Scénario: Score d'appétence de la carte visa premier

Résumé

Cette aventure reprend rapidement l'exploration des données bancaires avant d'aborder systématiquement la construction de modèles de prévision de la probabilité de possession d'une carte visa premier ou score d'appétence. la plupart des méthodes sont abordées comparées régression logistique, analyse discriminante.pdf, arbre de discrimination, agrégation de modèles, .pdfSVM. 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 Présentation

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 bancaires et se propose de comparer plusieurs méthodes de modélisation (régression logistique, analyse discriminante, réseaux de neurones, arbres de décision, SVM, agrégation de modèles) pour aborder un problème très classique en Gestion de la Relation Client : construire un score d'appétence. Il s'agit du score d'appétence de la carte Visa Premier 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.

2 Prise en compte des données

Cette partie n'est qu'un rappel de la description des données et des principaux traitements réalisés dans le scénario d'exploration de ces données. Il ne vise qu'à illustrer la plus ou moins bonne séparation des deux classes de la variable "possession de la carte VP".

2.1 Description des données

Les variables

La liste des variables est issue d'une base de données retraçant l'historique mensuel bancaire et les caractéristiques de tous les clients. Un sondage a été réalisé afin d'alléger les traitements ainsi qu'une première sélection de variables. Les variables contenues dans le fichier initial sont décrites dans le tableau ci-dessous. Elles sont observées sur 1425 clients.

2.2 Exploration dans SAS

Les donnés sont disponibles dans le répertoire "data" tandis que les programmes sont accessibles dans ce répertoire.

Lecture des données

Importer les données ainsi que les trois programmes SAS de lecture, transformation et recodage des données. Exécuter le premier programme de lecture.

2.3 Statistique élémentaire

Ces données, nécessitent tout un travail préliminaire d'exploration et de "nettoyage" conduisant à des suppressions, corrections et transformations. Ces pré-traitements sont pris en compte dans les programmes SAS de transformation et lecture.

En bref,

- certaines observations sont supprimées : clients de moins de 18 ou plus de 65 ans, comptes professionnels, interdits bancaires ;
- des variables sont rectifiées : ancienneté de la relation, nombre de cartes, nombre d'opérations par carte pour assurer un minimum de cohérence ;
- les variables sont transformées : logarithme des variables de revenu ou

TABLE 1 – Liste des variables et de leur libellé

Identif.	Libellé
matric	Matricule (identifiant client)
depts	Département de résidence
pvs	Point de vente
sexeq	Sexe (qualitatif)
ager	Age en années
famiq	Situation familiale (Fmar : marié, Fcel : célibataire, Fdiv :divorcé, Fuli :union libre, Fsep : séparé de corps, Fveu :veuf)
relat	Ancienneté de relation en mois
pcspq	Catégorie socio-professionnelle (code num)
quals	Code "qualité" client évalué par la banque
GxxGxxS	plusieurs variables caractérisant les interdits bancaires
impnbs	Nombre d'impayés en cours
rejets	Montant total des rejets en francs
opgnb	Nombre d'opérations par guichet dans le mois
moyrv	Moyenne des mouvements nets créditeurs des 3 mois en Kf
tavep	Total des avoirs épargne monétaire en francs
endet	Taux d'endettement
gaget	Total des engagements en francs
gagec	Total des engagements court terme en francs
gagem	Total des engagements moyen terme en francs
kvunb	Nombre de comptes à vue
qsmoy	Moyenne des soldes moyens sur 3 mois
qcred	Moyenne des mouvements créditeurs en Kf
dmvtp	Age du dernier mouvement (en jours)

TABLE 2 – Liste des variables et de leur libellé — suite

Identif.	Libellé
boppn	Nombre d'opérations à M-1
facan	Montant facturé dans l'année en francs
lgagt	Engagement long terme
vienb	Nombre de produits contrats vie
viemt	Montant des produits contrats vie en francs
uemnb	Nombre de produits épargne monétaire
uemmts	Montant des produits d'épargne monétaire en francs
xlgnb	Nombre de produits d'épargne logement
xlgmt	Montant des produits d'épargne logement en francs
ylvnb	Nombre de comptes sur livret
ylvmt	Montant des comptes sur livret en francs
nbelts	Nombre de produits d'épargne long terme
mtelts	Montant des produits d'épargne long terme en francs
nbcats	Nombre de produits épargne à terme
mtcats	Montant des produits épargne à terme
nbbecs	Nombre de produits bons et certificats
mtbecs	Montant des produits bons et certificats en francs
rocnb	Nombre de paiements par carte bancaire à M-1
ntcas	Nombre total de cartes
nptag	Nombre de cartes point argent
segv2s	Segmentation version 2
itavc	Total des avoirs sur tous les comptes
havef	Total des avoirs épargne financière en francs
jnbjd1s	Nombre de jours à débit à M
jnbjd2s	Nombre de jours à débit à M-1
jnbjd3s	Nombre de jours à débit à M-2
carvp	**Possession de la carte VISA Premier**

de solde des comptes, découpage en classes des variables quantitatives, codage en indicatrice (0,1) des variables binaires. L'idée est de faire apparaître les variables sous deux formes : l'une quantitative de distribution si possible "symétrique", l'autre qualitative avec des modalités d'effectifs équilibrés (bornes fixées aux quantiles). Cette démarche permet ensuite d'utiliser le type de variable (quantitatif ou qualitatif) le plus adapté à une méthode donnée de modélisation.

Analyse en composantes principales

Exécuter le programme de transformation des données pour la suite des opérations. Toujours avec le module d'analyse interactive (ouvrir la table `sasuser.vispremt`), réaliser l'ACP centrée réduite de toutes les variables quantitatives et représenter le diagramme boîte des composantes principales ainsi que le premier plan factoriel (biplot) ; choix du nombre d'axes, interprétation de ces axes.

Colorier les points (individus) en fonction des modalités de la variable CARVP. Que pouvez vous dire d'une discrimination entre ces deux modalités sur la base d'un modèle linéaire des variables quantitatives ?

Analyse factorielle multiple de correspondances

Exécuter le programme de recodage des données pour la suite des opérations. Avec les macros commandes réaliser l'AFCM de toutes les variables qualitatives incluant les quantitatives recodées en classes, interprétation des axes.

```
proc corresp data=sasuser.vispremv observed
   out=resul mca dim=8;
tables famiq sexeq pcspq kvunbq vienbq
       uemnbq xlgnbq ylvnbq zocnbq
       nptagq carvp endetq gagetq facanq
       lgagtq havefq ageq relatq qsmoyq
       opgnbq moyrvq dmvtpq boppnq itavcq
       jnbjdq tavepq;
run;
%gafcix;          /* plan principal 1 x 2 */
%gafcix(x=1,y=3); /* plan principal 1 x 3 */
```

Combien d'axe retiendriez-vous ? Sont-ils faciles à interpréter ? Montrer que l'interprétation des deux premiers axes est similaire à celle de l'ACP.

Pour obtenir une représentation simultanée des individus, le programme ci-dessous calcule l'équivalent de l'AFCM par AFC du tableau disjonctif complet (attention à la virgule). Il fournit donc des coordonnées pour les individus. Elles permettent à ce niveau d'en contrôler la dispersion.

```
proc corresp data=sasuser.vispremv  out=resul
   dim=8;
tables matric,famiq sexeq pcspq kvunbq vienbq
       uemnbq xlgnbq ylvnbq zocnbq
       nptagq carvp endetq gagetq facanq
       lgagtq havefq ageq relatq qsmoyq
       opgnbq moyrvq dmvtpq boppnq itavcq
       jnbjdq tavepq;
run;
```

La même macro permet la représentation graphique mais la coloration des individus dépendant de leur matricule n'a aucun intérêt. Pour qu'elle dépende de la possession de la carte Visa premier un petit traitement est nécessaire : fusion selon le matricule de la table initiale avec celle des résultats après tri. La variable `carvpr` recodée sert alors d'identificateur et la taille des points est paramétrée (`tp=1`). Ce type de gymnastique demande un peu de compétences avec SAS et s'avère très utile.

```
proc sort data=sasuser.vispremv out=vispremr;
by matric;
proc sort data=resul out=resul;
by _name_;
data resul;
merge vispremr (keep=matric carvp
      rename=(matric=_name_)) resul ;
by _name_;
select(carvp);
when('Coui') _name_= '0000';
when('Cnon') _name_= '1111';
otherwise;
end;run;
%gafcix(tp=1);
%gafcix(x=1,y=3,tp=1);
```

La discrimination des deux classes de CARVP vous semble-t-elle plus réalisable ?

