Data Science for Actuaries (ACT6100)

Arthur Charpentier

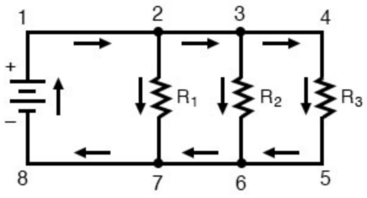
Aggregation # 5.1 (Ensemble & Parallel)

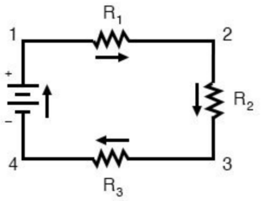
automne 2020

https://github.com/freakonometrics/ACT6100/

Apprentissage en parallèle ou en série ?

On va construire ici un ensemble de modèles





Apprentissage

Considérons un échantillon i.i.d. $\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^p \times \mathcal{Y}\}$,
une prévision est la construction d'une fonction $f : \mathbb{R}^p \to \mathcal{Y}$
E.g. $f^\star(\boldsymbol{x}) = \mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}]$, $f^\star = \text{argmin}\{\mathbb{E}[(Y - f(\boldsymbol{x}))^2]\}$
Comme $f^\star$ est inconnue (on ne connait pas la loi de $(\boldsymbol{X}, Y)$, on construit $\widehat{f}_n$ aussi proche que possible de f.
$\widehat{f}_n$ est consistante pour $\mathbb{P}$ si

$$\lim_{n\to\infty} \mathbb{E}\left(\int [f^\star(\boldsymbol{x}) - \widehat{f}_n(\boldsymbol{x})]^2 d\mathbb{P}(\boldsymbol{x})\right) = 0$$

et si $\widehat{f}_n$ est consistante pour toute distribution, on parle de consistance universelle.
Il existe des estimateurs universellement consistant.

Apprentissage

En classification, le classifier de Bayes est

$$f^\star(\boldsymbol{x}) = \mathbf{1}\big(\mathbb{P}[Y=1|\boldsymbol{X}=\boldsymbol{x}] \geq \mathbb{P}[Y=0|\boldsymbol{X}=\boldsymbol{x}]\big)$$

la éthode des *k*-plus proches voisins

$$\widehat{f}_n(\boldsymbol{x}) = \mathbf{1}\left(\sum_{i=1}^{k} \mathbf{1}(y_{i:\boldsymbol{x}}=1) \geq \sum_{i=1}^{k} \mathbf{1}(y_{i:\boldsymbol{x}}=0)\right)$$

où $y_{j:\boldsymbol{x}}$ est le label de la j-ème plus proche observation de $\boldsymbol{x}$ parmi $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$, et la méthode du noyau

$$\widehat{f}_n(\boldsymbol{x}) = \mathbf{1}\left(\sum_{i=1}^{n} \mathbf{1}(y_i=1)\mathbf{1}(\|\boldsymbol{x}-\boldsymbol{x}_i\| \leq h) \geq \sum_{i=1}^{n} \mathbf{1}(y_i=0)\mathbf{1}(\|\boldsymbol{x}-\boldsymbol{x}_i\| \leq h$$

sont des classifieurs universellement consistant, si $k\to\infty$ mais $k/n\to 0$, et $h\to\infty$ mais $nh^p\to 0$ lorsque $n\to\infty$.

