Université de Poitiers

Faculté des Sciences Fondamentales et Appliquées

Département de Mathématiques et Applications

Module : Projet-réalisation

Filière : Mathématiques Spécialité : Statistiques et Données du Vivant

---

Classification des variants génétiques humaines

Réalisé par :

H. NIANDOU AMADOU

M. NDIAYE

R. BOUKENNA

Dirigé par :

Pr. H. BIERME

Pr. F. ENIKEEVA

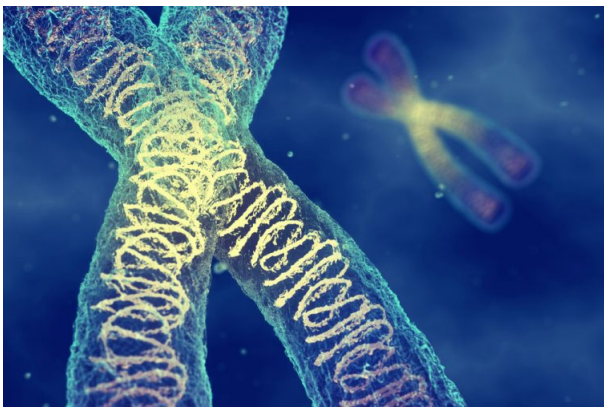


TABLE DES MATIÈRES

INTRODUCTION

Variant génétique

L'expression de : "variant génétique" est utilisée pour décrire une altération qui survient au niveau des séquences nucléotidiques de l'ADN les plus communes. Chez l'être humain, 99% des séquences nucléotidiques de l'ADN sont identiquement partagées entre les différents individus. Cette énorme proportion de similitude dans l'information génétique humaine a permis de considérer un génome de référence qui est géré par le *Genome Reference Consortium*. La séquence d'ADN d'une personne peut différer de la séquence d'ADN de référence de différentes manières, notamment :

- Les polymorphismes mononucléotidiques (*Single Nucleotide Polymorphism* (SNP)) qui sont des variations de séquence d'ADN qui se produisent lorsqu'un seul nucléotide diffère de la séquence d'ADN de référence.
- Les insertions (*Insertion*) qui se produisent lorsque des nucléotides supplémentaires sont insérés dans une séquence d'ADN, par rapport à la séquence de référence.
- Les délétions (*Deletion*) qui surviennent lorsqu'il manque des nucléotides par rapport à la séquence de référence.
- Les substitutions (*Substitution*) qui se produisent lorsque plusieurs nucléotides sont modifiés par rapport à la séquence de référence.
- Les variants structuraux sont des modifications dans lesquelles de grandes sections d'un chromosome ou même de chromosomes entiers sont dupliquées, supprimées ou réarrangées d'une manière ou d'une autre.

Le plus grand défi qui agite beaucoup les généticiens est celui de l'interprétation de ces différents types de variants génétiques. Actuellement, seulement un petit nombre de ces variants ont été liés à des maladies car la plupart possèdent des effets inconnus. Ces variants sont généralement manuellement classés par des laboratoires cliniques selon trois grandes catégories :

— Variants pathogènes.
— Variants d'importance inconnu (*Variant of Unkown Significance*).
— Variants Bénins.

Les variants génétiques dont les classifications sont conflictuelles (d'un laboratoire à un autre) peuvent être source de confusion lorsque les chercheurs tentent d'interpréter si de telles variants possèdent ou non de réels impacts sur l'apparition et l'évolution d'une certaine maladie. Les classifications en conflit surviennent lorsque deux ou trois des catégories citées ci-dessus sont présentes simultanément pour un même variant.

Source de données

Clinvar est une archive publique librement accessible contenant des annotations sur les relations entre les variants génétiques humains et les phénotypes associés, avec des preuves à l'appui. Elle permet de :

— Traiter des soumissions en signalant les variants trouvés dans les échantillons de patients.
— Fournir les affirmations relatives à leur signification clinique.
— Partager les informations sur le soumetteur et les données fournies pour les utilisateurs interactifs ainsi que pour ceux qui souhaitent utiliser ClinVar dans les flux de travail quotidiens et autres applications locales.

La base de données que nous avons traitée provient de ClinVar et elle est partagée sur Kaggle. Étant donné que ce problème ne concerne que les variants avec plusieurs classifications, l'utilisateur qui a partagé la base de données a gardé uniquement les variants qui possèdent deux ou plusieurs soumissions. Les informations sur la nature des classifications des variants sont rapportées au niveau de la variable binaire "CLASS". l'entrée 0 représente des classifications cohérentes, en revanche l'entrée 1 est pour les classifications

en conflit.