3 Construction des échantillons apprentissage et test dans R

3.1 Introduction

La tâche de partitionnement de l'échantillon est un préalable à la série des séances principalement axées sur l'objectif de discrimination appliqué aux données bancaires de GRC : prévoir la possession de la carte visa premier, ce qui revient à construire un score d'appétence. On se propose de tester différentes méthodes : régression logistique, analyse discriminante, réseau de neurones, arbre de décision, agrégation d'arbres, SVM. L'objectif final, à ne pas perdre de vue, sera la comparaison de ces méthodes afin de déterminer la plus efficace pour répondre au problème de prévision. Ceci passe par la mise en place d'un protocole très strict afin de s'assurer d'un minimum d'objectivité pour cette comparaison.

Les trois programmes SAS (lecture, transformation, codage) ainsi que ce travail de construction des échantillons doivent avoir été exécutés pour disposer des bons fichiers nécessaires à la poursuite les séances.

Penser à conserver dans des fichiers les commandes successives utilisées ainsi que les principaux résultats tableaux et graphes pour anticiper la rédaction du rapport.

3.2 Lecture, transformations et codage dans R

Importer les données spécifiques pour R ainsi que les trois programmes R qui réalisent les mêmes fonctions (lecture, transformation, recodage) que les programmes SAS. Exécuter successivement dans R les trois programmes afin de générer les bons fichiers. Attention, selon l'environnement de travail (windows, unix), il peut être nécessaire d'adapter le chemin d'accès aux fichiers dans le corps du programme de lecture. La partition de l'échantillon aurait tout aussi facilement pu se faire dans SAS avec la procédure `survey` d'échantillonnage. Le parti a été pris ici de vouloir comparer les méthodes entre elles même si elles sont exécutées par deux logiciels différents. La comparaison, n'est possible qu'à la condition d'utiliser les mêmes échantillons apprentissage et test dans SAS et R. D'où cette nécessité de construire la partition avec l'un des logiciels (R) avant de transférer les données sur l'autre logiciel. D'autre part, la partie exploratoire est plus efficace avec les outils de SAS tandis que certaines méthodes ne sont facilement accessibles et efficacement implémentées que dans R.

```
source("visa_lec.R")
source("visa_trans.R")
source("visa_code.R")
```

Ils construisent un objet `vispremv` de type "data frame" avec la structure suivante :

- 30 variables quantitatives,
- 24 variables qualitatives,
- la variable `CARVP` à prédire.

Vérifications :

```
summary(vispremv)
# variable contenant la liste des noms de variables
var=names(vispremv)
# liste des variables explicatives quantitatives
varquant=var[1:30]
# liste des variables explicatives qualitatives
varqual=var[31:55]
```

Remarques :

Comme annoncé ci-dessus, la plupart des variables apparaissent deux fois : sous forme quantitative ou qualitative. Ainsi, la variable sexe est soit de type facteur, soit une variable quantitative indicatrice (0,1), la variable âge est quantitative ou qualitative découpée en 3 classes. Ceci permet de s'adapter facilement à certaines méthodes qui n'acceptent que des variables explicatives quantitatives. Néanmoins, le choix de découper en classe une variable quantitative peut avoir un impact important sur la qualité de la prévision :

- le nombre de degrés de liberté diminue avec le nombre de modalités donc la qualité de l'estimation peut se dégrader pour de petits échantillons.
- le découpage en classes intervient en provoquant une approximation d'une transformation *non-linéaire* de la variable par une fonction étagée.

Le modèle construit s'en trouve donc plus flexible. Cela peut être un avantage (meilleure ajustement) comme un inconvénient (sur apprentissage). Il faut tester, comparer les qualités de prévision.

Ces remarques sont importantes. En effet, en fonction de la méthode utilisée, des effectifs, cela conduira à privilégier des variables quantitatives ou leur version qualitative découpée en classes.

3.3 Rappel : protocole de comparaison

La démarche mise en œuvre enchaîne les étapes classiques suivantes :

1. Après l'étape descriptive uni ou multidimensionnelle visant à repérer les 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 méthode pour l'analyse discriminante,
   - 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
   - noyau et complexité des SVM

   Remarques :
   - En cas d'échantillon petit face au nombre des variables il est recommandé d'itérer la procédure de découpage par validation croisée, c'est le cas de ces données, afin d'estimer les erreurs de classement avec des variances plus faibles.
3. Comparaison finale des qualités de prévision 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 !

3.4 Extraction des échantillons

La suite du traitement nécessite 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.

Utiliser les trois derniers chiffres de son numéro INSEE comme initialisation du générateur. Attention, comme chaque groupe dispose d'un échantillon différent, les résultats obtenus peuvent différer ainsi que certains choix intervenant dans la suite des traitements.

```
set.seed(111) # modifier 111
npop=nrow(vispremv)
# tirage de 200 indices sans remise
testi=sample(1:npop,200)
#Liste des indices restant qui n'ont pas été tirés
appri=setdiff(1:npop,testi)
# Extraction échantillon d'apprentissage
visappt=vispremv[appri,]
# Extraction échantillon de test
vistest=vispremv[testi,]
summary(visappt) # vérifications
summary(vistest)
```

Transfert des données

La bibliothèque `foreign` permet le transfert de données entre R et certains logiciels statistiques commerciaux à condition d'en posséder la licence. Ainsi, dans le cas de SAS, R lance un script SAS pour générer les bonnes tables. Comme les données sont très simples (peu de variables), une option plus élémentaire est employée en générant un fichier texte ; d'autre part, la bibliothèque `foreign` réserve parfois des surprises.

Création à partir de R de fichiers textes au format ASCII contenant les don-

nées :

```
write.table(visappt,"visappt.dat",row.names=FALSE)
write.table(vistest,"vistest.dat",row.names=FALSE)
```

Importation des données sous SAS (SAS et R doivent avoir été lancés dans le même répertoire sinon, le chemin d'accès aux fichiers doit être adapté). De façon assez étonnante, la correspondance entre le codage (entre quotes) des variables qualitatives avec R est reconnu automatiquement par la procédure d'importation de SAS.

```
proc import datafile="vistest.dat"
  out=sasuser.vistest
  dbms=dlm
  replace;
getnames=yes;
run;
proc import datafile="visappt.dat"
  out=sasuser.visappt
  dbms=dlm
  replace;
getnames=yes;
run;
```

S'assurer du bon transfert des données en visualisant rapidement les distributions des variables (pas toutes) et des liaisons potentielles avec la variable à modéliser en utilisant le module d'exploration interactive de SAS.

4 Régression logistique avec SAS et R

4.1 Introduction

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. Elle se généralise à plus de deux lorsque les modalités sont ordonnées. Cette séance a pour objectif d'appliquer pour comparer les procédures de SAS et la fonction de R de régression logistique aux données de cartes visa qui auront été préalablement générées au cours de la session "extraction". Différentes stratégies de choix de modèle sont utilisées. Les erreurs de prévision sont estimées.

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.

5 Estimation et choix de modèle avec SAS

À ce niveau de l'étude, on s'intéresserait aux types des variables : faut-il conserver les variables quantitatives en l'état ou les remplacer par leur version qualitative par découpage en classe ? Les résultats de l'analyse des correspondances font pencher pour cette deuxième solution (meilleure discrimination visuelle des possesseurs et non possesseurs). Dans cette première approche, pour faire simple, on se contente de prendre toutes les variables qualitatives sauf pour l'analyse discriminante où toutes les variables doivent être quantitatives ou binaires.

