Introduction to Machine Learning

(NPFL054)

Homework 2

François Leroy, PhD student at CZU

2021-05-24

Contents

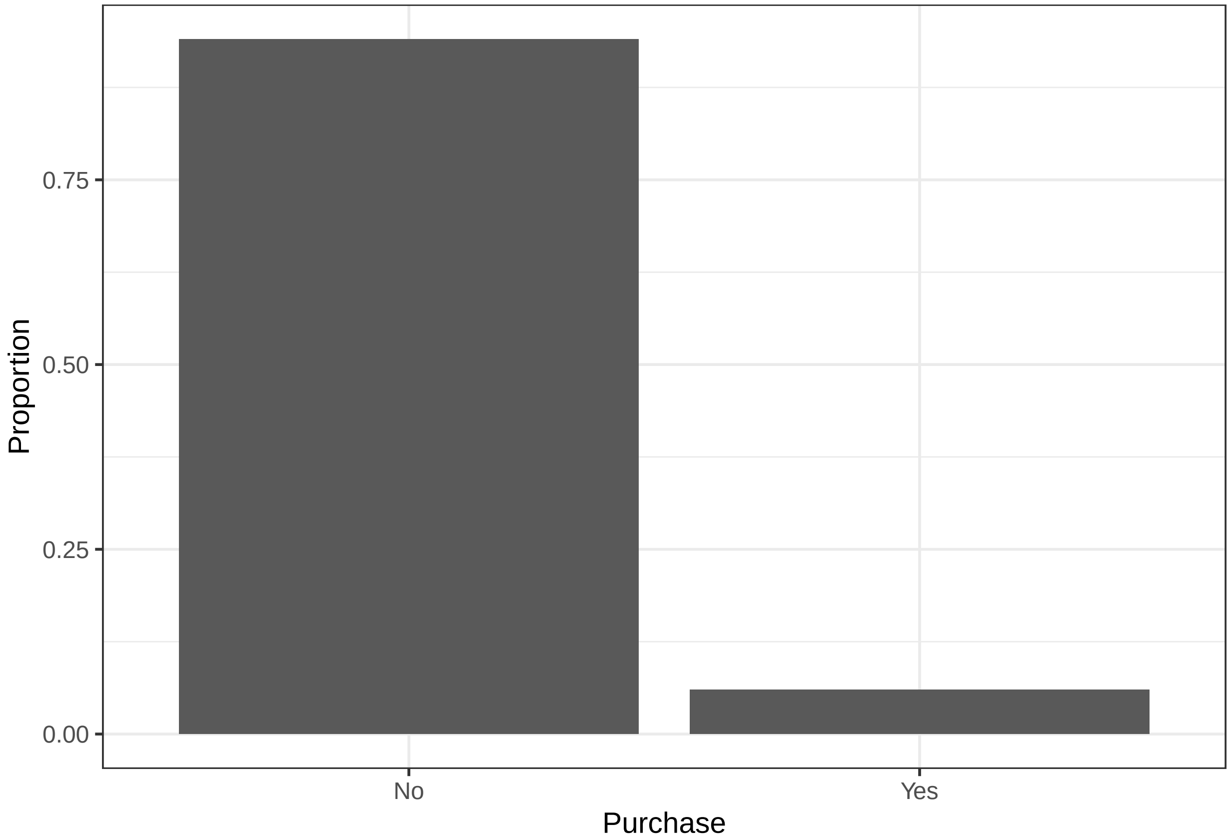
Set up the project

```r
rm(list = ls())
library(ISLR) # for the data
library(tidyverse) # convenient
library(rpart) # for decision trees
library(randomForest) # for ensemble learning
library(glmnet) # for regularized logistic regression
library(ROCR) # for ROC curves
```

```r
## Reproduce the result
set.seed(123)
## Create the splitting vector
split <- sample(nrow(Caravan), 1000)
## Create the test dataset
d_test <- Caravan[split,]
## Create the training dataset
d_train <- Caravan[-split,]
```

