Maximum Likelihood approach is used. The logit model and odds are depicted in equation 1, whereas the probability of unemployment in the logistic regression is modelled in equation 2:

$$LogitModel : ln(\frac{\pi_i}{1-\pi_i}) = \sum_{k=0}^{K} x_{ik}\beta_k \qquad\qquad Odds : \frac{\pi_i}{1-\pi_i} = e^{\sum_{k=0}^{K} x_{ik}\beta_k} \tag{1}$$

$$\pi_i = Pr(Y_i = 1) = \sigma(x_i'\beta) = \frac{exp(\sum_{k=0}^{K} x_{ik}\beta_k)}{1 + exp(\sum_{k=0}^{K} x_{ik}\beta_k)} = \frac{1}{exp(-\sum_{k=0}^{K} x_{ik}\beta_k)} \tag{2}$$

1.2 Classification Tree

There are two forms of decision trees: Regression trees and classification trees (cf. James et al., 2013). We are using a classification tree because our dependent variable is discrete (cf. Varian, 2014). To construct a classifcation tree one uses binary splitting, which means making successive binary splits on the predictor variable (cf. James et al., 2013). Classification trees divide a set of values of a predictor variable into several regions, so that the observations within each region are as homogeneous as possible, which is also called node purity (cf. James et al., 2013).

The optimal cutpoints between the regions are determined by a measure of accuracy. Regression trees use the residual sum of squares as a measure for making binary splits while classification trees cannot use it.

Instead there are three other measures that can be used: The classification error rate, the Gini index and the cross-entropy. The classification error rate measures the share of observations in each region that are not assigned to the most commonly assigned class. It is calculated as follows:

$$E = 1 - \max_k(\hat{p}_{mk}) \tag{3}$$

where $\widehat{p}_{mk}$ is the share of observations that are from the kth class in the mth region. The Gini-index measures the total variance across the K different classes and is calculated as follows:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}). \tag{4}$$

G is small if all $\hat{p}_{mk}$'s are either close to 0 or 1 and a small value means that a node mainly includes observations from one class. This is why the Gini index is also called a measure of node purity. Node purity is important because it gives us more certainty that all the observations in one node are really in the class assigned to all of them based on which class occurs the most. The third measure, the cross entropy is very similar numerically to the Gini index and is also a measure of node purity. It is calculated the following way:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk}). \tag{5}$$

$-\hat{p}_{mk}log(\hat{p}_{mk})$ is positive or equal to 0 as $\hat{p}_{mk}$ is a share and lies between 0 and 1. This is why D will be close to 0 if all $\hat{p}_{mk}$'s are either close to 0 or close to 1. The Gini index and the cross-entropy are mostly used for building a classification tree to evaluate a particular split since they both measure node purity. The classification error rate on the other hand is used when the focus is the prediction accuracy. All three methods can be used to prune a tree. (cf. James et al., 2013)

For all observations that are classified into one region, the same class is predicted (cf. Mullainathan and Spiess, 2017). When the terminal node, also called leaf, is reached, a prediction for all observations in that region is made based on the class that occurs most often in the region (cf. Mullainathan and Spiess, 2017).

Classification trees are very prone to overfitting because of being too complex, which is why most trees are pruned after the construction. A pruned tree will have a smaller variance even though the bias will increase a little. The idea is to find a pruned tree which has the lowest test error rate. This can be calculated using cross-validation, which we will be doing later as well. The idea is to minimize the number of terminal nodes while still minimizing our measure, which could be the classification error, the Gini index or the cross-entropy. (cf. James et al., 2013)

1.3 Boosting

Boosting is an alternative regularization method, where the estimation problems are described in terms of a loss function. Due to this attribute, boosting can also be applied to generalized linear models for non-Gaussian responses. (cf. Fahrmeier et al., 2013). Boosting and Bagging are similar in the sense, as both can be applied to many statistical methods for regression and classification. Boosting also works in a similar way as bagging, although in boosting trees are grown sequentially, meaning that each tree is grown by using information from the trees already grown. Unlike bagging, boosting does not use bootstrap sampling. In the latter, each tree is fit on an altered version of the original data set. The boosting approach works as a “slow-learning” method. (cf. James et al., 2013)

Boosting for binary classification is done here by one of the most popular boosting algorithms, called AdaBoost. Step by step a weak classification algorithm, which only has a slightly smaller error than a random guess, is applied to weighted versions of the data, which then results in a sequence of weak classifiers ($G_m(x)$).

After calculating $G_m(x)$, we firstly compute the error, $err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$. Based on that we calculate the weights for the classifier ($\alpha_m = log((1 - err_m)/err_m)$). Classifiers which are more accurate get a higher weight. But those weights ($\alpha_1, \alpha_2, ..., \alpha_m$) differ from the weights ($w_1, w_2, ..., w_N$) applied to each of the training observations. The inital weight is set to $w_i = 1/N$, and for each further step an individual modification of the observation weights takes place, by setting $w_i \leftarrow w_i exp[\alpha_m I(y_i \neq G_m(xi))],\ for\ i = 1, 2, ..., N$. Then the classification algorithm is again applied to the weighted observations. Every time observations have been misclassfied by the classfier before, their weights are increased for the following step. The final prediction ($G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$) then, is a weigthed sum of the individual weak classifiers. With this approach, successive classifiers have to focus their attention on misclassified observations. (cf. Hastie et al., 2001)

AdaBoost has the advantage of improving the performance of weak classfiers. The loss function of the AdaBoost algorithm is very similar to the negative binomial log-likelihood ($\rho(\eta)) = \sum_{i=1}^{n} exp(-\gamma_i \eta_i)$) (cf. Fahrmeier et al., 2013). Moreover, boosting can be seen as a way to fit an adaptive additive model, with the form of the model being equivalent to a linear basis function, with the only difference that the basis functions are now base learners. (cf. Hastie et al., 2001)

2 Empirical Implementation

In this report we want to find out what factors influence the probaility that a person has ever been unemployed and looked for a job for more than three months (uemp3m). The aim of this report is to predict which people have ever been unemployed for more than three months using a test data set and to then measure the prediction accuracy of different models.