Choix manuel

Tous les logiciels n'offrent pas de procédures automatiques de choix de modèle, il est utile de savoir le faire "à la main". Mettre en œuvre la procédure descendante de choix de modèle avec SAS/INSIGHT à partir du modèle incluant toutes les variables *qualitatives* en explicatif, mais pas les interactions, pour expliquer `carvp`. Attention, certaines procédures de SAS nécessitent que la variable à expliquer soit quantitative ainsi que la présence d'une variable dénombrant le nombre d'expériences (ici poids=1) ; d'où la procédure ci-dessous pour ajouter ces variables. La variable poids permets par ailleurs de distinguer échantillons d'apprentissage et de test.

```
data sasuser.visappt;
set sasuser.visappt;
if carvp ="Cnon" then carvpr=0 ; else carvpr=1;
poids=1;
data sasuser.vistest;
set sasuser.vistest;
if carvp ="Cnon" then carvpr=0 ; else carvpr=1;
poids=0;
```

```
run;
```

Estimer un modèle binomial : `Analyse>fit`
Sélectionner en `Y` la variable à expliquer (`CARVRr`). Elle est nécessairement de type réel (interval donc 0,1) pour cette procédure.
Sélectionner toutes les variables explicatives qualitatives (identificateurs se terminent par "Q") dans `X`
Cliquer sur `Method` et choisir `Binomial` au lieu de `Normal` qui est le choix par défaut correspondant au modèle gaussien ou régression multilinéaire. Laisser la fonction *lien canonique* qui, dans le cas binomial, est justement la fonction logit.
`OK,OK` Le modèle est estimé et le tableau de type III fournit les statistiques des tests de Wald sur la significativité des paramètres du modèle.

Itérer la procédure suivante pour choisir le modèle :

1. Choisir, parmi les variables explicatives, celle X^j pour lequel le test de Wald ($H_0 : b_j = 0$) est le moins significatif, c'est-à-dire avec la plus grande "prob value".
2. La retirer du modèle et recalculer l'estimation. Il suffit pour cela de sélectionner le nom de la variable dans le tableau (TYPE III) et d'exécuter la commande `delete` du menu `edit` de la même fenêtre.

Arrêter le processus lorsque tous les coefficients sont considérés comme significativement (à 5%) différents de 0. Attention, la "variable" INTERCEPT ne peut pas être considérée au même titre que les autres variables ; sauf exception, il faut toujours *conserver* le terme constant dans le modèle.

Noter la séquence des modèles ainsi obtenus pour la comparer avec la procédure automatique ci-dessous.

Une estimation de l'erreur apparente (ou par resubstitution) de classement est obtenue en suivant la procédure suivante :
transformer la variable `p_carvpr` qui est la prévision, calculée par le modèle, de la probabilité de détention de la carte visa premier.
`Edit>Variables>Other> (a<=Y<=b) et poser a=0.5`
permet donc de créer une nouvelle variable binaire (0,1) prédisant CARVP
transformer le type `interval` de cette variable en type `nominal`
puis créer une représentation graphique de la matrice de confusion :
`Analyse>Mosaic plot` croisant `I_P_CARV` et `CARVP`.

La diagonale fournit le pourcentage (apparent) de bien classés, les autres blocs, le pourcentage *apparent* de mal classés.

Choix de modèle automatique

Dans les versions précédentes de SAS (6 et 7), la procédure `logistic` ne pouvait prendre en compte que des variables explicatives quantitatives. Elle a été adaptée dans la version 8 pour considérer des variables explicatives qualitatives et quantitatives. Différents types d'algorithmes de sélection sont disponibles (backward, forward, stepwise). Attention, le critère utilisé pour ce choix est basé sur un test de Wald, il n'atteint donc pas nécessairement l'objectif de meilleure prévision. Vérifier que l'algorithme "backward" redonne bien votre sélection avec SAS/Insight.

```
proc logistic data=sasuser.visappt descending;
class sexeq famiq pcspq kvunbq--itavcq;
model carvp =  sexeq famiq pcspq kvunbq--itavcq
  /selection=stepwise pprob=0.5 ctable;
  /* ou backward ou forward */
run;
```

Exécuter les trois algorithmes de sélection. Que dire des taux de classement ? Comparer les modèles choisis. Avec la paramétrisation de cette procédure, l'interprétation des paramètres des modèles est difficile. Il est préférable d'utiliser celle ci-dessous.

Comparaisons

Deux modèles sont en concurrence : celui obtenu par la méthode descendante et celui obtenu par la méthode ascendante. Il s'agit maintenant de finaliser le choix d'un modèle pour la régression logistique parmi les deux et éventuellement de l'améliorer encore en supprimant ou ajoutant quelques variables de façon à minimiser une erreur de prédiction plutôt que d'ajustement. Nous allons calculer deux estimations de cette erreur, la première, biaisée, est obtenue par resubstitution des données d'apprentissage. C'est un taux apparent d'erreur nécessairement optimiste. La deuxième estimation opère par validation croisée, elle n'est en principe pas biaisée ou alors de façon pessimiste.

Par resubstitution

Le programme ci-dessous utilise la procédure `genmod` qui fournit exactement les mêmes résultats que SAS/insight tandis que la procédure `logistic` n'utilisant pas la même paramétrisation aboutit à des résultats en apparence différents.

```
proc genmod data=sasuser.visappt ;
    class  moyrvq dmvtpq pcspq sexeq opgnbq relatq
    kvunbq itavcq facanq nptagq uemnbq ;
    /* liste variables qualitatives */
    output out=outglm p=pred;
    model carvpr/poids =  moyrvq dmvtpq pcspq
    sexeq opgnbq relatq kvunbq itavcq facanq nptagq
    uemnbq / d=bin;
    /* liste de toutes les variables du modèle */
    run;
/* Estimation biaisée du taux de mal classés */
data prev ;
set outglm (keep=carvpr pred);
if pred ge 0.5 then predy=1;
               else predy=0;
run;
proc freq data=prev;
tables carvpr*predy/ nocol norow;
run;
```

Exécuter ce programme pour les deux modèles précédemment trouvés en changeant donc les listes de variables. Interpréter les signes des paramètres.

Par validation croisée

Deux macros commandes SAS de validation croisée ont été programmées

`vc1logit` Estime n fois le modèle de régression en supprimant à chaque fois une observation. Cet algorithme classique (delete one cross validation) n'est pas adapté aux gros échantillons.

`vcklogit` Découpe aléatoirement l'échantillon en k groupes ou "plis" (fold) et estime k fois la régression sur l'échantillon constitué de $k-1$ groupes.

Les deux algorithmes estiment un taux d'erreur en cumulant les taux observés sur les parties "validation" des échantillons.

Estimer par validation croisée en exécutant la macro ci-dessous avec les bonnes listes de variables, toujours pour comparer les deux modèles.

```
%vcklogit(visappt,moyrvq dmvtpq pcspq sexeq opgnbq
    relatq kvunbq itavcq facanq nptagq uemnbq,
    carvpr,moyrvq dmvtpq pcspq sexeq opgnbq relatq
         kvunbq itavcq facanq nptagq uemnbq,k=10);
```

Les taux d'erreur sont généralement moins optimistes que ceux obtenus par resubstitution. Noter le meilleur modèle au sens de la validation croisée.

Tenter d'améliorer encore ce modèle en supprimant par exemple des variables jugées parmi les moins significatives lors de l'étape de sélection de façon à minimiser l'erreur par validation croisée. Quel choix définitif retiendriez vous ?

Prévision de l'échantillon test

Utiliser le choix issu des étapes précédentes pour prévoir l'échantillon test et estimer à nouveau l'erreur indépendamment de la procédure de choix de modèle. Ne plus retoucher au modèle obtenu à l'étape précédente.

```
data visconct;
set sasuser.visappt sasuser.vistest;
run;
proc genmod data=visconct;
    class moyrvq dmvtpq pcspq sexeq opgnbq relatq
    kvunbq itavcq facanq nptagq uemnbq;
    output out=outglm p=pred;
    freq poids;
    model carvpr/poids = moyrvq dmvtpq pcspq
    sexeq opgnbq relatq kvunbq itavcq facanq
    nptagq uemnbq/ d=bin;
    run;
data prev ;
set outglm (keep=carvpr poids pred);
if poids=0 then do;
```

```
      if pred ge 0.5 then predy=1;
                     else predy=0;
          output;
          end;
run;
proc freq data=prev;
tables carvpr*predy/ nocol norow;
run;
```

Noter le résultat afin de le comparer avec ceux des autres techniques de discrimination.