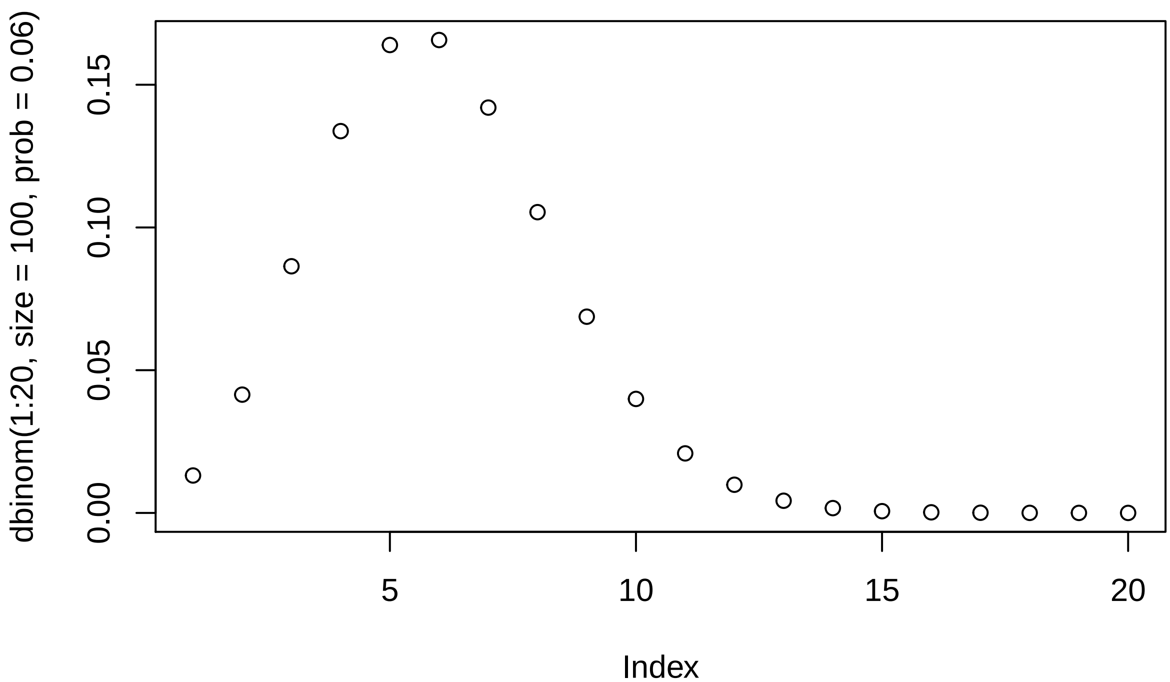
1. Task 1 - Data analysis

- **First, check the distribution of the target attribute. What would be your precision if you select 100 examples by chance?**



We can see that there is 94% of customers who didn't purchase the insurance and that 6% who did. From this, we can compute the following Probability Mass Function of this binomial distribution:

```
plot(dbinom(1:20, size = 100, prob = .06))
```



The precision is the number of examples classified as *Yes* when the value is actually *Yes*. Here, the precision should be 0.06, which is actually the ratio between “Yes” and “No”.

- **1.a. Focus on the customer type MOSHOOFD: create a table with the number of customers that belong to each of 10 L2 groups and the percentage of customers that purchased a caravan insurance policy in each group. Comment the figures in the table. Then do the same for the customer subtype MOSTYPE (41 subgroups defined in L1).**

MOSHOOFD type:

```
Caravan %>%
  count(MOSHOOFD, Purchase) %>%
  group_by(MOSHOOFD) %>%
  summarise(size = sum(n),
            purchase_prop = round(n[Purchase == "Yes"]/sum(n), 2)) %>%
  rename(group = MOSHOOFD) %>%
  kableExtra::kable()
```

group	size	purchase_prop
1	552	0.09
2	502	0.13
3	886	0.07
5	569	0.03
6	205	0.02
7	550	0.04
8	1563	0.06
9	667	0.06
10	276	0.02

This table shows the number of individuals (column $size$) and the proportion of customers that will buy an insurance in each group (column $purchase\_prop$) of the *MOSHOOFD* variable. The *MOSHOOFD* attribute correspond to the customer main type. We can see that the customers that are more prone to purchase an insurance are the one belonging to the group 2, *i.e.* the *driven growers* (13% of them will buy an insurance). Then, the *successful hedonist* are more likely to buy an insurance (*group 1*, 9% of them). The names of these two groups suggest that they are rather wealthy individuals. On the other hand, the customers belonging to the class 6 and 10, respectively the *cruising seniors* and the *farmers*, are less likely to subscribe to the insurance (only 2% in each group). This is also quite expected, as seniors and farmers can be in precarious situations.