We are using a data set called European Social Survey, conducted across Europe since 2001. It is a cross-national survey which surveys all persons aged 15 and over living in private households every two years, regardless of their nationality, citizenship, language or legal status. It measures the attitudes, beliefs and behavioural patterns of the population. The data we are analyzing is from the ninth ESS wave conducted in 2018, which means that we are dealing with cross-sectional data. This wave includes data from thirty nations in Europe, including EU and non-EU countries. (cf. European Social Survey, 2019)

```r
library(foreign)
data <- read.dta("ESS9e01_1.dta")
```

Therefore, we need several predictor variables, which can be seen in the following table:

Variable	Definiton
agea	Age of respondent, calculated
atncrse	Improve knowledge/skills: course/ lecture/ conference, last 12 months
brncntr	Born in country
cntry	Country
dscrgrp	Member of a group discriminated against in this country
dvrcdeva	Ever been divorced/had civil union dissolved
eduyrs	Years of full-time education completed
emplrel	Employment relation
nbthcld	Number of children ever given birth to/ fathered
gincdif	Government should reduce differences in income levels
gndr	Gender
health	Subjective general health
hhmmb	Number of people living regularly as member of household
hincsrca	Main source of household income
hswrk	Doing last 7 days: housework, looking after children, others
iagpnt	Become mother/ father, ideal age
impfree	Important to make own decisions and be free
imprich	Important to be rich, have money and expensive things
impsafe	Important to live in secure and safe surroundings
ipcrtiv	Important to think new ideas and being creative
ipfrule	Important to do what is told and follow rules
ipsuces	Important to be successful and that people recognise achievements
lvpntyr	Year first left parents for living separately for 2 months or more
netusoft	Internet use, how often
ppltrst	Most people can be trusted or you can't be too careful
sclact	Take part in social activities compared to others of same age
sclmeet	How often socially meet with friends, relatives or colleagues
stfedu	State of education in country nowadays
stfhlth	State of health services in country nowadays
stflife	How satisfied with life as a whole
wkhtot	Total hours normally worked per week in main job overtime included
wrkac6m	Paid work in another country, period more than 6 months last 10 years

2.1 Preparation of the data set

We need to prepare the data set before we can use it. We need to make cntry a factor varibale. Additionally, we construct a variable for the age at the birth of the first child `age_birth`.

```
data$cntry <- as.factor(data$cntry)

# age at birth of first child
data$age_birth <- ifelse((data$fcldbrn-data$yrbrn)>13, data$fcldbrn-data$yrbrn, NA)
```

Furthermore, we omit the observations which have missing values in our depentend variable `uemp3m` and drop the levels for those levels, who do not have any observation. We also relevel our dependent variable, such that 1 is defined as unemployed and 0 as employed.

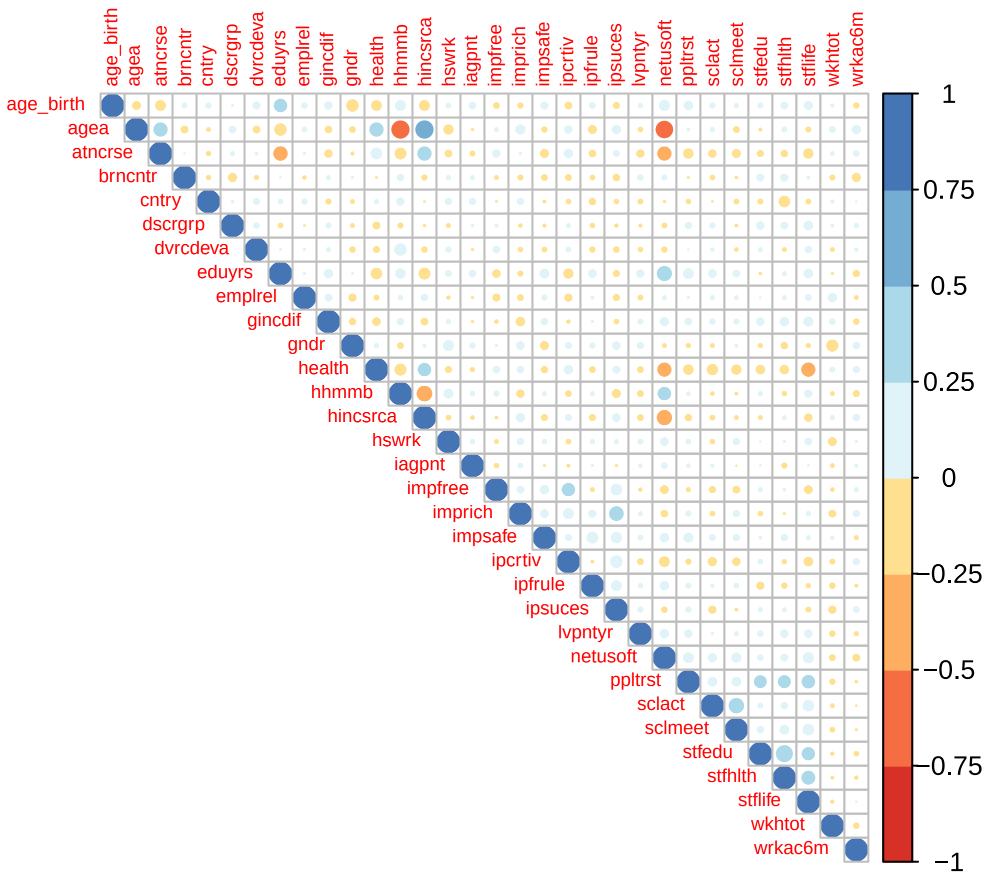
```
attach(data)
dataset <- data.frame(uemp3m,agea,atncrse,brncntr,cntry,dscrgrp,dvrcdeva,eduyrs,emplrel,
                      gincdif,gndr,health,hhmmb,hincsrca,hswrk,iagpnt,impfree,imprich,
                       impsafe,ipcrtiv,ipfrule,ipsuces,lvpntyr,netusoft,ppltrst,sclact,
                       sclmeet,stfedu,stfhlth,stflife,wkhtot,wrkac6m,age_birth)
detach(data)
summary(dataset$uemp3m)
```

```
##        Yes         No    Refusal Don't know  No answer       NA's
##       9785      25996          0          0          0        234
```

```
library(dplyr)
dataset <- dataset %>% filter(!is.na(uemp3m))
dataset$uemp3m <- droplevels(dataset$uemp3m, exclude =c("Refusal", "Don't know",
                                                        "No answer"))
dataset$uemp3m <- factor(dataset$uemp3m, levels = c("No", "Yes"))
```