6 Régression logistique avec R

6.1 Estimation

Estimation d'un modèle de régression logistique expliquant `CARVP`. On se limite aux seuls prédicteurs qualitatifs plus efficaces dans le cas de la régression logistique mais la même démarche pourrait être mise en œuvre avec les prédicteurs quantitatifs voire tous les prédicteurs et d'en comparer les résultats. 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).

```
# sélection des prédicteurs qualitatifs
visapptq=visappt[,c("CARVP", varqual)]
# pour l'échantillon test
vistestq=vistest[,c("CARVP", varqual)]
# Estimation du modèle complet sans interaction
visa.logit=glm(CARVP~.,data=visapptq,
     family=binomial,na.action=na.omit)
# tests de nullité des coefficients
anova(visa.logit,test="Chisq")
```

Commentaires sur les prob-values.

6.2 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 :

```
visa.logit=glm(CARVP~.,data=visapptq,
    family=binomial,na.action=na.omit)
visa.step<-step(visa.logit)
# variables du modèles
anova(visa.step,test="Chisq")
```

Celle pas à pas :

```
visa.logit=glm(CARVP~1,data=visapptq,
     family=binomial,na.action=na.omit)
visa.step<-step(visa.logit,direction="both",
   scope=list(lower=~1, upper=~SEXEQ + FAMIQ+PCSPQ +
   kvunbq + vienbq+uemnbq + xlgnbq+ylvnbq+nptagq +
   endetq + gagetq + facanq +lgagtq+ havefq +
   ageq+relatq + qsmoyq + opgnbq + moyrvq + dmvtpq +
   boppnq + jnbjdq + itavcq))
# observer les p-values
anova(visa.step,test="Chisq")
```

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

Les modèles obtenus peuvent différer et certaines variables (boppnq ?) 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
visa1.logit=glm(CARVP ~ SEXEQ + PCSPQ + kvunbq +
   uemnbq + nptagq + endetq + gagetq + facanq +
   havefq + relatq + qsmoyq + opgnbq + moyrvq +
   dmvtpq + boppnq + jnbjdq + itavcq,
   data=visapptq,family=binomial,na.action=na.omit)
cv.glm(visapptq,visa1.logit,K=10)$delta[1]
# deuxieme modele
visa2.logit=glm(CARVP ~ SEXEQ + PCSPQ + kvunbq +
   uemnbq + nptagq + endetq + gagetq + facanq +
   havefq + relatq + qsmoyq + opgnbq + moyrvq +
   dmvtpq + jnbjdq + itavcq, data=visapptq,
   family=binomial,na.action=na.omit)
cv.glm(visapptq,visa2.logit,K=10)$delta[1]
```

...

Retenir le meilleur modèle (ce n'est pas nécessairement celui ci-dessous), l'estimer une dernière fois.

```
visa.logit=glm(CARVP ~ SEXEQ + PCSPQ + kvunbq +
   uemnbq + nptagq + endetq + gagetq + facanq +
   havefq + relatq + qsmoyq + opgnbq + moyrvq +
   dmvtpq + boppnq + jnbjdq + itavcq,
   data=visapptq,family=binomial,na.action=na.omit)
```

6.3 Prévision de l'échantillon test

Les commandes ci-dessous calculent la prévision de la variable `CARVP` 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 seul de 0,5.

```
pred.vistest=predict(visa.logit,
   newdata=vistestq)>0.5
table(pred.vistest,vistestq$CARVP=="Coui")
```

Noter le taux d'erreur. Comparer avec les résultats fournis par SAS. Notez que les deux logiciels n'utilisant pas le même critère (Fisher pour SAS, AIC pour R) peuvent conduire à des résultats différents.

6.4 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(visa.logit, newdata=vistestq,
   type="response")
predlogistic=prediction(roclogit,vistestq$CARVP)
perflogistic=performance(predlogistic, "tpr","fpr")
plot(perflogistic,col=1)
```

7 Discrimination avec SAS

La procédure `discrim` de SAS propose plusieurs méthodes associées à de très nombreuses options pour le classement d'individus c'est-à-dire la prévision de la modalité d'une variable qualitative connaissant leurs valeurs sur un ensemble de variables quantitatives. Le modèle de prévision est estimé sur un échantillon d'apprentissage. Pour chaque individu de cet échantillon, sont connues leurs valeurs sur le même ensemble de variables quantitatives ainsi que leur appartenance à une classe. La procédure `discrim` peut calculer, dans la même exécution, les estimations des paramètres sur un échantillon d'apprentissage ainsi que les prévisions des classes d'un échantillon de validation ou de test. Une étude préalable (analyse factorielle discriminante) avec la procédure `candisc` permet de s'assurer du bon pouvoir discriminant des variables quantitatives tandis que la procédure `stepdisc` permet d'aider au choix des variables participant au modèle.

La prise en compte de variables qualitatives est réalisée en considérant les coordonnées ou scores issues d'une AFCM. Cette approche est encore appelée méthode DISQUAL en France. Différentes approches concurrentes sont donc à comparer à travers une stratégie rigoureuse d'évaluation de l'erreur de prévision : options (paramétrique ou non) de l'analyse discriminante, variables quantitatives ou recodage par AFCM des variables qualitatives... Cette dernière option utilisant l'AFCM n'est pas reprise ici.

7.1 Sélection de variables

La procédure `stepdisc` procède pas à pas. À chaque étape, elle compare l'intérêt qu'il y a, au sens d'un certain critère, de faire entrer, ou faire sortir, une des variables de l'ensemble des variables sélectionnées. Elle ne peut prendre en compte que des variables quantitatives.

```
proc stepdisc data=sasuser.visappt;
class carvp;
var familr--gageml qcredl--lgagtl viemtl
   xlgmtl ylvmtl itavcl--jnbjdl;
run;
```

Noter la liste des variables sélectionnées (fenêtre `output`) ; elle est utilisée dans la suite de cette section pour chacune des versions de l'ADD. Attention, elle est différente de celle donnée en exemple dans les programmes puisque chaque utilisateur a tiré un échantillon différent.

7.2 Analyse discriminante

Méthode non paramétrique des k plus proches voisins.

```
proc discrim data= sasuser.visappt
  method=npar k=11 crossvalidate;
  class CARVP;
  var moyrvl sexer boppnl kvunb rocnb opgnbl
  facanl nptag relat havefl qcredl
  dmvtpl viemtl ager xlgmtl tavepl endetl
  gagetl gageml;
  run;
```

Méthode paramétrique avec hypothèse d'égalité des variances (ADD linéaire).

```
proc discrim data= sasuser.visappt
  method=NORMAL POOL=YES crossvalidate;
  class CARVP;  /* discrimination lineaire */
  var nptag sexer boppnl moyrvl opgnbl gagecl
  facanl xlgmtl qcredl kvunb havefl relat ylvmtl
  ager endetl vienb gageml  rocnb lgagtl;
  run;
```

Méthode paramétrique avec variances différentes (ADD quadratique).

```
proc discrim data= sasuser.visappt
  method=NORMAL POOL=NO crossvalidate;
  class CARVP; /* discrimination quadratique */
  var nptag sexer boppnl moyrvl opgnbl gagecl
  facanl xlgmtl qcredl kvunb havefl relat ylvmtl
  ager endetl vienb gageml  rocnb lgagtl;
  run;
```

Comparer les estimations très "théoriques" d'erreur de classement obtenues par *resubstitution* ainsi que celles obtenues par (pseudo) *validation croisée* entre les méthodes et pour différentes valeurs de k. Remarquer que pour $k = 1$, l'erreur estimée par resubstitution est très optimiste car nulle mais naturellement beaucoup plus élevée lorsqu'estimée par validation croisée. Dans ce cas trivial, l'estimation de l'erreur par resubstitution conduit en effet à prédire la classe d'une observation par celle de l'observation la plus proche qui n'est autre qu'elle même ! Ce n'est évidemment pas le cas par validation croisée puisque l'observation dont on cherche à prédire la classe est exclue des voisins potentiels.

