Scénario: Patrimoine et score d'appétence de l'assurance vie

Résumé

Cette aventure poursuit celle d'exploration des données de patrimoine. Il s'agit d'aborder systématiquement la construction de modèles de prévision de la probabilité de possession d'un contrat d'assurance vie ou score d'appétence. la plupart des méthodes sont abordées et comparées régression logistique, analyse discriminante.pdf, arbre de discrimination, agrégation de modèles, SVM. Les estimations des erreurs de prévision et les courbes ROC sont comparées à la suite du tirage itératif d'un ensemble d'échantillons tests.

1 Introduction

1.1 Objectif

Le travail proposé vient compléter le cours sur l'apprentissage statistique. Son objectif est double :

1. illustrer l'emploi de chacune des méthodes du cours sur un exemple en vraie grandeur mais relativement simple,
2. mettre en œuvre une procédure systématique de comparaison des erreurs de prévisions ainsi que des courbes ROC estimées sur un échantillon test par les différentes méthodes.

Il s'intéresse à un jeu de données décrivant le patrimoine des français (enquête INSEE) et se propose de comparer plusieurs méthodes de modélisation (régression logistique, réseaux de neurones, arbres de décision, agrégation de modèles) pour aborder un problème très classique en Gestion de la Relation Client (GRC / CRM) : construire un score d'appétence. Il s'agit du score d'appétence pour l'assurance vie mais ce pourrait être un score d'attrition (churn) d'un opérateur téléphonique ou encore un score de défaillance d'un emprunteur ou de faillite d'une entreprise ; les outils de modélisation sont les mêmes et sont très largement utilisés dans tout le secteur tertiaire pour l'aide à la décision.

1.2 Les données

Origine

Les données sont publiques et accessibles sur le site de l'INSEE. Elles sont issues d'une enquête "Patrimoine" réalisée en 2003-2004 qui faisait suite à des enquêtes "Actifs financiers" de 1986 et 1992. L'enquête "Patrimoine" visait une connaissance globale du patrimoine des ménages ainsi que de ses principales composantes. Elle recense tous les différents types d'actifs financiers, immobiliers et professionnels détenus par les individus ainsi que les différents types d'emprunts (immobilier, à la consommation, achat voiture, biens professionnels, ...). Outre la connaissance de souscriptions à différents types de produits, l'enquête interroge aussi les individus sur leur motivation de détention permettant ainsi d'appréhender le comportement patrimonial des ménages.

Attention : La structure des données est relativement complexe, il est vivement recommandé d'avoir réalisé le scénario précédent d'exploration des données, où elles sont précisément décrites, avant de se lancer dans le travail de modélisation qui va suivre.

Variables

Finalement, à l'issue du travail préliminaire et de l'exploration, l'étude présente débute avec une base contenant 11887 individus décrits par 36 variables dont 2 sont redondantes (qualitatives et quantitatives) :

- 2 quantitatives : `Age, Nbenf`,
- 34 qualitatives : `Asvi, Sexe, Tage, Couple, Vmatri, Diplome, Ocupa, Work, statut, Herit, Pere, Mere, Gparp, gparm, Jgrav, Livep, Epalo, fepsal, vmob, livdf, pel, cel, qpep, asdecv, Retrai, Qpea, Urbani, Zeat, Nbenfq, Iogoc, terre, dette, bdetre, hdetvo`.

définies dans le Tableau 1.

Une première conclusion de l'étude précédente montre qu'il aurait été sûrement beaucoup plus efficace, à même coût, d'interroger beaucoup moins de monde avec beaucoup moins de questions mais en prenant le temps d'obtenir des réponses précises à l'ensemble des questions ! C'est malheureusement

un comportement excessivement répandu dans beaucoup de disciplines, des Sciences humaines à la Biologie, de viser un niveau de détail beaucoup trop fin au regard de la précision des données ou de la taille de l'échantillon. Ainsi, les variables décrivant les ressources et le patrimoine n'ont même pas été prises en compte car elles présentent beaucoup trop de données manquantes : 90 % pour le patrimoine financier, 60 % pour l'autre ; même chose pour les variables relatives aux petits enfants. C'est la faiblesse majeure de cette enquête ; trop d'informations tue l'information.