As we have a variety of independent variables, we decide to plot a correlation matrix, to see if there is any multicollinearity between the variables.

```
library(corrplot)
library(RColorBrewer)
dataset[] <- lapply(dataset, function(x) if(is.factor(x)) factor(x) else x)
dataset$cntry <- as.factor(dataset$cntry)
X <- dataset[,-1]
X[] <- lapply(X, function(x) as.numeric(x))
M <- cor(X, use = "complete.obs")
corrplot(M, type="upper", order="alphabet", tl.cex = 0.6,
         col=brewer.pal(n=8, name="RdYlBu"))
```



As we can see in the correlation plot, only few variables are highly correlated with each other. We can see that the correlation plot depicts a high positive relation of the age (`agea`) with the main source of household income (`hincsrca`), a high negative relation with the number of household members (`hhmmb`) and with how much a person uses the internet (`netusoft`). The main source of household income influences how fast a person tries to find an employment and the amount of time she spends using the internet is a good indicator of how well a person can work with a computer which is very important for the modern labour market. We decided to not include the variable `agea`, as the other three variables are more important to explain unemployment, and people of every age could be unemployed. All other variables can be used for the analysis.

```r
dataset$agea <- NULL
```

In the next step we divide our dataset into training and test data, where we fit our models on the training set and evaluate their performance on the test data. Moreover, we specify a validation set, to choose the parameters in the boosting approach. At the end the training set includes 50 %, the validation set 20 % and the test data 30 % of the data.

```r
set.seed(9999)
idx.train <- caret::createDataPartition(y = dataset$uemp3m, p = 0.6, list = FALSE)
train1 <- dataset[idx.train, ]
test <-  dataset[-idx.train, ]
idx.valid <- caret::createDataPartition(y = train1$uemp3m, p = 0.7, list = FALSE)
train <- train1[idx.valid,]
valid <- train1[-idx.valid,]
```

Due to the fact, that the logistic regression cannot work with missing values, we decide to exclude all observations which have missing values in any variable. Thus, we are doing a complete case analysis.

```r
dataset_nonmiss <- na.omit(dataset)
dataset_nonmiss[] <- lapply(dataset_nonmiss,
                            function(x) if(is.factor(x)) factor(x) else x)

set.seed(9999)
idx.train <- caret::createDataPartition(y = dataset_nonmiss$uemp3m, p = 0.5, list = FALSE)
train_nonmiss <- dataset_nonmiss[idx.train, ]
test_nonmiss <-  dataset_nonmiss[-idx.train, ]
```

2.2 Logistic Regression

The first method we use to predict unemployment is the logistic regression, for which we use the `glm()` function. When dividing our dataset into test and train, it is possible that one or more variables do not occur with all their levels. This can be a problem when predicting. For the variable `hincsrca`, not all levels are present in the test data, which is why whe have to modify the variable.

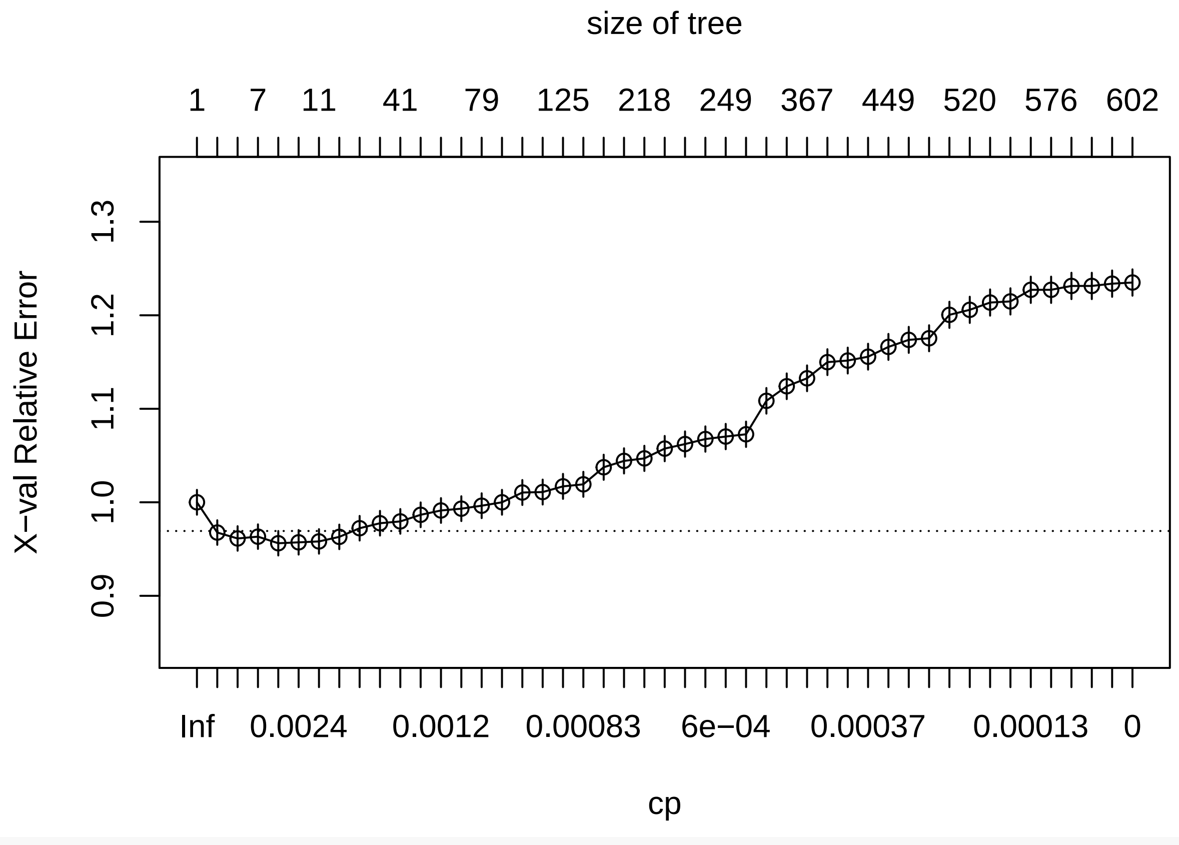
```r
logit <- glm(uemp3m ~ atncrse+brncntr+cntry+dscrgrp+dvrcdeva
             +eduyrs+emplrel+gincdif+gndr+health+hhmmb+hincsrca
             +hswrk+iagpnt+impfree+imprich+impsafe+ipcrtiv+ipfrule
             +ipsuces+lvpntyr+netusoft+ppltrst+sclact
             +sclmeet+stfedu+stfhlth+stflife+wkhtot+wrkac6m+age_birth,
             data = train_nonmiss, family = binomial(link = logit))

logit$xlevels[["hincsrca"]] <- union(logit$xlevels[["hincsrca"]], levels(test$hincsrca))
```