<u>MOSTYPE type:</u>

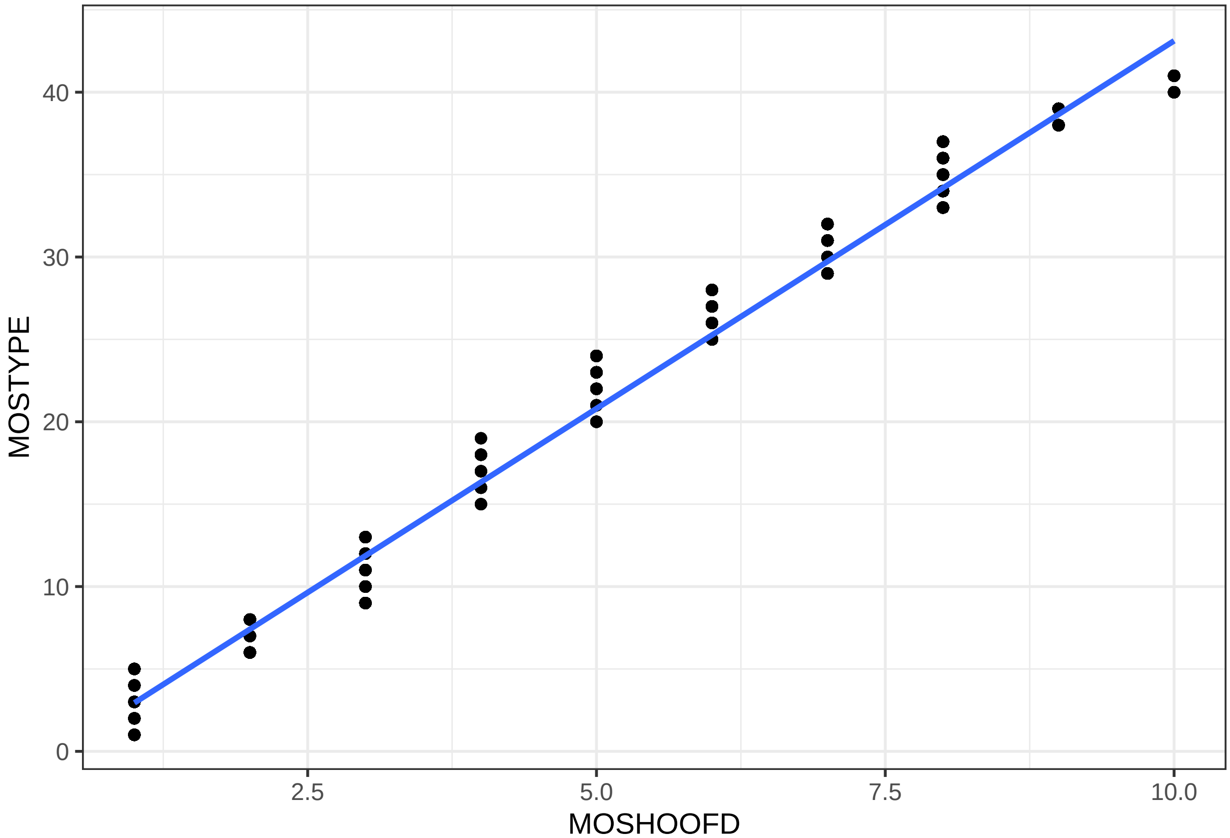
```r
table <-
Caravan %>%
  count(MOSTYPE, Purchase) %>%
  group_by(MOSTYPE) %>%
  summarise(size = sum(n),
            purchase_prop = round(n[Purchase == "Yes"]/sum(n), 2)) %>%
  rename(group = MOSTYPE) %>%
  arrange(desc(purchase_prop))
## Display in 2 columns
kableExtra::kable(list(table[1:(nrow(table)/2),],
```

```r
                         table[((nrow(table)/2)+1):nrow(table),])) %>%
  kableExtra::kable_styling(latex_options = "HOLD_position")
```

group	size	purchase_prop	group	size	purchase_prop
8	339	0.15	10	165	0.05
12	111	0.14	34	182	0.05
1	124	0.10	4	52	0.04
3	249	0.10	5	45	0.04
6	119	0.10	9	278	0.04
20	25	0.08	22	98	0.04
37	132	0.08	35	214	0.04
2	82	0.07	24	180	0.03
7	44	0.07	30	118	0.03
13	179	0.07	31	205	0.03
36	225	0.07	23	251	0.02
38	339	0.07	25	82	0.02
11	153	0.06	26	48	0.02
32	141	0.06	27	50	0.02
33	810	0.06	29	86	0.02
39	328	0.06	41	205	0.02

This table is the same than the previous one but for the *MOSTYPE* variable, which gives more information about the social status. It is order by descending proportion of purchase. The two groups more prone to buy an insurance are the group 8 and 12, which correspond respectively to *middle class families* and *affluent young families*. Thus, we can say that families are potential good targets to sell insurances. We can see that the class 25, 26, 27 and 29 all have a low proportion of individuals buying a insurance. They are all related to old people (*i.e.*, *Young seniors in the city*, *Own home elderly*, *Seniors in apartments*, *Porchless seniors: no front yard*). Thus, as said just above for the *MOSHOOFD* variable, old people don't seem to be good targets to sell insurances. Moreover, the group 41, *i.e.* the *mixed rurals* are also less prone to subscribe an insurance, as expected with the *MOSHOOFD* variable.

1.b. Analyze the relationship between features MOSHOOFD and MOSTYPE.