Ensemble Method

Galton(1907, Vox Populi) *guess the weight of the cow*, county fair in Cornwall, England, 1906

Correct answer = 1198 lbs

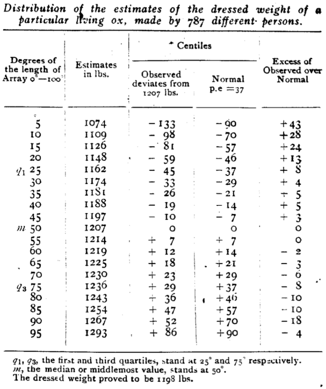
787 participants, $x_1, \cdots, x_n$

- quantile 25%: 1162 lbs
- quantile 50%: 1207 lbs
- quantile 75%: 1236 lbs
- average: 1197 lbs

Pick a single prediction x_j v.s average $\overline{x}$,

$$\mathbb{E}\left[(x_j - t)^2\right] = (\overline{x} - t)^2 + \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

where t is the truth (ambiguity decomposition)

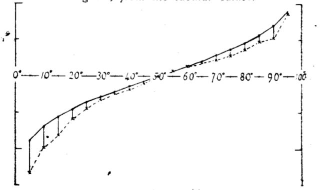


Distribution of the estimates of the dressed weight of a particular living ox, made by 787 different persons.

Degrees of the length of Array 0°—100°	Estimates in lbs.	Centiles: Observed deviates from 1207 lbs.	Centiles: Normal p.e =37	Excess of Observed over Normal
5	1074	− 133	− 90	+ 43
10	1109	− 98	− 70	+ 28
15	1126	− 81	− 57	+ 24
20	1148	− 59	− 46	+ 13
q_1 25	1162	− 45	− 37	+ 8
30	1174	− 33	− 29	+ 4
35	1181	− 26	− 21	+ 5
40	1188	− 19	− 14	+ 5
45	1197	− 10	− 7	+ 3
m 50	1207	0	0	0
55	1214	+ 7	+ 7	0
60	1219	+ 12	+ 14	− 2
65	1225	+ 18	+ 21	− 3
70	1230	+ 23	+ 29	− 6
q_3 75	1236	+ 29	+ 37	− 8
80	1243	+ 36	+ 46	− 10
85	1254	+ 47	+ 57	− 10
90	1267	+ 52	+ 70	− 18
95	1293	+ 86	+ 90	− 4

q_1, q_3, the first and third quartiles, stand at 25° and 75° respectively.
m, the median or middlemost value, stands at 50°.
The dressed weight proved to be 1198 lbs.

Diagram, from the tabular values.



The continuous line is the normal curve with p.e.=37.
The broken line is drawn from the observations.
The lines connecting them show the differences between the observed and the normal.

Ensemble Method

- statistical interpretation *Hopefully the ensemble generalises better than a single chosen model*
- computational interpretation *Averaging can sometimes be a fast way of reaching closer to the optimal than direct optimisation*
- representational interpretation *Averaging models of some model class can sometimes take you outside of that model class*

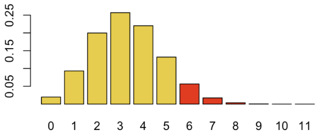
Regression problem: averaging
Classification problem: voting (majority)

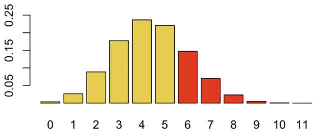
Votes

Condorcet (1785, Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix)
p: probabilité de se tromper, individuellement
La probabilité que la majorité se trompe (avec n votant)

$$\sum_{k\geq[(n+1)/2]} \binom{n}{k} p^k (1-p)^{n-k}$$

e.g. $n = 11$ et $p = 30\%$, and $p = 40\%$

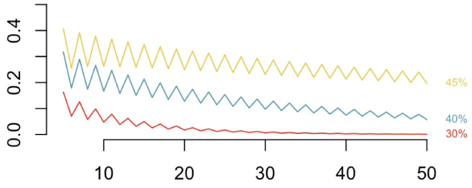




Probability that the majority is wrong, 7.8% and 24.6%.