L'absence de ces variables va pénaliser sévèrement la qualité des modèles mais c'est une réalité à laquelle il est nécessaire de se confronter.

Lecture des données

A l'issue de l'analyse exploratoire prémiminaire, les données sont disponibles dans une table SAS `sasuser.patriminsee`. La recherche de scores est réalisée dans R car ce logiciel offre beaucoup plus de possibilités dans le choix des méthodes, sans parler en plus des problèmes de coût surtout si le module Enterprise Miner est utilisé. Les données SAS ont donc été exportées dans un fichier au format "csv" par les commandes :

```
proc export data=sasdata.patriminsee
        outfile= "patriminsee.csv"
        dbms=csv replace;
```

qu'il suffit de télécharger du répertoire `http://wikistat.fr/data` avant de le lire dans R.

```
insee = read.csv("patriminsee.csv")
insee=insee[,-c(2,38)]
summary(insee)
```

1.3 Stratégie de construction d'un score

L'objectif est la recherche d'un meilleur modèle de prévision de la probabilité de possession d'un contrat d'assurance vie. Meilleur au sens de la minimisation d'une erreur de prévision estimée sur une partie tes de l'échantillon.

La démarche mise en œuvre enchaîne les étapes classiques suivantes :

1. Après l'étape descriptive uni ou multidimensionnelle visant à repérer les

TABLE 1 – Signification des variables retenues et liste des modalités

Identif.	Libellé	Modalités
Asvi	Possession ou non assurance vie	AsO, AsN
Sexe	Genre	Sh, Sf
Age	Age	Quantitatif
Tage	Tranches d'âge	T30 à T80
Couple	Vie ou non en couple	CouO CouN
Vmatri	Statut matrimonial	Vcel Vmar Vsep
Diplome	Niveau de diplôme	Dsan, Dcep, Dtec (cap, bep), Dbepc, Dbac Dbac, Db+2, Db+5
Occupa	Type d'occupation	Oact, Oina (foyer, chom, other), Oret
Work	Niveau professionnel	WctA (cadre, catA), WctB (agent, catB, tech), Wemp, WctC (osp, ouv)
statut	Statut professionnel	spri, spub, sind
Herit	Bénéfice ou non d'un héritage	HerO, HerN
Pere	Présence ou non du père	PerO PerN
Mere	Présence ou non de la mère	MerO, MerN
Gparp	Grands parents paternels	GppO GppN
gparm	Grands parents maternels	gpmO gpmN gpmI
Jgrav	Evènement grave dans la jeunesse	JgvO JgvN
Livep	Livret d'épargne	LivO LivN
Epalo	Epargne logement	EplO EplN
qpep	Plan d'épargne populaire	qppO qppN
vmob	Valeurs mobilières	vmoO vmoN
asdecv	Assurance décès volontaire	asdO asdN
Retrait	Epargne retraite	RepO RepN
livdf	Livret défiscalisé	ldfO, ldfN
pel	Plan épargne logement	pelO, pelN
cel	Compte épargne logement	celO, celN
fepsal	Epargne salarial ou stock options	fesO fesN
Qpea	Plan épargne action	QpeO QpeN
Urbani	Niveau d'urbanisation	U1 à U5
Zeat	Région de résidence	Zso Zpar Zoue Zne Zmed Zidf Zcen
Nbenf	Nombre d'enfants	Quantitatif
Nbenfq	Nombre d'enfants	Nbe0, Nbe1, Nbe2, Nb>3
Iogoc	Type d'occupation du logement	Iloc, Iprp (usufruit)
terre	Possession de terres	terO terN
dette	Dettes ou emprunts	detO detN
bdetre	Emprunt achat maison	bemO bemN
hdetvo	Emprunt voiture	hevO hevN

incohérences, les variables non significatives, les individus non concernés. . . et à étudier les structures des données, procéder à un tirage aléatoire d'un échantillon *test* qui ne sera utilisé que lors de la *dernière étape*.