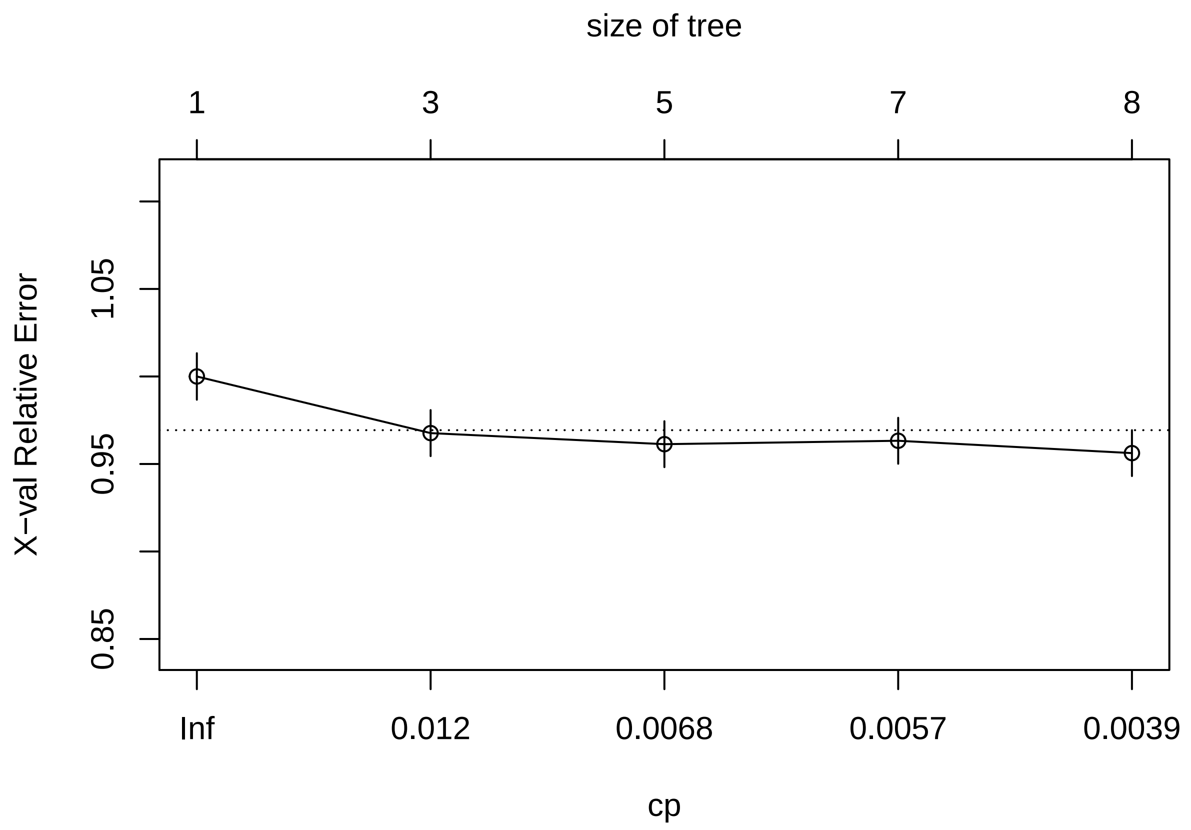
2.3 Classification Tree

At first, we construct a large classification tree without pruning. When the complexity parameter (cp) is set to zero, a very large tree is grown. In the CP-Plot one can see, that at first the error decreases with complexity, but after some time it increases again. Due to this fact, we decide to prune our classification tree. Pruning occurs until we cannot reduce the cross validation error anymore, which is in the 7th split. This can also be seen in the CP-plot, where the error reaches its min, just slightly below 1 with a complexity parameter of about 0.0026. In general, the smaller the complexity parameter, the higher the number of splits. In the `fancyRpartPlot` one can see the different splits made and that the pruned three is still quite complex.

```r
library(rpart)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
tree_large <- rpart(uemp3m ~ atncrse+brncntr+cntry+dscrgrp
             +dvrcdeva+eduyrs+emplrel+gincdif+gndr+health+hhmmb
             +hincsrca+hswrk+iagpnt+impfree+imprich+impsafe+ipcrtiv+ipfrule
             +ipsuces+lvpntyr+netusoft+ppltrst+sclact
             +sclmeet+stfedu+stfhlth+stflife+wkhtot+wrkac6m+age_birth,
             data = train, control=rpart.control(cp=0))
plotcp(tree_large)
```