Votes

Evolution of the probability that the majority is wrong, as a function of *n*, for various *p*



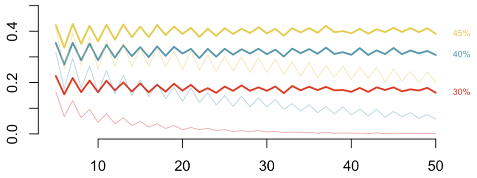
but this is valid only if votes are independent !
In ensemble learning, we want

- models to be reasonably* accurate
- models to be as independent as possible

* ensemble learning can turn a collection of poor learners (stumps (single-node decision trees) or linear) into a good one

Independence ?

with correlation among prediction, there is no real ‘decrease’



- split training sample into m groups of n/m observations

  pb, groups are small, poor predictions
- split features into m groups

  pb, poor predictions if only small number of (true) predictors
- use boostrap

  pb if models are not independent enough

Bayesian model averaging

Consider m models, suppose that one is the true one, but we don't know which one...
Let T denote the choice of the model, $T \in \{1, 2, \cdots, m\}$,

$$\mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}] = \sum_{t=1}^{m} \mathbb{P}[Y = 1, T = t|\boldsymbol{X} = \boldsymbol{x}]$$

$$\mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}] = \sum_{t=1}^{m} \underbrace{\mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}, T = t]}_{\widehat{y}^{(t)}} \underbrace{\mathbb{P}[T = t|\boldsymbol{X} = \boldsymbol{x}]}_{\omega_t}$$

which is a weighted average of predictions (or posterior class prediction), where weights

$$\omega_t = \mathbb{P}[T = t|\boldsymbol{X} = \boldsymbol{x}] \propto \mathbb{P}[\boldsymbol{X} = \boldsymbol{x}|T = t]$$

are likelihoods of models.

Stacking

Estimating model likelihood can be complicated
we can learn weights using a regression
(where individual model outputs are treated as features)

```r
url = "http://freakonometrics.free.fr/german_credit.
    csv"
credit = read.csv(url, header = TRUE, sep = ",")
F = c(1,2,4,5,7,8,9,10,11,12,13,15,16,17,18,19,20)
for(i in F) credit[,i]=as.factor(credit[,i])
```

create a training and a testing datasets

```r
set.seed(123)
i_test = sample(1:nrow(credit),size=333)
i_train = (1:nrow(credit))[-i_test]
```

Credit scoring with ensemble techniques

- model 1 : logistic regression

```
GLM = glm(Creditability~., data=credit[i_train,],
    family= binomial)
pGLM = predict(GLM, newdata=credit[i_test,], type="
    response")
classGLM = c(0,1)[1+(pGLM>.5)*1]
```

- model 2 : classification tree

```
library(rpart)
CART = rpart(Creditability~., data=credit[i_train,])
classCART = predict(CART, newdata=credit[i_test,],
    type="class")
```

- model 3 : SVM (with a Gaussian kernel)

```
library(kernlab)
SVM = ksvm(Creditability~., data=credit[i_train,],
    kernel = "rbfdot", C=1)
classSVM =predict(SVM, newdata=credit[i_test,])
```

Credit scoring with ensemble techniques

On the correlation of those 3 models,

```
> df = data.frame(as.numeric(classCART)-1,
+                 as.numeric(classGLM),
+                 as.numeric(classSVM)-1)
> names(df)=c("CART","GLM","SVM")
> cor(df)
          CART       GLM       SVM
CART 1.0000000 0.3777031 0.4888801
GLM  0.3777031 1.0000000 0.5528032
SVM  0.4888801 0.5528032 1.0000000
```

Consider a simple (majority based) voting procedure

```
> vote = ifelse(rowSums(df)>1.5,1,0)
```

Use AUC a classification metrics

```
> library(caret)
> auc = function(y) caret::confusionMatrix(data=as.
   factor(y),reference=credit[i_test,"Creditability
   "])$overall["Accuracy"]
> y = credit[i_test,"Creditability"]
```

Credit scoring with ensemble techniques

On the classification tree,

```
> table(df$CART,y)
       0    1
  0   42   46
  1   52  193
> auc(df$CART)
 Accuracy
0.7057057
```

on the GLM

```
> table(df$GLM,y)
       0    1
  0   43   43
  1   51  196
> auc(df$GLM)
 Accuracy
0.7177177
```