Objectif et oganisation du groupe

Notre groupe est constitué de trois membres : Habibou Niandou Amadou, Moustapha Ndiaye et Raouf Boukenna. Après une légère discussion, nous nous sommes mis d'accord que la responsabilité du groupe sera attriubuée à Raouf Boukenna. Nous voulions traiter une base de données liée au domaine médical et notre sélection est tombée sur la base de données ClinVar. Les tâches ont été attribuées équitablement entre les trois membres du groupe. l'avancement du projet était plus basé sur la recherche individuelle de chacun, de telle sorte que nous avions à trouver et proposer l'application de méthodes qui pourrait mieux répondre à notre problématique.
L'objectif était de repérer les principaux facteurs qui peuvent mieux expliquer et/ou prédire si un nouveau variant génétitique de ClinVar aura des classifications contradictoires (conflictuelles).

CHAPITRE 1

BASE DE DONNÉES CLINVAR

1.1 Description de la base de données originale et sa version restructurée

La base de données ClinVar téléchargé sur Kaggle possède 65000 observations mesurées sur 46 variables. chaque observation représente un variant génétique qui peut être sujet soit à une classification conflictuelle (”CLASS” = 1), soit à une classification cohérente (”CLASS” = 0). Les variables donnent les majeures caractéristiques des variants génétiques de ClinVar, nous pouvons en citer :

— ”CHROM” : Le chromosome où la variante est située sur.

— ”AF” : Fréquences alléliques des variantes.

— ”CLNDN” : Noms des maladies qui peuvent être associées au variant.

— ”CLNVC” : Type de variant.

— ”ORIGIN” : Origine de l’allèle du variant.

— ”POLYPHEN” : Prédiction sur la nature du polyphénotype.

— ”CADD” : Outil qui génère un score évaluant le caractère délétère d’un variant.

— ”EXON” : Le nombre et la taille des séquences nucléotidiques qui seront conservées dans l’ARN mature après épissage.

— ”INTRON” : Le nombre et la taille des séquences nucléotidiques qui ne seront pas

conservées dans l'ARN mature après épissage.

— "IMPACT" : Degré de l'impact du variant génétique sur la constitution du polyphénotype.

Les données présentées étaient loin d'être facile à exploiter. La première étape de notre étude était de transformer l'intégralité de la base de données en un nouvel format utilisable. Nous avons repéré des variables absurdes que nous les avons enlevées : variables redondantes, vides d'informations (valeur "NA" uniquement) ou bien elles contenaient uniquement une seule valeur. Une fois ces dernières enlevées, nous avons restructuré les modalités présentes dans certaines variables qualitatives qui étaient soient présentées sous un mauvais format soient elles possédaient des valeurs presques identiques.

La deuxième étape consistait à chercher et trouver une méthode assez robuste dans le sens où elle sera capable à prédire, quelles que soient la nature et le format des variables, l'énorme nombre de données manquantes qui étaient dans la base.

Une fois les données manquantes complétées, nous avons passé à la prochaine et dernière étape : puisque les méthodes statistiques peuvent uniquement fonctionner avec des données numériques, nous étions contraints d'utiliser une fonction qui transforme notre base de données en un tableau disjonctif ce qui veut dire que les modalités des variables qualitatives sont devenues elles mêmes des variables codées en binaire.

À ce niveau, la base de données est suffisamment exploitable mais il reste à vérifier si le logiciel *R* parvient à compiler des méthodes statistiques avec un aussi grand nombre d'observations : 65000.

1.1.1 Méthodes d'imputation multiple par équations chaînées

La méthode d'imputation multiple par équations chaînées (MICE) permet de traiter les données manquantes. Elle stipule que,compte tenu des variables utilisées dans la procédure d'imputation, les données manquantes sont Missing at Random (MAR), ce qui signifie que la probabibilité qu'une valeur soit manquante dépend uniquement des valeurs observées et non des valeurs non observées. Elle peut être décomposé en quatre étapes générales :

1. Une imputation simple, telle que l'imputation de la moyenne, est effectuée pour chaque valeur manquante dans l'ensemble de données. Ces imputations moyennes

peuvent être considérées comme des "repères".