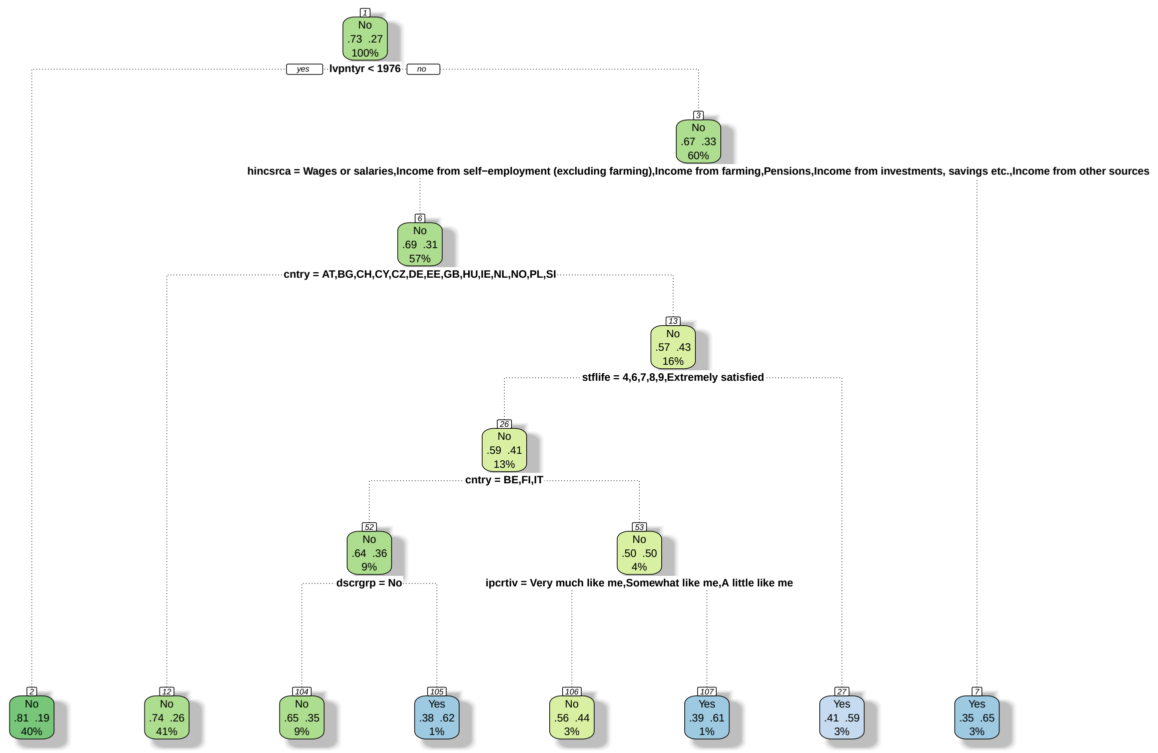
```r
tree <- prune(tree_large,cp=tree_large$cptable[which.min(tree_large$cptable[,"xerror"]),
                                                "CP"])
plotcp(tree)
```



```r
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = uemp3m ~ atncrse + brncntr + cntry + dscrgrp +
##     dvrcdeva + eduyrs + emplrel + gincdif + gndr + health + hhmmb +
##     hincsrca + hswrk + iagpnt + impfree + imprich + impsafe +
##     ipcrtiv + ipfrule + ipsuces + lvpntyr + netusoft + ppltrst +
##     sclact + sclmeet + stfedu + stfhlth + stflife + wkhtot +
##     wrkac6m + age_birth, data = train, control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] cntry    dscrgrp  hincsrca ipcrtiv  lvpntyr  stflife
##
## Root node error: 4110/15029 = 0.27347
##
## n= 15029
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0173966      0   1.00000 1.00000 0.013296
## 2 0.0080292      2   0.96521 0.96764 0.013158
## 3 0.0057178      4   0.94915 0.96131 0.013130
## 4 0.0055961      6   0.93771 0.96326 0.013139
## 5 0.0026764      7   0.93212 0.95620 0.013108
```

```r
fancyRpartPlot(tree, caption="", palettes="YlGn")
```


1
No
.73 .27
100%
yes
lvpntyr < 1976
no
3
No
.67 .33
60%
hincsrca = Wages or salaries,Income from self−employment (excluding farming),Income from farming,Pensions,Income from investments, savings etc.,Income from other sources
6
No
.69 .31
57%
cntry = AT,BG,CH,CY,CZ,DE,EE,GB,HU,IE,NL,NO,PL,SI
13
No
.57 .43
16%
stflife = 4,6,7,8,9,Extremely satisfied
26
No
.59 .41
13%
cntry = BE,FI,IT
52
No
.64 .36
9%
dscrgrp = No
53
No
.50 .50
4%
ipcrtiv = Very much like me,Somewhat like me,A little like me
2
No
.81 .19
40%
12
No
.74 .26
41%
104
No
.65 .35
9%
105
Yes
.38 .62
1%
106
No
.56 .44
3%
107
Yes
.39 .61
1%
27
Yes
.41 .59
3%
7
Yes
.35 .65
3%