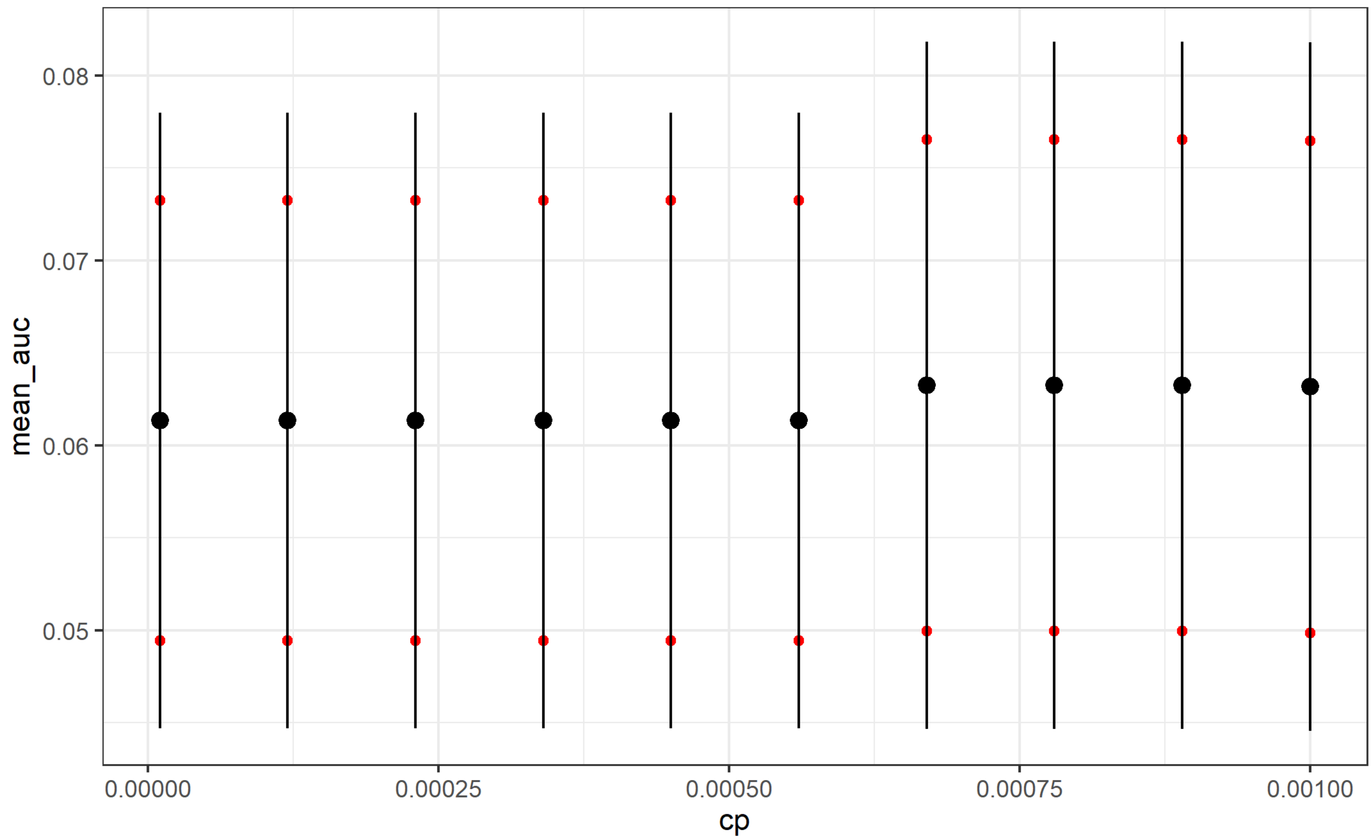


We can clearly see a relationship between these two features which are MOSHOOFD = *Customer main type* and MOSTYPE = *Customer Subtype*. This is expected because MOSTYPE is just a more precise social position. For instance, we can see that when $MOSHOOFD = 10$, $MOSTYPE = 40|41$. We can see that $MOSHOOFD = 10$ correspond to *Farmers* and that $MOSTYPE = 40|41$ are two subclasses of farmers: *Large family farms* and *Mixed rurals*, respectively.

2. Task 2 - Model fitting, optimization, and selection

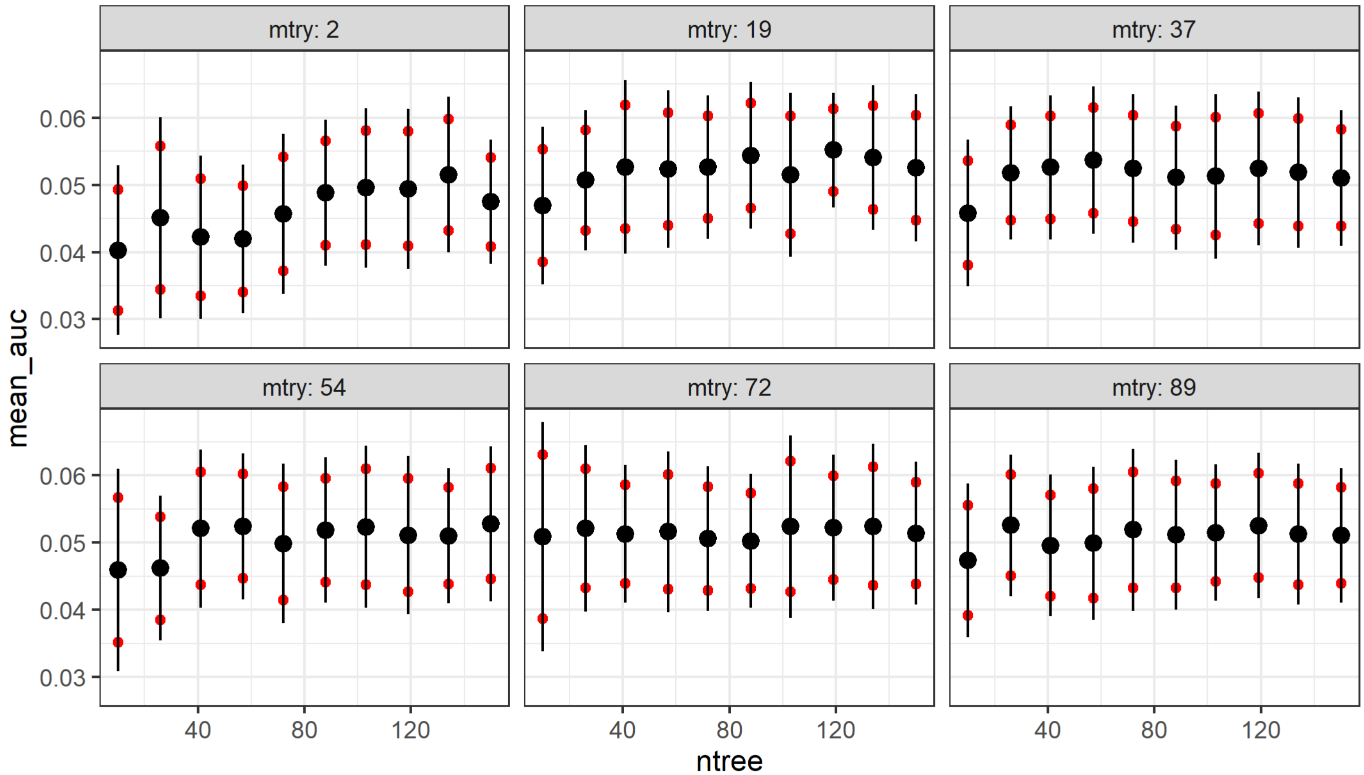
```r
## Function to randomly extract the test dataset in d_train
## using always the same number of positive and negative
## values of Purchase
prepare_cv_folds <-  function(k){
  # Create the subsets data containing Purchase == Yes
  # in one hand and Purchase == No in an other hand
  pos_data <- d_train[d_train$Purchase == "Yes",]
  neg_data <- d_train[d_train$Purchase == "No",]
  ## Compute the size of each fold
  fold.size.pos <- nrow(pos_data)%/%k
  fold.size.neg <- nrow(neg_data)%/%k
  ## Randomly rearrange the indexes
  set.seed(12); s_pos <- sample(nrow(pos_data))
  set.seed(12); s_neg <- sample(nrow(neg_data))
  ## create the list that will contain the test folds
  f.idx <-  list()
  ## For each fold, extract the dataset that will be used as test
  for(i in 1:k){
      f.idx[[i]] <-
        rbind(pos_data[s_pos[(1 + (i-1)*fold.size.pos):(i*fold.size.pos)],],
              neg_data[s_neg[(1 + (i-1)*fold.size.neg):(i*fold.size.neg)],])
  }
  return(f.idx)
}
## Use the function to create the 10 test datasets
split_data <- prepare_cv_folds(10)
```