À ce niveau, il est toujours possible de modifier la liste des variables en conservant l'objectif de minimiser l'erreur de prédiction estimée par validation croisée.

Retenir le type d'ADD et le modèle optimaux ainsi obtenus.

7.3 Prévision de l'échantillon test

Utiliser l'association, méthode $\times$ liste de variables, jugée optimale dans l'étape précédente et seulement celle-là pour prévoir l'échantillon test.

```
proc discrim data= sasuser.visappt/* ou varprinc */
  testdata=sasuser.vistest testout=testout
  method=????;
  class CARVP;
  var ????;
  run;
proc freq data=testout;
tables carvp*_into_ / nocol norow;
```

```
run;
```

Noter les résultats afin de les comparer avec ceux des autres techniques de discrimination.

8 Analyse discriminante avec R

Nous pouvons comparer les trois méthodes d'analyses discriminantes disponibles dans R : `lda` paramétrique linéaire (homoscédasticité), `lqa` paramétrique quadratique (hétéroscédasticité) sous hypothèse gaussienne et celle non-paramétrique des k plus proches voisins.

Attention, ces techniques n'acceptent par principe que des variables explicatives ou prédictives quantitatives. Néanmoins, une variable qualitative à deux modalités, par exemple le sexe, peut être considérée comme quantitative sous la forme d'une fonction indicatrice prenant ses valeurs dans $\{0, 1\}$.

8.1 Estimations

Il faudrait se préoccuper du problème du choix des variables au moins pour l'approche paramétrique de l'analyse discriminante. Les librairies standards de R ne proposent pas de procédure automatique contrairement à la procédure de SAS.

```
library(MASS) # chargement des librairies
library(class)
visapptr=visappt[,c("CARVP", varquant)]
vistestr=vistest[,c("CARVP", varquant)]
# analyse discriminante linéaire
visa.disl=lda(CARVP~.,data=visapptr)
# analyse discriminante quadratique
visa.disq=qda(CARVP~.,data=visapptr)
# k plus proches voisins
visa.knn=knn(visapptr[,-1],vistestr[,-1],
   visapptr$CARVP,k=10) #
```

Noter le manque d'homogénéité des commandes de R issues de librairies différentes. L'indice de colonne négatif (-1) permet de retirer la première colonne contenant la variable à prédire de type facteur. Celle-ci est mentionnée en troisième paramètre pour les données d'apprentissage.

Le choix de k peut être fait par validation croisée mais la procédure proposée par la libraire `class` est celle *leave-one-out* donc trop couteuse en temps pour des gros fichiers. Il serait simple de la programmer mais une autre librairie (`e1071`) propose également une batterie de fonctions de validation croisée pour de nombreuses techniques de discrimination.

```
library(e1071)
plot(tune.knn(visapptr[,-1],visapptr$CARVP,
   k=seq(2,30, by=2)))
```

Remarquer que chaque exécution de la commande précédente donne des résultats différents donc très instables. Choisir une valeur "raisonnable" de k et l'utiliser pour prédire l'échantillon test :

```
visa.knn=knn(visapptr[,-1],vistestr[,-1],
   visapptr$CARVP,k=16)
```

8.2 Taux d'erreur sur l'échantillon test et courbe ROC

Les commandes suivantes calculent les tables de confusion pour chacune des méthodes d'analyse discriminante.

```
table(vispremv[testi,"CARVP"],
   predict(visa.disl,vistestr)$class)
table(vispremv[testi,"CARVP"],
   predict(visa.disq,vistestr)$class)
table(visa.knn,vistestq$CARVP)
```

Puis calcule et trace la courbe ROC.

```
ROCdiscrim=predict(visa.disl,vistestr)$posterior[,2]
preddiscrim=prediction(ROCdiscrim,vistestq$CARVP)
perfdiscrim=performance(preddiscrim,"tpr","fpr")
plot(perflogistic,col=1) # tracer les courbes ROC
# en les superposant pour mieux comparer
plot(perfdiscrim,col=2,add=TRUE)
```

Comparer les erreurs obtenues selon les procédures et/ou logiciels utilisés ainsique les courbes ROC.

9 Arbres de décision binaires

La comparaison des métodes et logiciels se poursuit mais avec un souci pour SAS : les procédures algorithmiques d'apprentissage : arbres, réseaux de neurones, agrégation de modèles, ne sont pas disponibles dans les modules standards de SAS. Elles n'ont été développées que dans le module SAS Enterprise Miner très peu répandu car très cher. Il n'est de plus accessible qu'à partir d'un poste client sous windows.

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

9.1 Estimation avec prédicteurs qualitatifs

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
vis.treeq=rpart(CARVP~.,data=visapptq,
   parms=list(split='information'),cp=0.001)
summary(vis.treeq) # Quelques renseignements
plot(vis.treeq) # Tracé de l'arbre
text(vis.treeq) # 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. Il s'agit déjà d'une procédure mal explicitée dans la documentation.

```
plotcp(vis.treeq)
```

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.

```
vis.tree2=prune(vis.treeq,cp=0.015)
plot(vis.tree2)
text(vis.tree2,use.n=TRUE)
```

9.2 Estimation avec prédicteurs quantitatifs

La régression logistique opérant sur une combinaison linéaire des prédicteurs, reste une méthode linéaire et le découpage en classe des variables quantitatives, même s'il conduit à réduire le nombre de degrés de liberté, permet d'approcher une transformation non linéaire des variables par un combinaison de fonctions indicatrices. C'est une des raisons qui fait que les résultats obtenus sont, sur ce jeu de données, tout à fait satisfaisant. En revanche, comme la construction de l'arbre construit des découpages en classes optimaux dans l'objectif de la prédiction, ceux-ci n'apparaissent plus indispensables a priori ; au contraire il peut être préférable de laisser faire l'algorithme. Les transformation log étant monotones ne sont pas non plus nécessaires.

```
vis.treer=rpart(CARVP~.,data=visapptr,
   parms=list(split='information'),cp=0.001)
summary(vis.treer) # Quelques renseignements
plot(vis.treer) # Tracé de l'arbre
text(vis.treer)
# Choix élémentaire du coefficient de complexité
plotcp(vis.treer)
vis.tree3=prune(vis.treer,cp=0.0096)
```

9.3 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.

9.3.1 Prédicteurs qualitatifs

```
xmat = xpred.rpart(vis.treeq,xval=10,
   cp=seq(0.03,0.01,length=10))
```

```
# Comparaison de la valeur prédite
# avec la valeur observée.
xerr=as.integer(visapptq$CARVP)!= 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
vis.tree2=prune(vis.treeq,cp=0.014)
plot(vis.tree2)
text(vis.tree2,use.n=TRUE)
```

9.3.2 *Prédicteurs quantitatifs*

Meme chose avec les prédicteurs quantitatifs :

```
xmat = xpred.rpart(vis.treer,xval=10,
   cp=seq(0.02,0.005,length=10))
xerr=as.integer(visapptr$CARVP)!= xmat
apply(xerr, 2, sum)/nrow(xerr)
vis.tree3=prune(vis.treer,cp=0.01)
plot(vis.tree3)
text(vis.tree3,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.

```
post(vis.tree2)
post(vis.tree3)
```

Interpréter les arbres. Que dire du choix des variables comparativement aux autres méthodes ? Splus mais pas R offre de nombreuses fonctionnalités de diagnostic permettant d'explorer arbre, sous-arbres, nœuds et feuilles. Voir l'aide en ligne de ce logiciel qui contient des exemples pour plus d'informations.

9.4 Prévision de l'échantillon test

Les commandes ci-dessous calculent la prévision de la variable `CARVP` 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.