2.4 Boosting

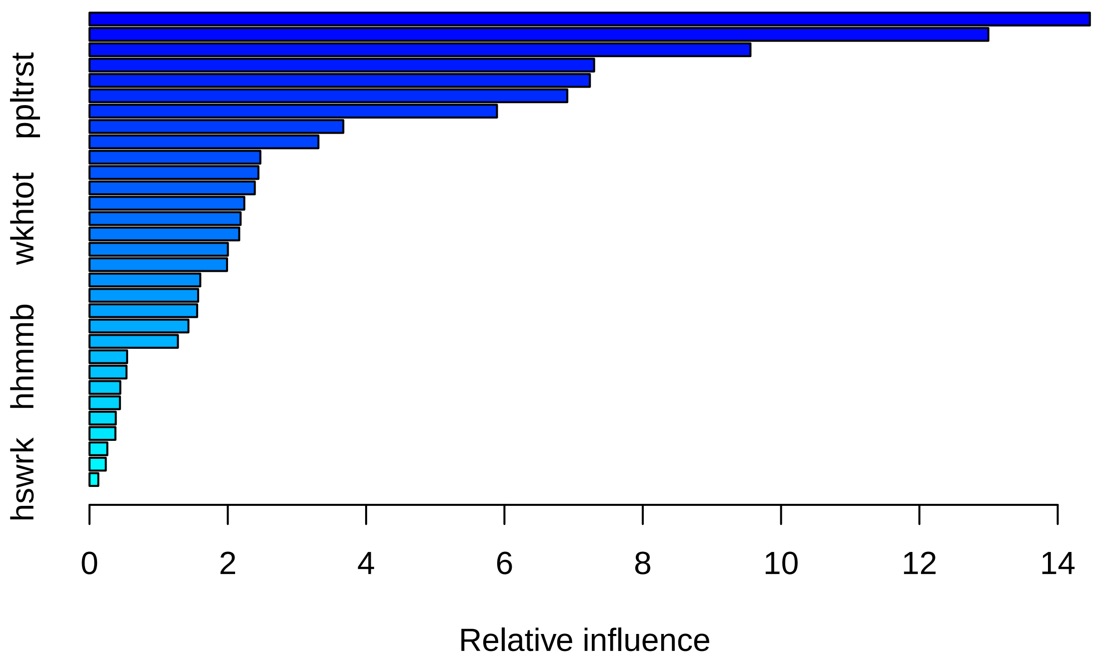
For the boosting approach, we decided to use a validation set, to evaluate when boosting performs the best. This depends on the depth of the trees, the number of the trees and the shrinkage parameter. After trying out different values for these parameters, we found out that the depth of the trees make the most difference, when evaluating the out of sample performance on the validation set. Moreover, the lower we set the tuning parameter (`shrinkage`), the higher the out of sample error and when the tuning parameter is set to 1, the error is very large too. So a good value in our example would be 0.1. The number of trees only have a minimal effect on the error, but we examined, that a larger number of trees will result in a smaller error. At the end we choose the boosting approach, with 5000 trees, a depth of trees of 30 and a tuning parameter of 0.1, resulting in an out of sample error of 0.2624. In the summary of the boosting method, we can also see the relative influence of each variable in a graph, which shows that the variables "netusoft" (How often is internet used) and "cntry" (Country) and "health" (Health) have the biggest influence when predicting unemployment. This is accurate as different countries have different economic situations The frequency of a person using the internet is related to how good someone can work with the internet and maybe computers in general, which in today's world is very essential for many jobs. Health can influence a person's ability to work and thus should show an effect in predicting whether a person is unemployed or not. Also a person might feel that the state of health service in his or her country is not very good, when someone is having ongoing health problems with no or only little progress in curing and thus might not be able to work. The state of education in one's country (`stfedu`) is also very important in predicting unemployment, as it can influence how much education one can obtain, and this is related to the employment status. Moreover, we can't really explain why improving knowledge and skills (`atncrse`) has nearly no effect on unemplyoment.

```
library(gbm)
library(kableExtra)
data_boost <- transform(train,uemp3m=as.numeric(uemp3m)-1)
boostingFit1 <- gbm(uemp3m ~ atncrse+brncntr+cntry+dscrgrp+dvrcdeva
             +eduyrs+emplrel+gincdif+gndr+health+hhmmb+hincsrca
             +hswrk+iagpnt+impfree+imprich+impsafe+ipcrtiv+ipfrule
             +ipsuces+lvpntyr+netusoft+ppltrst+sclact
             +sclmeet+stfedu+stfhlth+stflife+wkhtot+wrkac6m+age_birth,
             data = data_boost, distribution = "adaboost",n.trees = 500,
             interaction.depth = 5, shrinkage = 0.01)

boostingFit2 <- gbm(uemp3m ~ atncrse+brncntr+cntry+dscrgrp+dvrcdeva
             +eduyrs+emplrel+gincdif+gndr+health+hhmmb+hincsrca
             +hswrk+iagpnt+impfree+imprich+impsafe+ipcrtiv+ipfrule
             +ipsuces+lvpntyr+netusoft+ppltrst+sclact
             +sclmeet+stfedu+stfhlth+stflife+wkhtot+wrkac6m+age_birth,
             data = data_boost, distribution = "adaboost",n.trees = 1000,
             interaction.depth = 2, shrinkage = 0.001)

boostingFit <- gbm(uemp3m ~ atncrse+brncntr+cntry+dscrgrp+dvrcdeva
             +eduyrs+emplrel+gincdif+gndr+health+hhmmb+hincsrca
             +hswrk+iagpnt+impfree+imprich+impsafe+ipcrtiv+ipfrule
             +ipsuces+lvpntyr+netusoft+ppltrst+sclact
             +sclmeet+stfedu+stfhlth+stflife+wkhtot+wrkac6m+age_birth,
             data = data_boost, distribution = "adaboost",n.trees = 5000,
             interaction.depth = 30, shrinkage = 0.1)

summary(boostingFit)
```



```
##                  var    rel.inf
## netusoft   netusoft 14.4652316
## cntry         cntry 12.9974263
## health       health  9.5573947
## stfedu       stfedu  7.2968019
## stfhlth     stfhlth  7.2359432
## ppltrst     ppltrst  6.9093186
## stflife     stflife  5.8924280
## lvpntyr     lvpntyr  3.6700406
## sclmeet     sclmeet  3.3101965
## ipsuces     ipsuces  2.4708132
## hincsrca   hincsrca  2.4426656
## ipfrule     ipfrule  2.3907979
## ipcrtiv     ipcrtiv  2.2386498
## wkhtot       wkhtot  2.1848353
## age_birth age_birth  2.1652067
## impsafe     impsafe  2.0012323
## imprich     imprich  1.9889049
## sclact       sclact  1.6019370
## impfree     impfree  1.5713269
## eduyrs       eduyrs  1.5571239
## iagpnt       iagpnt  1.4309293
## gincdif     gincdif  1.2784304
## hhmmb         hhmmb  0.5443257
## brncntr     brncntr  0.5348866
## dvrcdeva   dvrcdeva  0.4454694
## dscrgrp     dscrgrp  0.4404900
## emplrel     emplrel  0.3802117
## wrkac6m     wrkac6m  0.3751195
## gndr           gndr  0.2579636
## atncrse     atncrse  0.2360753
## hswrk         hswrk  0.1278236
```

```
tau <- length(which(dataset$uemp3m=="Yes"))/length(dataset[,1])
transform_valid <- transform(valid,uemp3m=as.numeric(uemp3m)-1)
ytrue_boost <- transform_valid$uemp3m

yhat.boost.valid1 <- predict(boostingFit1,valid,n.trees=500, type ="response")
yhat.boost.class.valid1 <- as.numeric(yhat.boost.valid1 > tau)
err_boosting1 <- 1 - mean(yhat.boost.class.valid1==ytrue_boost)

yhat.boost.valid2 <- predict(boostingFit2,valid,n.trees=1000, type ="response")
yhat.boost.class.valid2 <- as.numeric(yhat.boost.valid2 > tau)
err_boosting2 <- 1 - mean(yhat.boost.class.valid2==ytrue_boost)

yhat.boost.valid3 <- predict(boostingFit,valid,n.trees=5000, type ="response")
yhat.boost.class.valid3 <- as.numeric(yhat.boost.valid3 > tau)
err_boosting3 <- 1 - mean(yhat.boost.class.valid3==ytrue_boost)

validation.boost.error <- cbind("Boost 1" = err_boosting1, "Boost 2" = err_boosting2,
                                "Boost 3" = err_boosting3)
kable(validation.boost.error, "latex", caption = "Validation Error - Boosting",
      booktabs = T) %>%  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 2: Validation Error - Boosting

Boost 1	Boost 2	Boost 3
0.3274845	0.3462733	0.2624224

2.5 Evaluation of Prediction Accuracy

For each of the model, we calculate the predicted values and define all observations as having been unemployed at a point in time if the predicted probability is higher than tau. Tau is 27 percent and the average probability in our sample to have been unemployed at some point. When comparing the predicted class with the true class it is obvious that all methods misclassify a significant share of the observations. Next, we compare the classification erros of the four different models. We calculate the classification error for the logistic regression using only all non-missing cases as mentioned above. For the other three methods we can use observations that have missing values as well. It becomes clear that the boosting has the smallest classification error, followed by a slightly smaller classifications errors for the pruned tree, as well as similar classifications errors for the large tree and the logistic model.All four methods missclassify about one third of the sample.