2.1 Decision tree



The graphique above shows the mean AUC as a function of different values of cp. The black lines represent the standard deviation and the red dots the Confidence Intervals (computed with the `t.test()` function and with $\alpha = 5\%$). As we can see the mean $AUC_{0.2}$ stay stable for the six firsts values of cp and increase a bit for $cp = 6.7 \times 10^{-3}$ and then stay stable with increasing cp. reducing the complexity parameter below $cp = 0.001$ doesn't change the mean $AUC_{0.2}$. The cp parameter indicates the minimum change of the training error rate to consider the splitting process. As a low cp means a more complex model, we are looking for the highest value of cp maximizing the mean $AUC_{0.2}$. We can see that $cp = 0.001$ is already sufficiently low. Thus, we can select $cp = 0.001$ to learn the decision tree.

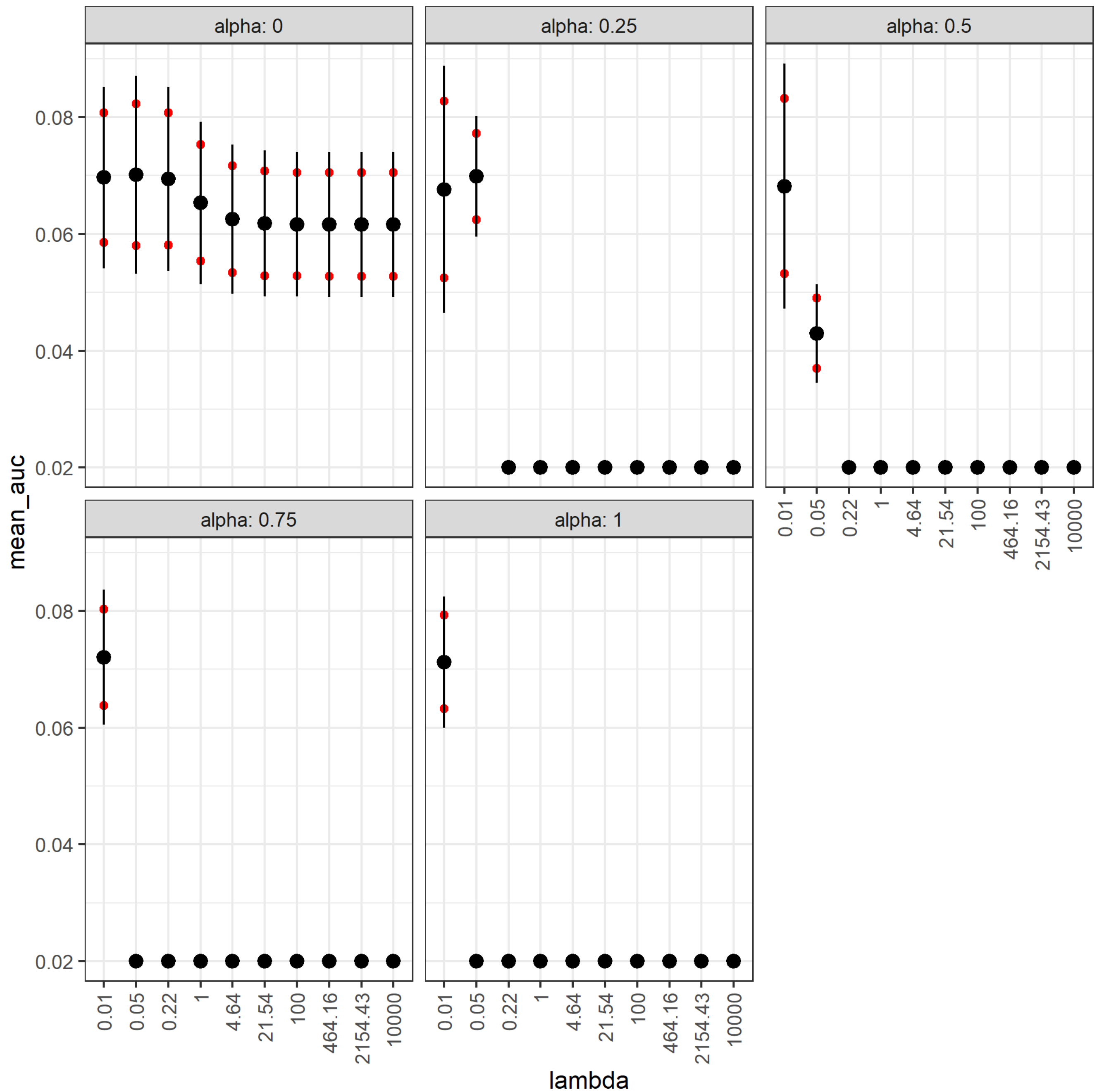
2.2 Random Forest



This plot shows the mean auc as a function of the number of trees (*i.e.* $ntree$). Each square correspond to a value of $mtry$, *i.e.* the number of features used in the splitting process. As we can see, the highest value of $AUC_{0.2}$ is for $mtry = 19$ and $ntree = 120$.

We know that the theoretical value of $mtry$ for the classification task is $\sqrt{number\ of\ feature}$. Here, it is equal to 10, which is quite close to the selected $mtry = 19$.

2.3 Regularized logistic regression



This figure shows the mean $AUC_{0.2}$ for different values of two hyperparameters of the elastic net regularization:

- α: which is the weight given to the two types of penalties (L1 and L2, see *Element of Statistical Learning*, Hastie *et al.* 2001). $\alpha = 1$ correspond to the lasso penalty (*i.e.* the L1 penalty) and

$\alpha = 0$ correspont to the ridge regularization (*i.e.* the L2 penalty).

- λ: which is the weighting of the penalties to the loss function. Thus, $\lambda = 0$ means no weighting and is an unregularized logistic regression whilst $\lambda = 1$ means a fully weighted penalty.

As we can see on this figures, the highest value of $AUC_{0.2}$ seems to be for $\alpha = 0.75$ and $\lambda = 0.01$.

2.4 Note about hyperparameters selection

As represented on the previous plots, most of the $AUC_{0.2}$ confidence intervals overlap, which means that their differences are statistically non significant. However, a choice was necessary and I decided to always choose the hyperparameters giving the lowest $AUC_{0.2}$ values.

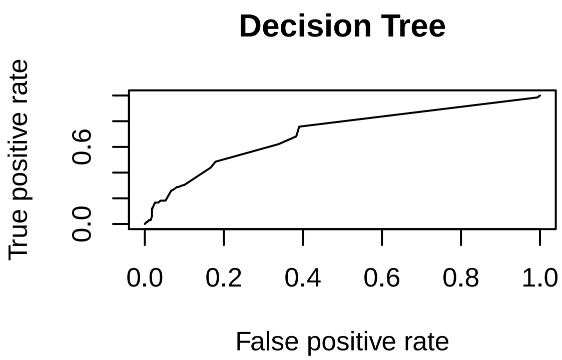
2.5 Evaluation on the test dataset

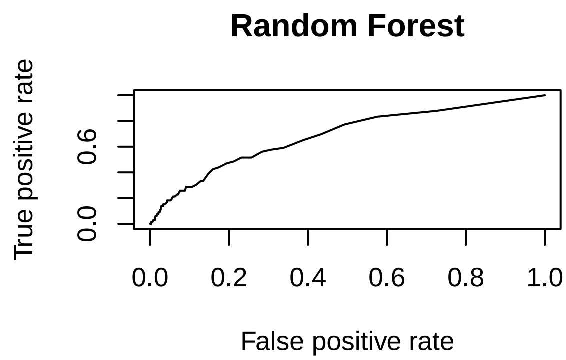
Thanks to the previous steps, we have chosen:

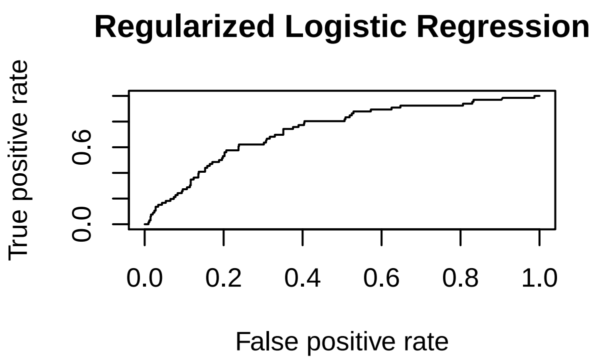
1. Decision tree with $cp = 0.001$
2. Random forest with $ntree = 120$ and $mtry = 19$
3. Regularized logistic regression with $\lambda = 0.01$ and $\alpha = 0.75$

Now, we can **1)** learn the model on the entire training dataset and **2)** compute the $AUC_{0.2}$ when predicting on the test dataset.

The following plots show the ROC curve (Receiver Operating Characteristic) of the three optimized models with their respective $AUC_{0.2}$ displayed in the table.