On the SVM

```
> table(df$SVM,y)
       0    1
  0   33   26
  1   61  213
> auc(df$SVM)
 Accuracy
0.7387387
```

on the majority votes

```
> table(vote,credit[
    i_test,y])
vote    0    1
   0   37   29
   1   57  210
> auc(vote)
 Accuracy
0.7417417
```

Credit scoring with ensemble techniques

```
> df$CREDIT = as.numeric(credit[i_test,"Creditability
   "])-1
```

```
> reglm = lm(CREDIT~CART+GLM+SVM,data=df)
> df$STACKLM = predict(reglm)
> auc((df$STACKLM>.5)*1)
 Accuracy
0.7627628
```

```
> reg = glm(CREDIT~CART+GLM+SVM,data=df,family=
   binomial)
> df$STACKGLM = predict(reg, type="response")
> auc((df$STACKGLM>.5)*1)
 Accuracy
0.7627628
```

idea of slacking

Credit scoring with ensemble techniques

Instead of a (majority based) voting procedure, why not consider predictive probabilities?

```
> pGLM = pGLM
> pCART = predict(CART, newdata=credit[i_test,], type
    ="prob")[,2]
> SVM = ksvm(Creditability~., data=credit[i_train,],
    kernel = "rbfdot",C=1, prob.model = TRUE)
> pSVM = predict(SVM, newdata=credit[i_test,],type="
    probabilities")[,2]
> pdf = data.frame(as.numeric(pCART),
+                  as.numeric(pGLM),
+                  as.numeric(pSVM))
> names(pdf)=c("CART","GLM","SVM")
> cor(pdf)
          CART      GLM       SVM
CART 1.0000000 0.516073 0.6101085
GLM  0.5160730 1.000000 0.8641040
SVM  0.6101085 0.864104 1.0000000
```

Credit scoring with ensemble techniques

Instead of a (majority based) voting procedure, why not consider predictive probabilities?

```
> pGLM = pGLM
> pCART = predict(CART, newdata=credit[i_test,], type
    ="prob")[,2]
> SVM = ksvm(Creditability~., data=credit[i_train,],
    kernel = "rbfdot",C=1, prob.model = TRUE)
> pSVM = predict(SVM, newdata=credit[i_test,],type="
    probabilities")[,2]
> pdf = data.frame(as.numeric(pCART),
+                  as.numeric(pGLM),
+                  as.numeric(pSVM))
> names(pdf)=c("CART","GLM","SVM")
> cor(pdf)
          CART      GLM       SVM
CART 1.0000000 0.516073 0.6101085
GLM  0.5160730 1.000000 0.8641040
SVM  0.6101085 0.864104 1.0000000
```

Credit scoring with ensemble techniques

One can consider the average (probability) prediction

$$\widehat{y}_i = \text{average}\big(\widehat{y}_i^{tree}, \widehat{y}_i^{glm}, \widehat{y}_i^{svm}\big) = \frac{1}{3}\big(\widehat{y}_i^{tree} + \widehat{y}_i^{glm} + \widehat{y}_i^{svm}\big)$$

```
> auc((apply(pdf,1,mean)>.5)*1)
 Accuracy
0.7417417
```

or

$$\widehat{y}_i = \text{median}\big(\widehat{y}_i^{tree}, \widehat{y}_i^{glm}, \widehat{y}_i^{svm}\big)$$

```
> auc((apply(pdf,1,median)>.5)*1)
 Accuracy
0.7477477
```

Ensemble techniques: Netflix price

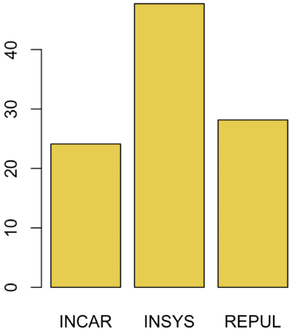
2009: US$1,000,000 Netflix Prize, Winner BellKor's Pragmatic Chaos
ensemble of 107 models
"*Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a simple technique* [...] *We strongly believe that the success of an ensemble approach depends on the ability of its various predictors to expose different complementing aspects of the data. Experience shows that this is very different than optimizing the accuracy of each individual predictor*"