2. Sur la partie restante qui sera découpée en échantillon d'*apprentissage (des paramètres du modèle) et échantillon de* validation (pour estimation sans biais du taux de mauvais classés), optimiser les choix afférents à chacune des méthodes de discrimination :
   - variables et interactions à prendre en compte dans la régression logistique,
   - nombre de nœuds dans l'arbre de classification,
   - architecture (nombre de couches, de neurones par couche, fonctions de transferts, nombre de cycles. . .) du perceptron.
   - algorithme d'agrégation de modèles
   - éventuellement d'autres méthodes...

   Remarques :
   - En cas d'échantillon petit face au nombre des variables il est recommandé d'itérer la procédure de découpage par validation croisée, ce n'est réellement le cas de ces données mais, afin d'estimer les variances d'erreur de classement ou leurs distributions, la procédure est itérée.
3. Comparaison finale des qualités de prévision et des courbes ROC sur la base du taux de mal classés pour le seul échantillon test qui est resté à l'écart de tout effort ou acharnement pour l'optimisation des modèles.

   Attention, ne pas "tricher" en modifiant le modèle obtenu lors de l'étape précédente afin d'améliorer le résultat sur l'échantillon test !

Deux stratégies ou approches sont mises en œuvre :

Artisanale à l'aide des méthodes les plus utilisées dans le contexte de la GRC : régression logistique très majoritaire et arbres de décision ; avec l'idée de chercher à interpréter les modèles.

Industrielle comparaison systématique et automatique de plusieurs modèles, dont réseaux de neurones, agrégation..., à l'aide du package `caret` (Kunh, 2008 [1]) ; la qualité de prévision prévaut.

2 Approche artisanale

2.1 Partitionnement de l'échantillon

Il s'agit d'isoler un *échantillon test* qui ne servira qu'à la seule comparaison des méthodes mais pas à l'estimation ni au choix de modèle inhérent à chacune d'elles. La séparation entre échantillon d'apprentissage et échantillon test se fait de la façon ci-dessous après initialisation du générateur de nombres aléatoires afin d'en contrôler la séquence pour d'autres exécutions. Il est vivement recommandé de conserver dans un fichier la liste des commandes R exécutées. Cela permettra ainsi de les ré-exécuter facilement pour une autre valeur de l'initialisation du générateur de nombres aléatoires.

```
set.seed(xxx) # remplacer xxx par un entier
npop=nrow(insee)
# tirage de 3000 indices sans remise
testi=sample(1:npop,3000)
#Liste des indices restant qui n'ont pas été tirés
appri=setdiff(1:npop,testi)
# Extraction échantillon d'apprentissage
inseeAppt=insee[appri,]
# Extraction échantillon de test
inseeTest=insee[testi,]
summary(inseeAppt) # vérifications
summary(inseeTest)
```

2.2 Régression logistique

Il est proposé dans le scénario d'estimation de score de la carte visa premier de comparer les résultats obtenus avec SAS et R. Ici seul R est utilisé, se reporter à cet autre scénario pour connaître les syntaxes SAS.

La régression logistique est une adaptation du modèle linéaire en vue de la prédiction d'une variable (binomiale) à deux modalités correspondant bien à l'objectif recherché.

Attention, les instructions ci-dessous sont données à titre indicatif ; en effet, comme chaque groupe dispose d'un échantillon différent, les sélections et listes de variables doivent être adaptées à la situation rencontrée.

La régression logistique est estimée dans R un considérant un cas particulier de modèle linéaire général (fonction `glm`) de densité binomiale et de fonction

lien la fonction lien canonique (logistique).

```
# Estimation du modèle complet sans interaction
res.logit=glm(Asvi~.,data=inseeAppt,family=binomial)
# tests de nullité des coefficients
anova(res.logit,test="Chisq")
```

Commentaires sur les p-valeurs.

Choix de modèle

Une sélection de variables s'impose en utilisant une procédure automatique. Compte tenu du nombre raisonnable de variables, plusieurs stratégies peuvent être envisagées par minimisation du critère d'Akaïke (AIC). Noter bien que le critère utilisé par cette fonction de R est plus prédictif qu'explicatif.