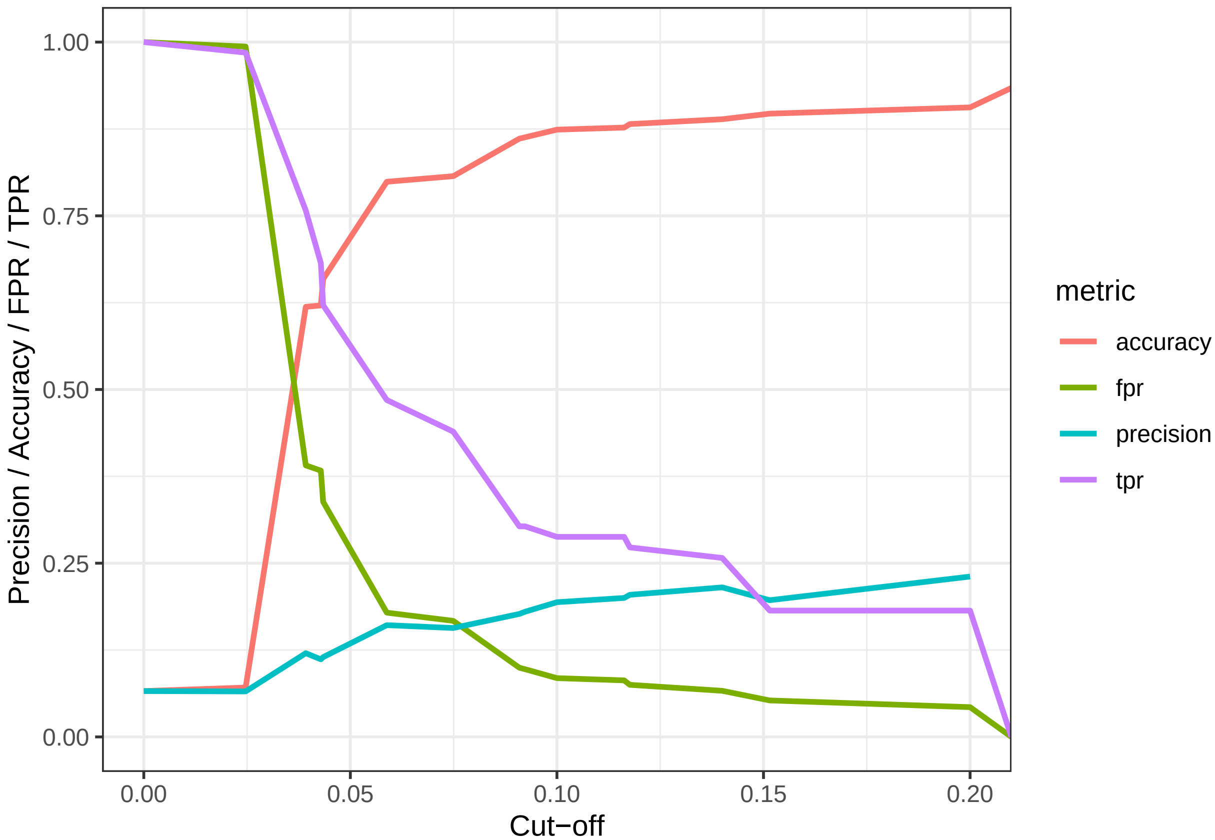


model	auc
Decision Tree	0.060
Random Forest	0.055
Regularized Logistic Regression	0.056

As we can see, the highest value of $AUC_{0.2}$ is for the decision tree with $cp = 0.001$. It is interesting to note the difference between the $AUC_{0.2}$ computed on the test dataset and the one computed from the cross-validation. From the cross-validation, the highest value of $AUC_{0.2}$ was for the regularized logistic regression, with $AUC_{0.2} \approx 0.07$ whilst for the decision tree it was slightly lower than 0.065. Now, we can see that $AUC_{0.2}$ is higher for the decision tree. We can think that the decision tree is better at generalization than the regularized logistic regression, which was slightly overfitted.

2.6 Setting cutoff threshold

The cut-off indicates the probability at which we should start to consider the prediction as True. The following plot shows the values of precision, accuracy, True Positive Rate (TPR) and False Positive Rate (FPR) according to different cut-off threshold. So far, all the learning processes have been done using the $AUC_{0.2}$, which is the Area Under the Curve just up to $FPR \leq 20\%$. Thus, I decided not to consider cut-off higher than 0.2.



First, we can see that the precision and accuracy start closely to what a random classifier would do, *i.e.* close to 0.06. We can also observe that the maximum value of accuracy is very quickly reach: this is due to the fact that our dataset is really homogeneous, *i.e.* we have a lot of *No* for very little *Yes*. Thus, the classifier will quickly correctly classify all the actual *No* as *No* for low values of the cut-off and the accuracy will be high. Here, we need to focus on the correctly classified *Yes*, and for this we use the precision ($P = \frac{TP}{TP+FP}$). We can see that the highest value of precision is reached for $cutoff = 0.2$.