```r
library(dplyr)
tau <- length(which(dataset$uemp3m=="Yes"))/length(dataset[,1])
tau
```

```
## [1] 0.2734692
```

```r
# logit
transform_test_nonmiss <- transform(test_nonmiss,uemp3m=as.numeric(uemp3m)-1)
ytrue.nonmiss <- transform_test_nonmiss$uemp3m
yhat.logit <-  predict(logit,test_nonmiss, type = "response")
yhat.logit.class <- as.numeric(yhat.logit > tau)
err_logit <- 1 - mean(yhat.logit.class==ytrue.nonmiss)
# classification trees:
transform_test <- transform(test,uemp3m=as.numeric(uemp3m)-1)
ytrue <- transform_test$uemp3m
## tree not pruned:
yhat.tree.large.nonmiss <- predict(tree_large, newdata = test_nonmiss, type = "prob")[,2]
yhat.tree.large <- predict(tree_large, newdata = test, type = "prob")[,2]
yhat.tree.large.class <- factor(yhat.tree.large>tau, labels = c(0,1))
err_tree_large <- 1-mean(yhat.tree.large.class==ytrue)
## pruned tree
yhat.tree.nonmiss <- predict(tree, newdata = test_nonmiss, type = "prob")[,2]
yhat.tree <- predict(tree, newdata = test, type = "prob")[,2]
yhat.tree.class <- factor(yhat.tree>tau, labels = c(0,1))   # mind the order!
err_tree <- 1-mean(yhat.tree.class==ytrue)
# boosting
yhat.boost.nonmiss <- predict(boostingFit,test_nonmiss,n.trees=5000, type ="response")
yhat.boost <- predict(boostingFit,test,n.trees=5000, type ="response")
yhat.boost.class <-  factor(yhat.boost>tau, labels = c(0,1))
err_boosting <- 1 - mean(yhat.boost.class==ytrue)