(un)-stability of trees

```
> variable=rep(NA,10000)
> for(i in 1:10000){
+ arbre = rpart(PRONO~., data=
    myocarde[sample(1:71,size=47)
    ,])
+ variable[i] = as.character(
    arbre$frame[1,"var"])
+ }
> table(variable)/100
INCAR INSYS REPUL
24.12 47.72 28.16
```

the first split in the tree in the myocarde dataset is

- INSYS (47.7%)
- REPUL (28.2%)
- INCAR (24.1%)



Principe

- Les arbres sont peu robustes (sur la forme, pas forcément la prévision)
- L'approche présentée avec les arbres de décision conduit à des prédictions ayant une **grande variance**
- Le *bagging*, ou bootstrap aggregating est un ensemble de (méta-)algorithmes conçus pour améliorer la stabilité et la précision d'un modèle (ou d'un algorithme) utilisé. Il permet également de réduire la variance et d'éviter le surajustement.

Algorithme Bagging

1. Générer B bases de données d'entrainement différentes.
2. À partir de chacune des bases de données d'entrainement, construire un modèle $\widehat{f}_b$, $b = 1, \ldots, B$.
3. Pour une nouvelle observation $\boldsymbol{x}$, la prédiction agrégée (*bagging*) sera

$$\widehat{f}_{ag}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B}\widehat{f}_b(\boldsymbol{x}).$$

Étape 1

(1) Générer les B bases de données.

- En pratique, on n'a pas accès à B bases de données différentes.
- Si on divise la base de données initiale en B bases indépendantes (comme pour de la validation croisée), on perd de l'information et les bases ainsi créées sont trop petites (il faut que B soit grand pour que le *Bagging* soit intéressant).
- On va plutôt utiliser du *bootstrap* pour générer ces B bases de données.
- La base de données b, $b = 1, \dots, B$, de taille n est obtenue en pigeant au hasard **avec remise** n observations de la base de données initiale.
- Les B bases de données sont créées à partir de la même base initiale $\rightarrow$ les prédictions obtenues $\widehat{f_b}$ **ne sont pas** indépendantes.

Étape 2

(2) Construction du modèle b, $b = 1, \dots, B$.

- En pratique, quand la technique du *bagging* est utilisée pour des arbres de décision, ces derniers **ne sont pas** élagués.
- Chacun des arbres b, $b = 1, \dots, B$, a donc un faible biais mais une grande variance.
- La réduction de la variance se fait à l'étape 3 en agrégeant les différents arbres obtenus.

Étape 3

(3) Agrégation des modèles

$$\begin{aligned}
\text{Var}\big(\widehat{f}_{ag}(\boldsymbol{x})\big) &= \text{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\widehat{f}_b(\boldsymbol{x})\right)\\
&= \frac{1}{B^2}\text{Var}\left(\sum_{b=1}^{B}\widehat{f}_b(\boldsymbol{x})\right)\\
&= \frac{1}{B^2}\sum_{b_1=1}^{B}\sum_{b_2=1}^{B}\text{Cov}(\widehat{f}_{b_1}(\boldsymbol{x}),\widehat{f}_{b_2}(\boldsymbol{x}))\\
&\leq \frac{1}{B^2}B^2\text{Var}(\widehat{f}_b(\boldsymbol{x})) = \text{Var}(\widehat{f}_b(\boldsymbol{x}))
\end{aligned}$$

puisque lorsque $b_1 \neq b_2$, on a $\text{Corr}[\widehat{f}_{b_1}(\boldsymbol{x}),\widehat{f}_{b_2}(\boldsymbol{x})] \leq 1$.