```
# arbre utilisant des prédicteurs qualitatifs
pred.vistest=predict(vis.tree2,
   newdata=vistestq,type="class")
table(pred.vistest,vistestq$CARVP)
# arbre utilisant les prédicteurs quantitatifs
pred.vistest=predict(vis.tree3,
   newdata=vistestr,type="class")
table(pred.vistest,vistestq$CARVP)
```

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

```
ROCrpart=predict(vis.tree3,newdata=vistestr,
   type="prob")[,2]
predrpart=prediction(ROCrpart,vistestq$CARVP)
perfrpart=performance(predrpart,"tpr","fpr")
# tracer le socurbes ROC
plot(perflogistic,col=1)
# en les superposant pour mieux comparer
plot(perfdiscrim,col=2,add=TRUE)
plot(perfrpart,col=3,add=TRUE)
```

Notez qu'il n'y a pas de métode uniformément meilleure et en conséquence, l'aire sous la courbe ROC (AUC) ne génère pas un ordre total.

10 Agrégation de modèles

Des séances précédentes ont permis d'expérimenter les techniques maintenant classiques de construction d'un modèle de prévision assorties de leur problème récurrent liés à l'optimisation de la complexité du modèle. Cette séance aborde d'autres stratégies récemment développées dont l'objectif est de s'affranchir de ce problème de choix, par des méthodes se montrant moins sensible au sur-apprentissage ; c'est le cas des algorithmes d'agrégation de modèles.

Sur le plan logiciel, R montre dans cette situation tout son intérêt. La plupart des techniques récentes sont en effet expérimentées avec cet outil et le plus souvent mises à disposition de la communauté scientifique sous la forme d'une librairie afin d'en assurer la "promotion". Pour les techniques d'agrégation de modèles, nous pouvons utiliser les librairies `gbm` et `randomForest` respectivement réalisées par Greg Ridgeway et Leo Breiman. Ce n'est pas systématique, ainsi J. Friedman a retiré l'accès libre à ses fonctions (MART) et créé son entreprise (Salford).

10.1 objectifs

1. tester le bagging et le choix des ensembles de variables ainsi que le nombre d'échantillons considérés,
2. étudier l'influence des paramètres (profondeur d'arbre, nombre d'itérations, shrinkage) sur la qualité de la prévision par boosting ;
3. même chose pour les forêts aléatoires (nb de variables mtry, nodesize).
4. Expérimenter les critères de Breiman qui lui permettent de mesurer l'influence des variables au sein d'une famille agrégée de modèles. Les références bibliographiques sont accessibles sur le site de l'auteur décédé en 2005 : `www.stat.Berkeley.edu/users/breiman`

10.2 Méthodologie

Sur le plan méthodologique, il serait intéressant comme pour les réseaux de neurones de monter un plan d'expérience pour évaluer la sensibilité des techniques aux valeurs des paramètres contrôlant l'exécution des algorithmes (nombre d'itérations, profondeur d'arbre, nombre de variables extraites à chaque décision pour la construction d'un nœud...

10.3 Bagging

En utilisant la fonction `sample` de R, il est très facile d'écrire un algorithme de bagging. Il existe aussi une librairie qui propose des exécutions plus efficaces. Par défaut, l'algorithme construit une famille d'arbres complets (cp=0) et donc de faible biais mais de grande variance. L'erreur out-of-bag permet de contrôler le nombre d'arbres ; un nombre raisonnable semble suffire ici.

Estimations

Avec les seules variables quantitatives comme prédicteurs :

```
library(ipred)
bagging(CARVP~., nbag=50, data=visapptr,
   coob=TRUE)
bagging(CARVP~., nbag=25, data=visapptr,
   coob=TRUE)
```

Avec les seules variables qualitatives comme prédicteurs :

```
bagging(CARVP~., nbag=50, data=visapptq,
   coob=TRUE)
bagging(CARVP~., nbag=25, data=visapptq,
   coob=TRUE)
```

Il est aussi possible de considérer globalement toutes les variables ; ceci montre la pertinence de ce type d'algorithme robuste face au risque de sur apprentissage.

```
vp.bag=bagging(CARVP~., nbag=50, data=visappt,
   coob=TRUE)
vp.bag=bagging(CARVP~., nbag=25, data=visappt,
   coob=TRUE)
```

Retenir l'approche (nombre d'arbres, ensemble de variables) dont l'erreur oob apparaît la plus faible sur quelques exécutions.

Les valeurs d'autres paramètres peuvent être explorées. Ainsi, pour se limiter à des arbres élagués :

```
bagging(CARVP~.,nbag=50,control=rpart.control(cp=0.1),
   data=visapptq,coob=TRUE)
```

Cela nécessite une optimisation du choix du paramètre. Remarquer néanmoins que le nombre d'arbres (nbag) n'est pas un paramètre "sensible" et qu'il suffit de se contenter d'arbres "entiers".

Comme cela a été remarqué avec la construction d'arbres binaires de décision (rpart), l'utilisation des variables quantitatives comme prédicteurs est pré-

férable à celui des variables qualitatives obtenues par découpages en classes. Ce choix est privilégié par la suite.

Prévision de l'échantillon test

Retenir la meilleure combinaison de paramètres qui n'est pas nécessairement celle ci-dessous puis ré-estimer le modèle :

```
vp.bag=bagging(CARVP~., nbag=50, data=visapptr,
   coob=TRUE)
```

Avant de construire la matrice de confusion :

```
pred.vistest=predict(vp.bag,newdata=vistestr,
   type="class")
table(pred.vistest,vistestq$CARVP)
```

Noter le taux d'erreur et construire les courbes ROC.

```
ROCbag=predict(vp.bag,vistestr,type="prob")[,2]
predbag=prediction(ROCbag,vistestq$CARVP)
perfbag=performance(predbag,"tpr","fpr")
# tracer les courbes ROC
plot(perflogistic,col=1)
plot(perfdiscrim,col=2,add=TRUE)
plot(perfrpart,col=3,add=TRUE)
plot(perfnnet,col=4,add=TRUE)
plot(perfbag,col=5,add=TRUE)
legend("bottomright",legend=c("log","disc",
   "rpart","nnet","bag"),col=1:5,pch="_")
```

10.4 Boosting

Fonction ad'hoc

L'algorithme basique (adaboost) a été programmé en utilisant la librairie (rpart) spécifique pour la recherche d'un arbre qui offre la possibilité, indispensable en boosting, de faire varier les pondérations des observations.

Il faut faire attention au cadre d'utilisation de cet algorithme et à la place de l'échantillon test. A moins de stocker tous les arbres de la famille, il est nécessaire de ré-exécuter l'apprentissage à chaque utilisation de l'algorithme pour construire une prévision. C'est la raison de la présence de l'échantillon test à chaque exécution visant à évaluer l'impact d'un paramètre. Ceci ne doit pas empêcher la rigueur et demande de l'attention : il ne faut pas optimiser ces paramètres en considérant l'erreur sur l'échantillon test mais bien en considérant des erreurs "endogènes" calculées à partir du seul échantillon d'apprentissage comme l'erreur "out of bag" (oob).

```
# librairie pour les arbres appelées
# par la fonction adaboost
library(rpart)
# entrer la fonction de boostin
source("adaboost.txt")g
# Sur les seule prédicteurs quantitatifs
# avec des paramètres par défaut :
fit=adaboost(CARVP~., visapptr, vistestr)
table((fit+3)/2,as.numeric(vistestr$CARVP))
# matrice de confusion pour le test
# Retrouve évidemment l'erreur en fin
# d'exécution de l'algorithme
# Faire varier le nombre d'itérations et
# a profondeur d'arbres :
fit=adaboost(CARVP~., visapptr, vistestr,
   mfinal=20,prof=3)
fit=adaboost(CARVP~., visapptr, vistestr,
   mfinal=50,prof=1)
```

En fait, il semble qu'il n'y ait pas réellement de paramètre à régler car l'erreur n'apparaît pas comme très sensible à ceux-ci (nombre d'itérations, profondeur d'arbre). Il suffit en général de prendre un nombre d'itérations suffisant au regard de la complexité des arbres : 50 à 100 pour une profondeur réduite (1), 20 à 50 pour une profondeur supérieure (3).

Remarquer le comportement exceptionnel de cet algorithme qui, alors que l'erreur d'apprentissage s'est annulée, ne provoque toujours pas de sur-apprentissage : l'erreur sur l'échantillon test continue toujours de décroître !

Librairie `gbm`

Deux librairies proposent des versions plus sophistiquées des algorithmes de boosting dans R. La librairie `boost` propose 4 approches : adaboost, bagboost et deux logitboost. Développées pour une problématique particulière : l'analyse des données d'expression génomique, elle n'est peut-être pas complètement adaptée aux données étudiées ; elles se limitent à des prédicteurs quantitatifs et peut fournir des résultats étranges. La librairie `gbm` lui est préférée ; elle offre aussi plusieurs versions dépendant de la fonction coût choisie. Cependant, le chargement de cette librairie peut poser des problèmes en fonction de l'environnement système ; d'où l'intérêt de la fonction ad'hoc précédente.

La variable à prédire doit être codée numériquement (0,1) pour cette implémentation. Le nombre d'itérations, ou nombre d'arbres, est paramétré ainsi qu'un coefficient de "shrinkage" contrôlant le taux ou pas d'apprentissage. Attention, par défaut ce paramètre a une valeur très faible (0.001) et il faut un nombre important d'itérations pour atteindre une estimation raisonnable. La qualité est visualisée par un graphe représentant l'évolution de l'erreur d'apprentissage ; d'autre part, une procédure de validation croisée est incorporée qui fournit le nombre optimal d'itérations à considérer.

Apprentissage sans shrinkage