Voici celle descendante :

```
res.logit=glm(Asvi~.,data=inseeAppt,family=binomial)
res.step=step(res.logit)
# variables du modèles
anova(res.step,test="Chisq")
```

Celle pas à pas :

```
res2.logit=glm(Asvi~1,data=inseeAppt,family=binomial)
res2.step<-step(res2.logit,direction="both",
  scope=list(lower=~1, upper=~Sexe+Age+Tage+Couple+
  Vmatri+Diplome+Occupa+Work+statut+Herit+Pere+Mere+
  Gparp+gparm+Jgrav+Livep+Epalo+fepsal+vmob+livdf+
  pel+cel+qpep+asdecv+Retrait+Qpea+Urbani+Zeat+
  Nbenf+Nbenfq+Iogoc+terre+dette+bdetre+hdetvo))
# observer les p-values
anova(res2.step,test="Chisq")
```

Comparer les deux modèles.

On peut par ailleurs se poser la question de la nécessité ou non de prendre en compte des interactions.

```
res3.logit=glm(Asvi~1,data=inseeAppt,family=binomial)
res3.step=step(res3.logit,direction="both",
  scope=list(lower=~1,upper=~(Sexe+Tage+Work+Herit+
  Epalo+fepsal+vmob+livdf+pel+cel+qpep+Retrait+Qpea+
  Urbani+Zeat+Nbenfq+Iogoc+bdetre)**2))
# observer les p-values
anova(res2.step,test="Chisq")
```

Rappel, chaque groupe dispose d'un échantillon d'apprentissage différent dépendant de l'initialisation du générateur. Les "meilleurs" modèles peuvent donc différer d'un groupe à l'autre et par rapport à ce qui est présenté dans ce texte.

Les modèles obtenus peuvent différer et certaines variables présenter des p-values non significatives. Le modèle proposé par minimisation du critère AIC est peut-être trop complexe. Il s'agit donc de comparer les valeurs prédictives de ces modèles ou de sous modèles par validation croisée en retirant les variables les moins significatives une à une. La bibliothèque `boot` propose une procédure de validation croisée adaptée à la fonction `glm`.

```
library(boot)
# premier modele
res.logit=glm(Asvi~Sexe+Tage+ Work+Herit+fepsal+vmob+
  livdf+pel+cel+qpep+Retrait+Qpea+Urbani+Zeat+Nbenfq+
  Iogoc+bdetre,data=inseeAppt,family=binomial)
cv.glm(inseeAppt,res.logit,K=10)$delta[1]
# deuxieme modele
res2.logit=glm(Asvi ~ vmob+Tage+Epalo+Work+livdf+
   Herit+qpep+bdetre+Iogoc+Nbenfq+fepsal+Sexe+Qpea+
   Zeat+Urbani+Retrait+pel,data=inseeAppt,
   family=binomial)
cv.glm(inseeAppt,res2.logit,K=10)$delta[1]
# troisième modele avec interactions
res3.logit=glm(Asvi ~ vmob+Tage+Epalo+Work+livdf+
   Herit+qpep+bdetre+Iogoc+fepsal+Nbenfq+Sexe+
   Qpea+Zeat+Urbani+Retrait+pel+cel+vmob:Tage+
   Tage:Iogoc+Work:Iogoc+Work:Herit+qpep:Qpea+
   vmob:Epalo+livdf:Nbenfq+livdf:Urbani+
   qpep:Iogoc+vmob:fepsal+bdetre:pel+Nbenfq:pel+
   Zeat:pel+Tage:cel+Sexe:Urbani+bdetre:Sexe+
```

```
    bdetre:fepsal+Nbenfq:Retrait,data=inseeAppt,
    family=binomial)
cv.glm(inseeAppt,res3.logit,K=10)$delta[1]
```

Retenir le meilleur modèle.