Étape 3

(3) Agrégation des modèles

En fait, si $\text{Var}(\widehat{f}_b(\boldsymbol{x})) = \sigma^2(\boldsymbol{x})$ et $\text{Corr}[\widehat{f}_{b_1}(\boldsymbol{x}), \widehat{f}_{b_2}(\boldsymbol{x})] = r(\boldsymbol{x})$,

$$\text{Var}\big(\widehat{f}_{ag}(\boldsymbol{x})\big) = r(\boldsymbol{x})\sigma^2(\boldsymbol{x}) + \frac{1 - r(\boldsymbol{x})}{B}\sigma^2(\boldsymbol{x})$$

La variable sera d'autant plus faible que l'on aggrège des modèles différents.
L'instabilité des arbres en fait de bons candidats pour de l'aggrégation

Algorithme Bagging

- Empiriquement, on observe qu'agréger des centaines, voire des milliers d'arbres ($B = 100, 1000, 10000, \ldots$) augmente beaucoup le pouvoir prédictif des arbres de décision.
- Par contre, les résultats sont plus difficiles à interpréter: pas de graphique simple, etc.
- Les logiciels qui offrent ce type de méta-algorithme proposent généralement des méthodes permettant de mesurer l'importance des variables explicatives dans le modèle.

Random Forest

Algorithm 1: Random Forest

```
1 initialization : $m \leq p$ (number of features considered to split a
   node), $B$ (number of trees);
2 for $b = 1, 2, \ldots B$ do
3     generate a bootstrap sample $\mathcal{D}_n^b$;
4     $\widehat{f}_b \leftarrow$ tree (CART), where each split is done by minimizing
       some cost over a set of $m$ features randomly chosen
5 $\widehat{f}_{rf}(\boldsymbol{x}) = \dfrac{1}{B}\sum_{b=1}^{B}\widehat{f}_b(\boldsymbol{x})$
```

The smaller m, the smaller the correlation between trees

Classically, $m = p/3$ for regression, $m = \sqrt{p}$ for classification

Random Forest

- erreur out-of-bag

Soit $\mathcal{B}_i \subset \{1, 2, \cdots, B\}$ l'ensemble des arbres tels que $i \notin \mathcal{D}_n^b$,

$$\widehat{y}_i = \frac{1}{|\mathcal{B}_i|} \sum_{b \in \mathcal{B}_i} \widehat{f}_b(\boldsymbol{x}_i)$$

que l'on utilise pour mesurer un risque empirique out-of-bag

$$\frac{1}{n} \sum_{i=1}^{n} \big(y_i - \widehat{y}_i\big)^2 \text{ ou } \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\big(y_i \neq \widehat{y}_i\big)$$

- importance des variables (1)

cf discussion dans la partie 4 sur les arbres

Random Forest

- importance des variables (2)

Soit $\mathcal{I}_b$ l'échantillon associé à $\mathcal{D}_b$, et $\overline{\mathcal{I}}_b$ la partie out-of-bag. Sur cet échantillon, on définie le risque out-of-bag

$$\widehat{R}_b^{oob} = \frac{1}{\overline{\overline{\mathcal{I}}}_b} \sum_{i\in\overline{\mathcal{I}}_b} \big(\widehat{f}_b(\boldsymbol{x}_i) - y_i\big)^2$$

considérons une petite perturbation sur la variable j,

$$\widehat{R}_b^{oob}{}_{(j)} = \frac{1}{\overline{\overline{\mathcal{I}}}_b} \sum_{i\in\overline{\mathcal{I}}_b} \big(\widehat{f}_b(x_{i1},\cdots,x_{ij}+\varepsilon,\cdots,x_{ik}) - y_i\big)^2$$

et on définie

$$\text{importance}_j = \frac{1}{B}\sum_{b=1}^{B}\big(\widehat{R}_b^{oob}{}_{(j)} - \widehat{R}_b^{oob}\big)$$