classification.error <- cbind("Logit" = err_logit, "Large Tree" = err_tree_large,
                              "Pruned Tree" = err_tree, "Boosting" = err_boosting)
```

Table 3: Logit

	0	1
0	4365	2359
1	1001	1633

Table 4: Large Tree

	0	1
0	7664	2734
1	2301	1613

Table 5: Pruned Tree

	0	1
0	8952	1446
1	2708	1206

Table 6: Boosting

	0	1
0	9302	1096
1	2740	1174

Table 7: Classification Error

Logit	Large Tree	Pruned Tree	Boosting
0.3590511	0.3518027	0.2902459	0.2680268

In addition, we evaluate the performance of the methods by calculating the area under the ROC curve and ploting the ROC curve. The area under curve evaluates the ability of a method to correctly distinguish between unemployed and employed, thus the higher it is, the better. All of our methods have quite a good value of the area under the curve which is between 0.62 and 0.87, though the boosting appraoch has the highest value. This can also be seen in the ROC curve, where the highest curve is the one of the boosting approach, followed by the large tree, logistic model and the pruned three. Thus according to the ROC curve and AUC, the boosting and the large tree perform the best, where the latter differs from the result we got when comparing the classification errors. Here, the pruned three has the lowest AUC, indicating that the pruned tree can distinguish between employed and unemployed the least. We had to use the complete cases data set, as otherwise we could not compare the logistic regression with the other methods by the ROC curve and this could be a reason why the results differ. The larger tree being more accurate then the pruned one is plausible, when dealing with a complex dataset, which thus requires more complex methods. The reason why the pruned tree has a smaller classification error than the larger tree, is that the pruned tree is pruned by minimizing the classification error.

Using the two different measures to compare the four methods, it becomes clear that the boosting approach has the best performance, while the other three methods are not comparable in those two measures.

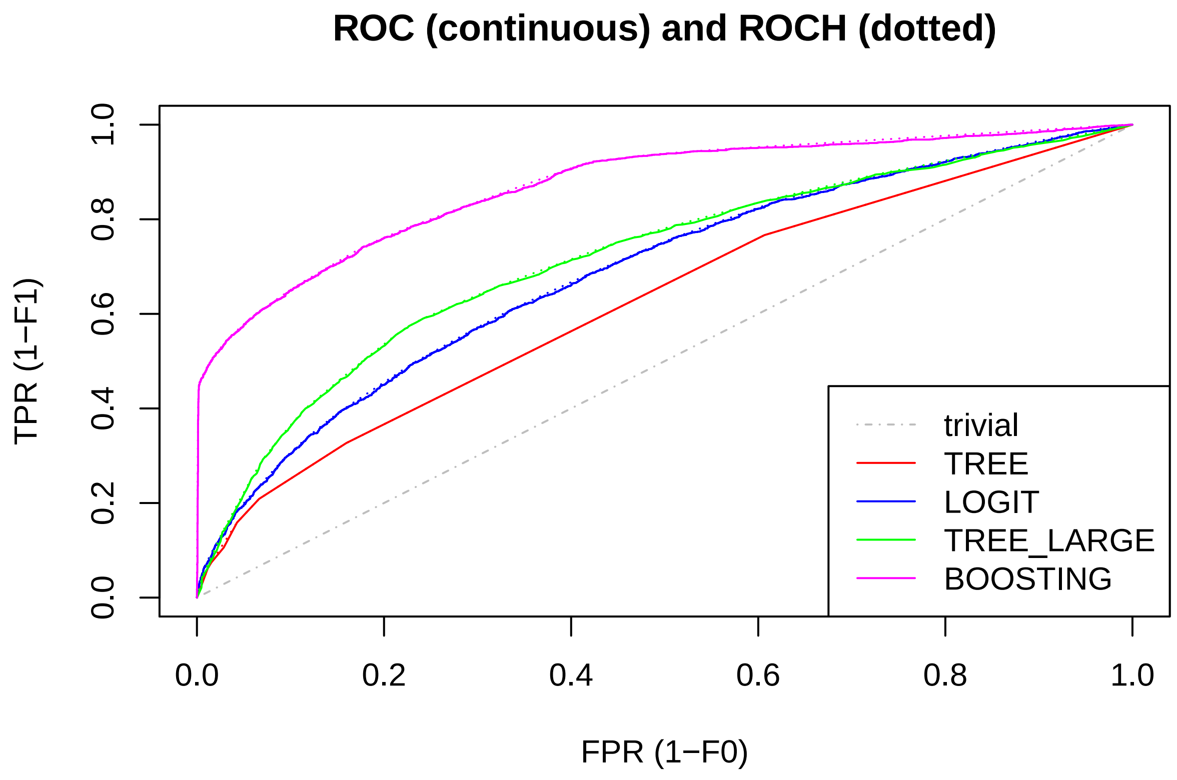
```r
library("caret")
library("hmeasure")

# ROC
prediction.roc <- data.frame(TREE = yhat.tree.nonmiss, LOGIT = yhat.logit,
                             TREE_LARGE = yhat.tree.large.nonmiss,
                             BOOSTING = yhat.boost.nonmiss)
h <- HMeasure(true.class = as.numeric(test_nonmiss$uemp3m),
              scores = prediction.roc) # scores is the distribution of probabilities
AUC <- cbind(h$metrics[1,3],h$metrics[2,3],h$metrics[3,3],h$metrics[4,3])
colnames(AUC) <- c("Pruned Tree", "Logit", "Large Tree", "Boosting")
kable(AUC, "latex", caption = "AUC",
      booktabs = T) %>%  kable_styling(latex_options = c("striped", "hold_position"))
```

```r
plotROC(h, which = 1)
```

Table 8: AUC

Pruned Tree	Logit	Large Tree	Boosting
0.6247731	0.6865699	0.7155503	0.8698514



3 Literature

European Social Survey, (2019). ESS9 – 2018 DOCUMENTATION REPORT.

Fahrmeier, L., Kneib, T., Lang, S. & Marx, B., 2013. Regression - Models, Methods and Applications. New York: Springer.

Fawcett, T., (2006). An introduction to ROC analysis. In: Pattern recognition letters 27 (8), pp. 861–874.

Hastie, T., Tibshirani, R. & Friedman, J., 2001. The Elements of Statistical Learning - Data Mining, Inference and Prediction. 2 Hrsg. Stanford: Springer.

James, G., Witten, D., Hastie, T. & Tibshirani, R., (2013). An Introduction ot Statistical Learning with Applications in R. 1 Hrsg. Stanford: Springer.

Mullainathan, S. and J. Spiess (2017). "Machine learning: an applied econometric approach".In: Journal of Economic Perspectives 31 (2), pp. 87–106.

Varian, H. R. (2014). "Big data: New tricks for econometrics". In: Journal of Economic Perspectives 28 (2), pp. 3–28.