Prévision de l'échantillon test

Les commandes ci-dessous calculent la prévision de la variable `Asvi` pour l'échantillon test et croisent la variable prédite avec la variable observée afin de construire la matrice de confusion et estimer le taux d'erreur avec une probabilité seuil de 0,5.

```
pred.logit=predict(res3.logit,newdata=inseeTest)>0.5
table(pred.logit,inseeTest$Asvi=="AsO")
```

Noter le taux d'erreur et le grand optimisme de la validation croisée.

Construction de la courbe ROC

Le tracer de cette courbe fait varier le seuil de la probabilité pour évaluer comment se comporte les caractéristiques de sensibilité et spécificité du modèle.

```
library(ROCR)
roclogit=predict(res2.logit, newdata=inseeTest,
   type="response")
predlogistic=prediction(roclogit,inseeTest$Asvi)
perflogistic=performance(predlogistic, "tpr","fpr")
plot(perflogistic,col=1)
```

2.3 Arbres de décision binaires

Cette section a pour objectif la construction d'un arbre de discrimination (classification tree) et son élagage par validation croisée pour optimiser sa capacité de discrimination.

Estimation

Deux librairies, `tree` et `rpart`, proposent les techniques CART avec des algorithmes analogues à ceux développés dans Splus mais moins de fonctionnalités ; `rpart` fournissant des graphes plus explicites et une procédure d'élagage plus performante est préférée.

La première estimation favorise un arbre très détaillé c'est-à-dire avec un très faible coefficient de pénalisation de la complexité de l'arbre et donc du nombre de feuilles. Le critère d'hétérogénéité est l'entropie.

```
library(rpart) # Chargement de la librairie
res.tree=rpart(Asvi~.,data=inseeAppt,
   parms=list(split='information'),cp=0.001)
summary(res.tree) # Quelques renseignements
plot(res.tree) # Tracé de l'arbre
text(res.tree) # Ajout des légendes des noeuds
```

Il est évident que l'arbre présente trop de feuilles. Une première façon de l'élaguer consiste à tracer la décroissance de l'erreur relative en fonction du coefficient de complexité c'est-à-dire aussi en fonction de la taille de l'arbre ou nombre de feuilles. La procédure est mal explicitée dans la documentation.

```
plotcp(res.tree)
```

Il faut alors choisir une valeur du coefficient de pénalisation de la complexité pour élaguer l'arbre. Attention, ce choix dépend de l'échantillon tiré. Il peut être différent de celui ci-dessous. La valeur de cp choisie correspond à la première valeur d"erreur relative" sous les pointillés.

```
res2.tree=prune(res.tree,cp=0.005381605)
plot(res2.tree)
text(res2.tree,use.n=TRUE)
```

Élagage par validation croisée "explicite"

La commande suivante calcule les prédictions obtenues pas 10-fold validation croisée pour chaque arbre élagué suivant les valeurs du coefficients de complexité. La séquence de ces valeurs doit être adaptée à l'exemple traité.

```
res.tree=rpart(Asvi~.,data=inseeAppt,
   parms=list(split='information'),
         control = rpart.control(cp = .001))
xmat = xpred.rpart(res.tree,xval=10,
```

```
   cp=seq(0.03,0.001,length=10))
# Comparaison de la valeur prédite
# avec la valeur observée.
xerr=as.integer(inseeAppt$Asvi)!= xmat
# Calcul et affichage des estimations
# des taux d'erreur
apply(xerr, 2, sum)/nrow(xerr)
```

Le choix est-il confirmé ?

```
# Elagage puis affichage de l'arbre
res3.tree=prune(res.tree,cp=0.007781788)
plot(res3.tree)
text(res3.tree,use.n=TRUE)
```

La commande suivante produit un graphique plus élaboré et ainsi plus facile à lire. L'arbre est généré dans un fichier au format postscript et rangé dans le répertoire courant.

```
post(res3.tree)
```

Interpréter les arbres. Que dire du choix des variables comparativement aux autres méthodes ?