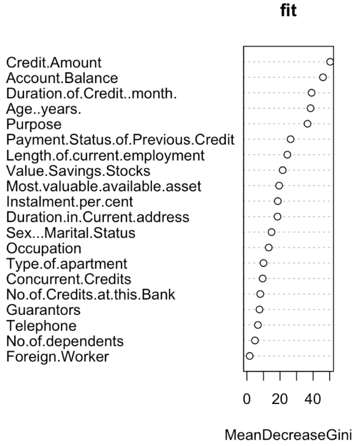
Random Forest

```
> credit = read.csv(url, header = TRUE, sep = ",")
> F=c(1,2,4,5,7,8,9,10,11,12,13,15,16,17,18,19,20)
> for(i in F) credit[,i]=as.factor(credit[,i])
> set.seed(123)
> i_test = sample(1:nrow(credit),size=333)
> i_train = (1:nrow(credit))[-i_test]
> > set.seed(456)
> library(randomForest)
> fit = randomForest(Creditability~., data = credit)
> print(fit)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 23.3%
Confusion matrix:
    0   1 class.error
0 121 179  0.59666667
1  54 646  0.07714286
```

Random Forest

```
> varImpPlot(fit)
> fit$importance
            MeanDecreaseGini
Account.Balance    45.94
Duration.of.Cre    39.09
Payment.Status.    26.38
Purpose            36.61
Credit.Amount      50.31
Value.Savings.S    21.62
Length.of.curre    24.42
Instalment.per.    18.66
Sex...Marital.S    14.87
Guarantors          7.62
Duration.in.Cur    18.48
Most.valuable.a    19.46
Age..years.        38.45
Concurrent.Cred     9.51
Type.of.apartme    10.01
No.of.Credits.a     8.05
Occupation         13.18
No.of.dependent     4.86
```

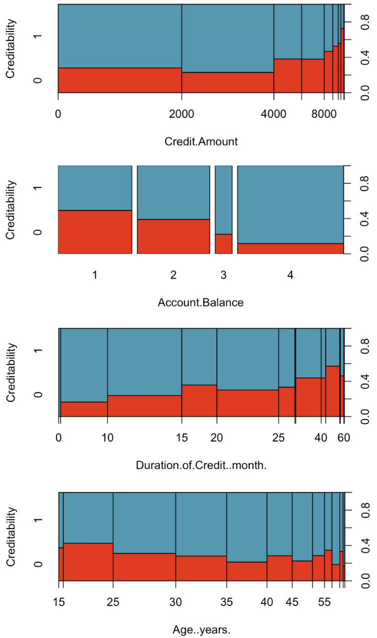


for variable importante

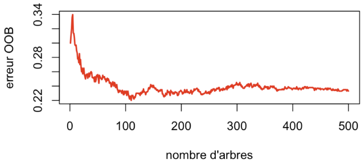
Random Forest

to visualize the impact of x_j's on y

```
> credit=read.csv(url, header
    = TRUE, sep = ",")
> plot(Creditability ~ Credit
   .Amount, data = credit)
> plot(Creditability ~
   Account.Balance, data =
   credit
> plot(Creditability ~
   Duration.of.Credit..month
   ., data = credit)
> plot(Creditability ~ Age..
   years., data = credit)
```



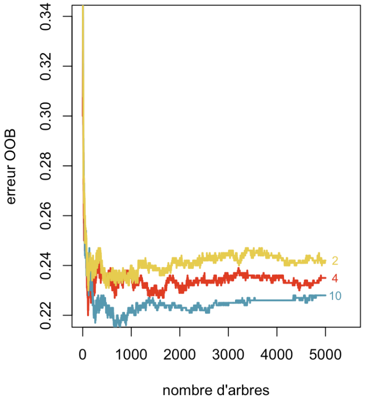
Random Forest



One can look at the out-of-bag error

```
> set.seed(456)
> fit = randomForest(
    Creditability ~ ., data =
    credit, ntree = 5000, mtry
     = 4)
> plot(fit$err.rate[, 1])
```