```
library(gbm)
vp.boost=gbm(as.numeric(CARVP)-1~., data=visapptr,
    distribution="adaboost",n.trees=100, cv.folds=10,
    n.minobsinnode = 5,shrinkage=1,verbose=FALSE)
best.iter=gbm.perf(vp.boost,method="cv")
# nombre optimal d'itérations par validation croisée
print(best.iter)
```

Shrinkage

Les commandes suivantes permettent d'évaluer l'impact du coefficient de shrinkage :

```
vp.boost1=gbm(as.numeric(CARVP)-1~., data=visapptr,
    distribution="adaboost",n.trees=1000, cv.folds=10,
    n.minobsinnode = 5,shrinkage=0.1,verbose=FALSE)
best1.iter=gbm.perf(vp.boost1,method="cv")
best1.iter
vp.boost2=gbm(as.numeric(CARVP)-1~., data=visapptr,
    distribution="adaboost",n.trees=3000, cv.folds=10,
    n.minobsinnode = 5,shrinkage=0.01,verbose=FALSE)
best2.iter=gbm.perf(vp.boost2,method="cv")
best2.iter
```

Comparer les valeurs des erreurs par validation croisée associées au nombre optimal d'itération pour chaque valeur du taux d'apprentissage :

```
vp.boost$cv.error[best.iter]
vp.boost2$cv.error[best2.iter]
```

et déterminer le "bon" choix ...

Echantillon test

On peut s'assurer de l'absence d'un sur-apprentissage critique en calculant puis traçant l'évolution de l'erreur sur l'échantillon test :

```
test=numeric()
for (i in 10:1000){
pred.vistest=predict(vp.boost1,newdata=vistestr,
   n.trees=i)
taux=table(as.factor(sign(pred.vistest)),
   vistestq$CARVP)
test=c(test,(taux[1,2]+taux[2,1])/200)}

# Tracé du graphe
plot(10:1000,test,type="l")
# Nb "optimal" d'itérations fixé par validation croisée
abline(v=best1.iter)
```

La prévision de l'échantillon test et de la matrice de confusion associée sont obtenus par les commandes :

```
pred1.vistest=predict(vp.boost1,newdata=vistestr,
   n.trees=best1.iter)
table(as.factor(sign(pred1.vistest)),vistestq$CARVP)
```

Courbes ROC

```
ROCboost=predict(vp.boost1,newdata=vistestr,
   n.trees=best1.iter)
predboost=prediction(ROCboost,vistestq$CARVP)
perfboost=performance(predboost,"tpr","fpr")
# tracer le socurbes ROC
plot(perflogistic,col=1)
plot(perfdiscrim,col=2,add=TRUE)
plot(perfrpart,col=3,add=TRUE)
plot(perfnnet,col=4,add=TRUE)
plot(perfbag,col=5,add=TRUE)
plot(perfboost,col=6,add=TRUE)
legend("bottomright",legend=c("log","disc",
   "rpart","nnet","bag","boost"),
     col=1:6,pch="_")
```

10.5 Forêt aléatoire

Le programme est disponible dans la librairie `randomForest`. Il est écrit en fortran, donc efficace en terme de rapidité d'exécution, mais facile à utiliser grâce à une interface avec R. Les paramètres et résultats sont explicités dans l'aide en ligne.

Estimation

```
# charger la librairie
library(randomForest)
# aide en ligne
?randomForest
# Avec les seules variables quantitatives :
fit=randomForest(CARVP~.,data=visapptr,
   xtest=vistestr[,varquant],
   ytest=vistestr[,"CARVP"],do.trace=20,
   importance=TRUE,norm.vote=FALSE)
print(fit)
```

Observer l'évolution des erreurs (oob, test) en fonction du nombre d'arbres.

Il n'y a pas de modèle a interpréter mais une liste de coefficients d'importance associés à chaque variable :

```
print(round(fit$importance, 2))
```

Optimisation

Les valeurs par défaut (nb d'arbres, nombre de variables tirées aléatoirement) semblent déjà donner de bons résultats. La fonction `tune` de la librairie `e1071` permet de plus une recherche automatique des valeurs optimales.

```
res=tune(randomForest, CARVP~., data=visapptr,
  tunecontrol=tune.control(sampling="cross",cross=10),
  ranges=list(mtry=c(10,17,24),ntree=c(200,400,600)))
res
```

S'armer de patience... L'utilisation de calculateurs parallèles ou de clusters commence à se justifier...

Il y a 3*10*(200+400+600)=36000 arbres à estimer.

Remarque, comme chaque exécution peut donner des résultats différents et que toutes les estimations d'erreurs sont très proches, donc les différences sans doute pas significatives, il n'est peut-être pas nécessaire de se focaliser dessus et les valeurs par défaut conviennent souvent.

Echantillon test

Estimer avec les valeurs "optimales" des paramètres :

```
fit=randomForest(CARVP~.,data=visapptr,
   xtest=vistestr[,varquant],
   ytest=vistestr[,"CARVP"],do.trace=20,
   mtry=17,ntree=600,norm.vote=FALSE)
```

Matrice de confusion pour l'échantillon test :

```
table(fit$test$predicted,  vistest$CARVP)
```

Courbes ROC.

```
ROCrf=fit$test$vote[,2]
predrf=prediction(ROCrf,vistestq$CARVP)
```

```
perfrf=performance(predrf,"tpr","fpr")
# tracer les courbes ROC
plot(perflogistic,col=1)
plot(perfdiscrim,col=2,add=TRUE)
plot(perfrpart,col=3,add=TRUE)
plot(perfnnet,col=4,add=TRUE)
plot(perfbag,col=5,add=TRUE)
plot(perfboost,col=6,add=TRUE)
plot(perfrf,col=8,add=TRUE)
legend("bottomright",legend=c("log","disc",
   "rpart","nnet","bag","boost","randF"),
   col=c(1:6,8),pch="_")
```

Quelle stratégie d'agrégation de modèles vous semble fournir le meilleur résultat de prévision ? Est-elle plus efficace que les modèles classiques expérimentés auparavant ?

11 Machines à vecteurs supports

Cette section propose d'aborder une nouvelle famille d'algorithmes : les SVM ou (*Support Vector Machines* traduit par Séparateurs à Vaste Marge ou machine à vecteurs support) dont le principe fondateur est d'intégrer l'optimisation de la complexité d'un modèle à son estimation ou plus exactement une partie de cette complexité ; cela concerne le nombre de vecteurs supports. Une bibliothèque de R, réalisée par Chang, Chih-Chung et Lin Chih-Jen, est destinée à cette approche ; elle est intégrée au package `e1071`.

Au delà du principe fondateur de recherche de parcimonie (nombre de supports) incorporé à l'estimation, il n'en reste pas moins que cette approche laisse, en pratique, un certain nombre de choix et réglages à l'utilisateur. Il est donc important d'en tester l'influence sur la qualité des résultats.

1. choix du paramètre de régularisation ou pondération d'ajustement,
2. choix du noyau,
3. le cas échéant, choix du paramètre associé au noyau : largeur d'un noyau gaussien, degré d'un noyau polynomial...

Notons la même remarque qu'avec les techniques précédentes sur l'intérêt à mettre en œuvre une approche "plan d'expérience" voire même "surface de réponse" afin d'optimiser le choix des paramètres.

11.1 Options par défaut

Une première exécution en utilisant simplement les options par défaut de l'algorithme conduit aux résultats suivants :