Prévision de l'échantillon test

Les commandes ci-dessous calculent la prévision de la variable `Asvi` pour l'échantillon test et croisent la variable prédite avec la variable observée afin de construire les matrices de confusion et donc d'estimer les taux d'erreur.

```
pred.tree=predict(res3.tree,
   newdata=inseeTest,type="class")
table(pred.tree,inseeTest$Asvi)
```

Noter et comparer les taux d'erreur ainsi que les courbes ROC :

```
ROCrpart=predict(res3.tree,newdata=inseeTest,
   type="prob")[,2]
predrpart=prediction(ROCrpart,inseeTest$Asvi)
perfrpart=performance(predrpart,"tpr","fpr")
# tracer les courbes ROC
plot(perflogistic,col=1)
# en les superposant pour mieux comparer
plot(perfrpart,col=2,add=TRUE)
```

3 Approche industrielle

3.1 Tests de différents modèles

L'étude se limite à une utilisation de la librairie `caret` (Kunh, 2008 [1]) pour quelques unes des méthodes de modélisation proposées. C'est un "méta-package" qui englobe dans une seule fonction `train` l'appel des différentes fonctions d'estimation d'autre packages. La commande `library` est incluse mais évidemment les packages utilisés doivent avoir été chargés.

3.2 Préparation des données

Extraction des échantillons d'apprentissage et de test. Attention, le mode d'utilisation ou d'appel de la principale fonction : `train` de `caret` dépend du type des variables explicatives et de la méthode utilisée ; c'est une limitation de l'aspect "généralisation" de ce méta-package. Pas de problème lorsque les variables X^j sont toutes quantitatives mais des traitements différents des variables X^j qualitatives. Les facteurs sont, ou non, transformés en *dummy* variables c'est-à-dire en indicatrices : une par modalité pour certaines méthodes. Ce n'est pas une procédure "optimale" en sélection de variables au moment de l'interprétation !

Dans le premier cas, tout quantitatif, il faut séparer les variables `X` de la variable cible à expliquer `Y`. Dans le 2ème, variables qualitatives, et pour certaines méthodes, il faut utiliser la syntaxe classique en R de définition d'un modèle (*formulæ*) pour éviter un message d'erreur.

La constitution de l'échantillon test est similaire au traitement précédent avec une proportion identique.

```
library(caret)
# indices de l'échantillon d'apprentissage
set.seed(xxx)
inTrain = createDataPartition(insee[,1],
```

```
   p = .7476, list = FALSE)
# Extraction des échantillons
# data frames apprentissage et test
inseeAppt=insee[inTrain,]
inseeTest=insee[-inTrain,]
```

Il est recommandé de centrer et réduire les variables quantitatives dans plusieurs méthodes. C'est fait systématiquement sauf dans le cas présent où les variables sont essentiellement qualitatives.

```
# estimation des erreurs par validation croisée
cvControl=trainControl(method="cv",number=10)
```

3.3 Estimation et optimisation des modèles

La librairie intègre beaucoup plus de méthodes que celles utilisées. Consulter la liste des méthodes disponibles dans l'aide de la fonction : `?train` pour en tester d'autres. Exécuter successivement les blocs de commandes pour tracer séparamment chacun des graphes afin de contrôler le bon comportement de l'optimisation du paramètre ou des paramètres de complexité de chaque modèle.