or with $m = 10$, or $m = 2$



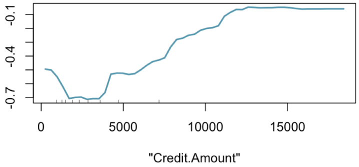
Random Forest

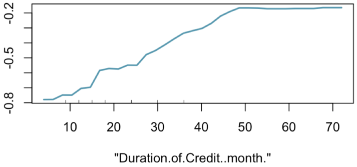
to visualize partial dependence plots

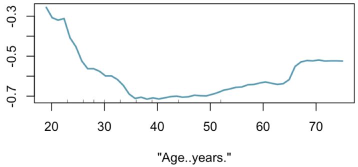
```
> lv = levels(
  credit$Creditability)
> partialPlot(fit, credit, "
  Credit.Amount", lv[1])
```

```
> set.seed(456)
> fit_train = randomForest(
  Creditability ~ ., data =
  credit[i_train,], ntree =
  500, mtry = 4)
> print(fit_train)

        OOB estimate of
  error rate: 23.54%
Confusion matrix:
   0   1 class.error
0 86 120   0.5825243
1 37 424   0.0802603
```



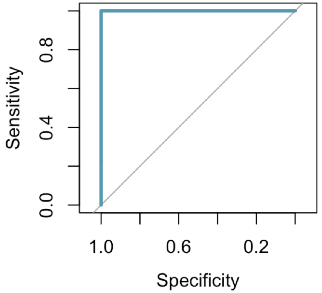


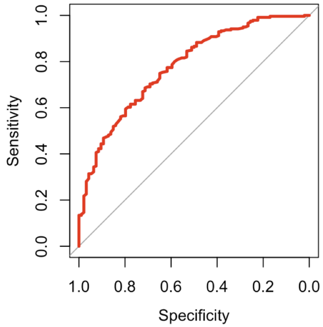


Random Forest

let us look at prediction from our random forest model

```
y = credit[i_train,"Creditability
    "]
y_test = predict(fit_train,type="
    prob",newdata=credit[i_train
    ,])[,2]
library(pROC)
roc_train = roc(y, y_test, plot=
    TRUE,col="blue")
y = credit[i_test,"Creditability"]
y_test = predict(fit_train,type="
    prob",newdata=credit[i_test,])
    [,2]
roc_test = roc(y, y_test, plot=
    TRUE,col="red")
```





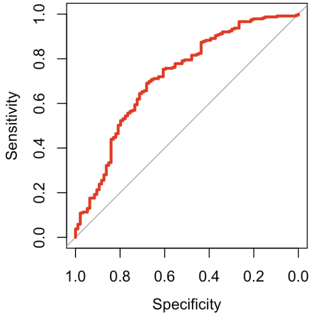
Random Forest

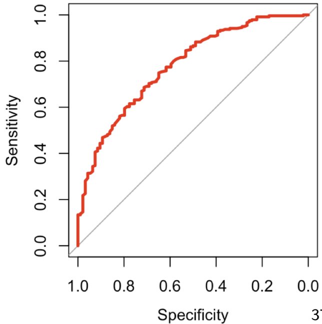
here, AUC is

```
> pROC::auc(roc_test)
Area under the curve: 0.7783
```

to be compared with a (basic) GLM

```
> glm_train <- glm(Creditability ~
     ., data = credit[i_train,],
    family=binomial)
> y = credit[i_test,"Creditability
    "]
> y_test = predict(glm_train,type
    ="response",newdata=credit[
    i_test,])
> roc_test <- roc(y, y_test, plot=
    TRUE,col=colr[2],lwd=3)
> pROC::auc(roc_test)
Area under the curve: 0.72
```


Sensitivity
Specificity


Sensitivity
Specificity