Now, we can compute the confusion matrix with a threshold of 0.2:

```
##      obs
## pred   0   1
##    0 894  54
##    1  40  12
```

3. Task 3 - Model interpretation and feature selection

Table 3.1: Variable importance for the decision tree

feature	MeanDecreaseGini	feature	MeanDecreaseGini
PPERSAUT	16.39	MBERMIDD	3.81
PBRAND	12.61	MGODPR	3.63
APERSAUT	12.06	MINKGEM	3.53
MKOOPKLA	10.73	APLEZIER	3.19
PWAPART	9.52	PPLEZIER	3.19
MOSTYPE	9.41	MHKOOP	3.14
MFGEKIND	9.25	MSKD	3.12
MBERHOOG	8.81	MFALLEEN	3.03
AWAPART	8.71	MSKC	2.90
MOSHOOFD	7.16	MGODOV	2.85
MOPLMIDD	6.77	MAUT0	2.73
MOPLLAAG	6.68	MSKB2	2.63
MRELGE	6.64	MBERARBO	2.52
ABRAND	6.16	MINK7512	2.35
MFWEKIND	5.93	MAUT1	2.10
MINK4575	5.69	MINKM30	1.83
MGODGE	5.58	MAUT2	1.77
MRELOV	5.25	MRELSA	1.13
MZPART	4.99	MAANTHUI	0.83
MOPLHOOG	4.83	MBERZELF	0.79
MZFONDS	4.68	MGODRK	0.78
MBERARBG	4.66	ALEVEN	0.70
MSKA	4.65	PLEVEN	0.70
MGEMLEEF	4.55	AWALAND	0.34
MHHUUR	4.53	PWALAND	0.34
MSKB1	4.24	ABYSTAND	0.31
MINK3045	4.14	PBYSTAND	0.31
MGEMOMV	3.85	PTRACTOR	0.30

Table 3.2: Variable importance for the random forest

feature	MeanDecreaseGini	feature	MeanDecreaseGini	feature	MeanDecreaseGini
PBRAND	20.09	MINKM30	7.55	AMOTSCO	2.73
MOSTYPE	17.20	MSKA	7.47	PBROM	2.70
PPERSAUT	16.73	MINK7512	7.41	PFIETS	2.62
APERSAUT	14.74	MZFONDS	7.40	ABROM	2.44
MKOOPKLA	11.94	MGODOV	7.33	MAANTHUI	2.43
PWAPART	11.54	MFALLEEN	7.29	PWAOREG	2.36
MGODGE	11.08	MRELGE	7.20	PGEZONG	1.65
MBERMIDD	11.03	MAUT1	6.95	AWAOREG	1.59
MGODPR	10.97	MZPART	6.82	PTRACTOR	1.20
MOPLMIDD	10.90	MRELOV	6.26	PINBOED	1.15
MOSHOOFD	10.27	MAUT0	6.18	AGEZONG	1.11
MBERARBG	10.12	MAUT2	5.99	AINBOED	1.02
MINK3045	9.88	MGEMLEEF	5.83	PAANHANG	0.97
MOPLLAAG	9.86	ALEVEN	5.55	PWABEDR	0.88
MFGEKIND	9.49	MSKD	5.39	ATRACTOR	0.78
MFWEKIND	9.46	PLEVEN	5.07	AWABEDR	0.73
MOPLHOOG	9.39	MGEMOMV	4.77	AAANHANG	0.69
MBERARBO	9.11	MGODRK	4.72	PBESAUT	0.49
MHHUUR	8.98	AFIETS	4.71	ABESAUT	0.47
MINK4575	8.82	MRELSA	4.61	AZEILPL	0.27
MSKB1	8.74	MBERZELF	4.37	APERSONG	0.25
MBERHOOG	8.71	PBYSTAND	3.99	PZEILPL	0.22
MSKC	8.31	ABYSTAND	3.73	AWALAND	0.19
MHKOOP	8.27	PPLEZIER	3.70	PPERSONG	0.17
MINKGEM	7.92	APLEZIER	3.53	PWALAND	0.15
ABRAND	7.72	MBERBOER	3.38	PWERKT	0.04
MSKB2	7.66	PMOTSCO	3.24	PVRAAUT	0.01
AWAPART	7.65	MINK123M	2.76	AWERKT	0.01
				AVRAAUT	0.00

Table 3.3: Variable importance for the lasso regression

feature	s0	feature	s0
APLEZIER	1.2945711	PWABEDR	-0.0584269
AZEILPL	1.1414119	MOPLHOOG	0.0515774
ABYSTAND	0.7523564	MGODRK	-0.0435352
AFIETS	0.4165938	MAUT1	0.0356634
PWAOREG	0.2678450	MHHUUR	-0.0344877
PPERSAUT	0.2038757	MBERMIDD	0.0309560
MBERBOER	-0.1672857	PGEZONG	0.0288318
PWAPART	0.1551758	MGEMLEEF	0.0271291
MINK123M	-0.1506243	PWERKT	-0.0263719
PWALAND	-0.1470406	MGODGE	-0.0245771
ATRACTOR	-0.1432662	PBESAUT	-0.0218699
PBRAND	0.1026457	MKOOPKLA	0.0191547
MINKGEM	0.0765117	MRELSA	-0.0169228
MRELGE	0.0675735	MINK7512	0.0124976
MOPLLAAG	-0.0633978	MSKD	-0.0099615
		PLEVEN	-0.0020227

4. Task 4 - Final prediction on the blind test set

The final prediction is done using the Decision Tree with a complexity parameter of 0.00067 chosen because of its highest AUC.