L'utilisation de certaines méthodes, comme la régression logistique, est sans doute moins flexible qu'en utilisation "manuelle" en particulier dans le choix de l'algorithme de sélection de variables. Il faut se montrer (très) patient pour certaines optimisations alors que d'autres sont immédiates, voire inutiles. Le paramètre `tuneLength` caractérise un "effort" d'optimisation, c'est en gros le nombre de valeurs de paramètres testées sur une grille fixée automatiquement. En prenant plus de soin et aussi plus de temps, il est possible de fixer précisément des grilles pour les valeurs du ou des paramètres optimisés pour chaque méthode.

```
#1 Régression logistique par sélection backward
# de toutes les indicatrices et CV ... c'est long
# pas d'autre paramètre à optimiser
glmFit = train(Asvi~.,data=inseeAppt,
    method = "glmStepAIC",trControl = cvControl)
glmFit
#2 Réseaux de neurones
nnetFit = train(Asvi~.,data=inseeAppt,
    method = "nnet", tuneLength = 6,
   trControl = cvControl)
nnetFit
plot(nnetFit)
#3 Arbre de décision
set.seed(2)
rpartFit = train(Asvi~.,data=inseeAppt,
   method = "rpart", tuneLength = 10,
   trControl = cvControl)
rpartFit
plot(rpartFit)
#4 Random forest
set.seed(2)
rfFit = train(Asvi~.,data=inseeAppt,
   method = "rf", tuneLength = 4,
   trControl = cvControl)
rfFit
plot(rfFit)
#5 boosting
set.seed(2)
gbmFit = train(Asvi~.,data=inseeAppt,
   method = "gbm", tuneLength = 4,
   trControl = cvControl)
gbmFit
plot(gbmFit)
#6 boosting 2
set.seed(2)
c50Fit = train(Asvi~.,data=inseeAppt,
   method = "C5.0", tuneLength = 4,
   trControl = cvControl)
c50Fit
plot(c50Fit)
```

Remarque : plusieurs méthodes comme k-nn, les SVM, l'analyse discriminante ne sont pas exécutables directement du fait du caractère qualitatif des variables explicatives.

3.4 Prévisions et erreurs en test

Les méthodes sélectionnées et optimisées sont ensuite appliquées à la prévision de l'échantillon test. Estimation du taux de bien classés :

```
models=list(rlog=glmFit,nnet=nnetFit,rtree=rpartFit,
   rf=rfFit,gbm=gbmFit,C50=c50Fit)
testPred=predict(models, newdata = inseeTest)
# taux de bien classés
lapply(testPred,function(x)mean(x==testY))
```

Tracer les courbes ROC pour analyser spécificité et sensibilité.

```
# Courbes ROC
library(ROCR)
testProb=predict(models, newdata = inseeTest,
   type="prob")
predroc=lapply(testProb,
   function(x)prediction(x[,1],
         inseeTest[,"Asvi"]=="AsN"))
perfroc=lapply(predroc,
   function(x)performance(x, "tpr", "fpr"))
# Tracer les courbes ROC
plot(perfroc$rlog,col=1)
plot(perfroc$nnet,col=2,add=TRUE)
plot(perfroc$rtree,col=3,add=TRUE)
plot(perfroc$rf,col=4,add=TRUE)
plot(perfroc$gbm,col=5,add=TRUE)
plot(perfroc$C50,col=6,add=TRUE)
legend("bottomright",legend=c("rlog","nnet",
  "Tree","RF","gbm","C50"),col=c(1:6),pch="_")
```

Comparer, commenter. Notez qu'il n'y a pas, en général, de méthode uniformément meilleure ; l'aire sous la courbe ROC (AUC) ne génère pas un ordre total.

Importance des variables

Certaines méthodes sont très peu explicites car de véritables "boîtes noires", quant au rôle et impact des variables sur la prévision des observations. Néanmoins, des indicateurs d'importance sont proposés pour les forêts aléatoires.

```
rfFit2=randomForest(inseeAppt[,-1],
   inseeAppt[,"Asvi"],importance=T)
imp.mdrr=sort(rfFit2$importance[,3],
   decreasing=T)[1:20]
par(xaxt="n")
plot(imp.mdrr,type="h",ylab="Importance",
   xlab="Variables")
points(imp.mdrr,pch=20)
par(xaxt="s")
axis(1,at=1:20,labels=names(imp.mdrr),
   cex.axis=0.8,las=3)
```

3.5 Automatisation

L'échantillon n'est pas de faible taille, mais les estimations des taux de bien classés comme le tracé des courbes ROC sont très dépendants de l'échantillon test ; on peut s'interroger sur l'identité du modèle le plus performant comme sur la réalité des différences entre les méthodes. Il est alors important d'itérer le processus sur plusieurs échantillons tests.