```
vis.svm=svm(CARVP~., data=visapptr)
summary(vis.svm) # affichage des options
vis.pred=predict(vis.svm,data=visapptr)
# erreur d'apprentissage
table(vis.pred,visapptr$CARVP)
vis.pred=predict(vis.svm,newdata=vistestr)
# erreur sur échantilon test
table(vis.pred,vistestr$CARVP)
```

Les résultats ne semblent pas fantastiques. Peut-on les améliorer en ajustant les paramètres ?

11.2 Choix d'options par validation croisée

Malgré les assurances théoriques concernant ce type d'algorithme, les résultats dépendant fortement du choix des paramètres. Nous nous limiterons d'abord au noyau gaussien (choix par défaut) ; la fonction `tune.svm` permet de tester facilement plusieurs situations en estimant la qualité de prédiction par validation croisée sur une grille. Le temps d'exécution est un peu long... en effet, contrairement à beaucoup d'algorithmes de modélisation, la complexité de l'algorithme de résolution des SVM croît très sensiblement avec le nombre d'observations mais moins avec le nombre de variables. 0.03125 2

```
obj = tune.svm(CARVP~., data = visapptr,
 gamma = 2^(-8:-5), cost = 2^(-2:4))
summary(obj)
plot(obj)
```

Modifier éventuellement les bornes de la grille. Noter le couple optimal de paramètres (difficile car cela change à chaque exécution !). Exécuter l'estimation avec ces valeurs de paramètre, puis prévoir l'échantillon test.

```
vis.svm=svm(CARVP~., data=visapptr,
  gamma=0.015,cost=6)
vis.pred=predict(vis.svm,newdata=vistestr)
# erreur sur échantilon test
table(vis.pred,vistestr$CARVP)
```

11.3 Autres noyaux

Les choix précédents utilisaient le noyau gaussien par défaut. Voici la syntaxe pour utilisation d'un noyau polynomial. Tester également d'autres valeurs des paramètres.

```
vis.svm=svm(CARVP~.,data=visapptr,
 kernel="polynomial",gamma=0.02,cost=1,degree=3)
vis.pred=predict(vis.svm,newdata=vistestr)
# erreur sur échantilon test
table(vis.pred,vistestr$CARVP)
```

Même chose pour un noyau sigmoïdal :

```
vis.svm=svm(CARVP~., data=visapptr,
 kernel="sigmoid",gamma=0.02,cost=1)
vis.pred=predict(vis.svm,newdata=vistestr)
# erreur sur échantilon test
table(vis.pred,vistestr$CARVP)
```

11.4 Remarques

Les résultats ne semblent pas beaucoup mieux qu'avec les choix par défaut. Plus d'investigations seraient nécessaires pour apprécier quelle serait la bonne combinaison de noyaux et paramètres. D'autre part, il faudrait tester la possibilité de remplacer les variables qualitatives par des paquets d'indicatrices pour la prise en compte de ces variables.

12 Synthèse des résultats

L'objectif final est d'arriver à une comparaison fine et synthétique des différentes méthodes pour aboutir à des prises de décision : Quelle méthode utiliser pour construire un "meilleur" score d'appétence ?

La taille de l'échantillon test est relativement modeste pour espérer une bonne précision sur la foi d'un seul échantillon. Pour préciser la comparaison c'est-à-dire pour prendre en compte la variance de l'estimation de l'erreur, le processus est itéré.

12.1 Itérations de l'estimation des erreurs

Consulter le programme `comparaison_visa.R`, définir les paramètres : initialisation du générateur (xxx) et nombre d'itérations (xx au moins 30) avant de le faire exécuter en "batch" ou de nuit.

Ce programme lit les données et itère N fois le tirage d'un échantillon test pour estimer N erreurs de prévision (taux de mauvais classés) pour chacune des méthodes de modélisation considérée. Sont calculées sur chaque échantillon test. Ces résultats sont stockés dans la matrice `res`.

Pour construire les courbes ROC, le programme stocke également les prévisions pour chacune des méthodes de modélisation considérée et pour chaque échantillon test dans des variables de type liste et de nom : `list.methode` où méthode est la méthode de modélisation considérée. On se limite aux méthodes apparues les "meilleures" lors de la comparaison des distributions des erreurs. Attention, cette sélection n'est pas forcément "optimale" ; elle pourrait être complétée utilement.

12.2 Synthèses des résultats

Calculer, comparer les moyennes et écarts-types des distributions des taux d'erreur. Tracer les diagrammes boîtes parallèles de ces distributions :

```
boxplot(data.frame(res))
```

Commentaires.

Tracer les courbes ROC par échantillon test en superposant des diagrammes boîtes visualisant les dispersion des courbes :

```
library(ROCR)
#création des objets ROC
pred <- prediction(list.log$predictions,
   list.log$labels)
perf.log <- performance(pred,"tpr","fpr")
```

```
plot(perf.log,col="blue",lty=3)
plot(perf.log,lwd=3,avg="vertical",
   spread.estimate="boxplot",add=TRUE)

pred <- prediction(list.tree$predictions,
   list.tree$labels)
perf.tree <- performance(pred,"tpr","fpr")
plot(perf.tree,col="blue",lty=3)
plot(perf.tree,lwd=3,avg="vertical",
   spread.estimate="boxplot",add=TRUE)

pred <- prediction(list.rn$predictions,
   list.rn$labels)
perf.rn <- performance(pred,"tpr","fpr")
plot(perf.rn,col="grey82",lty=3)
plot(perf.rn,lwd=3,avg="vertical",
   spread.estimate="boxplot",add=TRUE)

pred <- prediction(list.bag$predictions,
   list.bag$labels)
perf.bag <- performance(pred,"tpr","fpr")
plot(perf.bag,col="grey82",lty=3)
plot(perf.bag,lwd=3,avg="vertical",
   spread.estimate="boxplot",add=TRUE)

pred <- prediction(list.boost$predictions,
   list.boost$labels)
perf.boost <- performance(pred,"tpr","fpr")
plot(perf.boost,col="grey82",lty=3)
plot(perf.boost,lwd=3,avg="vertical",
   spread.estimate="boxplot",add=TRUE)

pred <- prediction(list.rf$predictions,
   list.rf$labels)
perf.rf <- performance(pred,"tpr","fpr")
plot(perf.rf,col="blue",lty=3)
plot(perf.rf,lwd=3,avg="vertical",
   spread.estimate="boxplot",add=TRUE)


pred <- prediction(list.svm$predictions,
   list.svm$labels)
perf.svm <- performance(pred,"tpr","fpr")
plot(perf.svm,col="grey82",lty=3)
plot(perf.svm,lwd=3,avg="vertical",
   spread.estimate="boxplot",add=TRUE)
```

Superposer les moyennes par méthode de ces courbes ROC.

```
plot(perf.log,col=1,lwd=2,avg="vertical")
plot(perf.tree,col=2,lwd=2,avg="vertical",add=TRUE)
plot(perf.rn,col=3,lwd=2,avg="vertical",add=TRUE)
plot(perf.bag,col=4,lwd=2,avg="vertical",add=TRUE)
plot(perf.boost,col=5,lwd=2,avg="vertical",add=TRUE)
plot(perf.rf,col=6,lwd=2,avg="vertical",add=TRUE)
plot(perf.svm,col=7,lwd=2,avg="vertical",add=TRUE)
legend("bottomright",legend=c("Logit","Tree",
   "RN","Bag","Boost","RF","SVM"),col=1:7,pch="_")
```

Adapter ces graphiques afin de mettre clairement en évidence la "meilleure" méthode. Remarque : la régression logistique est traditionnellement utilisée par les service GRC des entreprises.