Exécuter, plutôt la nuit :-) car c'est long, la fonction en annexe en choisissant les méthodes semblant les plus performantes :

```
pred.insee=pred.autom(Asvi~.,data=insee,p = .7476,
   methodes=c("glmStepAIC","nnet","rf","gbm"),N=10,
   size=c(4,4,4,4),type="prob")
```

Puis calculer :

```
# Taux de bien classés
obs=pred.insee$obs
prev.insee=pred.insee$pred
res.insee=lapply(prev.insee,function(x)apply((x>0.5)
   ==(obs==1),2,mean))
# Moyennes des taux de bien classés par méthode
lapply(res.insee,mean)
# Distributions des taux de bien classés
```

```
boxplot(data.frame(res.insee))
# Tracer les courbes ROC moyennes
predroc.insee=lapply(prev.insee,
   function(x)prediction(x,obs==1))
perfroc.insee=lapply(predroc.insee,
   function(x)performance(x,"tpr","fpr"))
plot(perfroc.insee$glmStepAIC,col=1,lwd=1.5,
   avg="vertical")
plot(perfroc.insee$nnet,col=2,add=TRUE,
   lwd=1.5,avg="vertical")
plot(perfroc.insee$rf,col=3,add=TRUE,
   lwd=1.5,avg="vertical")
plot(perfroc.insee$gbm,add=TRUE,col=4,lwd=1.5,
   avg="vertical")
legend("bottomright",legend=c("rlogit","nnet",
   "RF", "boost"),col=c(1:4),pch="_")
```

Commenter les résultats, peut-on conseiller une méthode ? Que dire du nombre très important de méthodes proposées au regard de ces résultats ?

Attention
- Ne pas généraliser sur la base de ces seuls résultats,
- l'apparente interprétabilité de la régression logistique est assez illusoire compte tenu du nombre de variables indicatrices introduites dans le modèle.

Références

[1] Max Kuhn, *Building Predictive Models in R Using the caret Package*, Journal of Statistical Software **28** (2008), n° 5.

Annexe : prévision de N échantilllons test

```
pred.autom=function(formul,data,p=1/2,
   methodes=c("knn","rf"),size=c(10,2),
   xinit=11,N=10,typerr="cv",number=4,type="raw")
# Fonction de prévision de N échantillons tests
# par une liste de méthodes de régression
# ou classification (uniquement 2 classes)
# Optimisation des paramètres par validation
# croisée (défaut) ou bootstrap ou... (cf. caret)
# data : data frame avec la variable cible Y
# en première colonne
# Formule sous la forme Y~.
# p : proportion entre apprentissage et test
# methodes : liste des méthodes de discrimination
# size : e grille des paramètres à optimiser
# xinit : générateur de nombres aléatoires
# N : nombre de réplications apprentissage / test
# typerr : "cv" ou "boo" ou "oob"
# number : nombre de répétitions CV ou bootstrap
# pred : liste des matrices de prévision
{# type d'erreur
Control=trainControl(method=typerr,number=number)
# initialisation du générateur
set.seed(xinit)
# liste de matrices stockant les prévisions
# une par méthode
inTrain=createDataPartition(data[,1],p=p,list=FALSE)
ntest=length(data[-inTrain,1])
pred=vector("list",length(methodes))
names(pred)=methodes
pred=lapply(pred,function(x)x=matrix(0,
   nrow=ntest,ncol=N))
obs=matrix(0,ntest,N)
set.seed(xinit)
for(i in 1:N)  # N itérations
{# indices de l'échantillon d'apprentissage
 inTrain=createDataPartition(data[,1],p=p,list=FALSE)
# Extraction des échantillons
Appt=data[inTrain,]
Test=data[-inTrain,]
# stockage des observés de testY
```

```
obs[,i]=Test[,1]
# estimation et optimisation des modèles
# pour chaque méthode de la liste
for(j in 1:length(methodes))
{# modélisation
modFit = train(formul,data=Appt,
   method = methodes[j], tuneLength = size[j],
   trControl = Control)
# prévisions
if (type=="prob")  pred[[j]][,i]=predict(modFit,
    newdata = Test,type=type)[,1]
else pred[[j]][,i]=predict(modFit,newdata=Test)
}}
list(pred=pred,obs=obs)} # résultats
```