2. Les imputations repères pour une variable X fixée sont redéfinies comme manquantes.
3. Les valeurs observées de la variable X à l'étape 2 sont régressées sur les autres variables dans le modèle d'imputation. En d'autres termes, X est la variable dépendante dans un modèle de régression et toutes les autres variables sont indépendantes dans ce modèle. Ces modèles de régression fonctionnent selon les mêmes hypothèses que celles que l'on ferait lorsqu'on applique des modèles de régression linéaire, logistique ou de Poisson en dehors du contexte d'imputation des données manquantes.
4. Les valeurs manquantes pour X sont ensuite remplacées par les prédictions (dites imputations) du modèle de régression. Lorsque X est utilisée par la suite comme variable indépendante dans les modèles de régression pour les autres variables, les valeurs observées et celles déja imputées sont également utilisées.
5. Les étapes 2 et 4 sont ensuite répétées pour chaque variable pour laquelle des données sont manquantes. Les itérations faites pour chacune des variables constitue un cycle. A la fin d'un cycle, toutes les valeurs manquantes ont été remplacées par des prédicitons issues de régressions reflétant les relations observées dans les données.
6. Les étapes 2 à 4 sont répétées pendant plusieurs cycles, les imputations étant mises à jour à chaque cycle. Le nombre de cycles à effectuer peut être spécifié par l'utilisateur. A la fin de ces cycles, les imputations finales sont conservées, ce qui donne un jeu de données imputé.

La méthode peut être implémenté sur R via le package : MICE. Le seul soucis de son application est qu'elle est très couteuse (temps de calcul très élevé) et converge très rarement. Nous avons été contrait de limiter le nombre de cycles pour que la méthode soit applicable.

1.1.2 Méthode d'échantillonnage

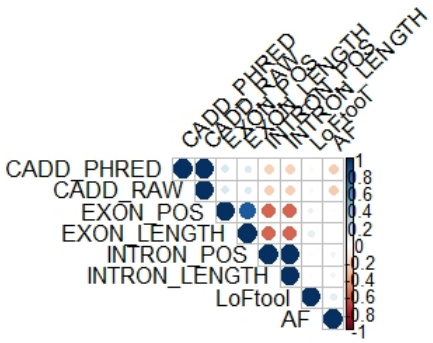
En effet, R ne pouvait pas performer certaines méthodes demandées pour un aussi grand nombre d'observations. Après une large discussion, nous nous sommes mis d'accord à utiliser l'une des méthodes d'échantillonage qui existe dans la littérature pour surmonter ce problème. L'idée est assez simple, nous considérons les 65000 observations comme une population et l'objectif était d'extraire un échantillon représentatif de taille fixée et suffisante pour que le traitement des méthodes sous R soit réalisable. La méthode d'échantillonage stratifiée consiste à retrouver dans l'échantillon les mêmes proportions pour chacune des strates (sous-groupes) selon les caractéristiques choisies pour l'étude dans la population visée. Dans notre contexte, c'est la classification des variants qui a été choisie comme caractéristique de l'étude et nous avons donc utiliser les proportions pour chacune des deux strates (variants cohéremment classés et variants conflictuellement classés) présentes dans la population pour générer un échantillon représentatif de cette dernière. Tout de même, une question est restée posée : l'échantillon généré était-il fidèlement représentatif de la population ? pour y répondre nous avons utilisé le test d'indépendance de khi-deux pour la plupart des variables. l'hypothèse H_0 : "Les deux échantillons sont indépendants" a été rejetée à tout les coups ce qui signifie que l'échantillon obtenu dépend de la population et est par conséquent fidèlement représentatif de celle-ci.

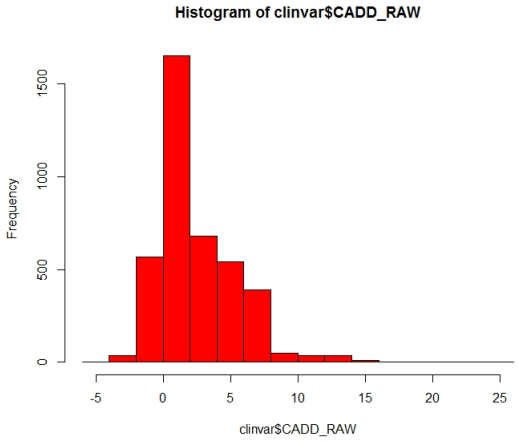
1.2 Analyses Descriptives

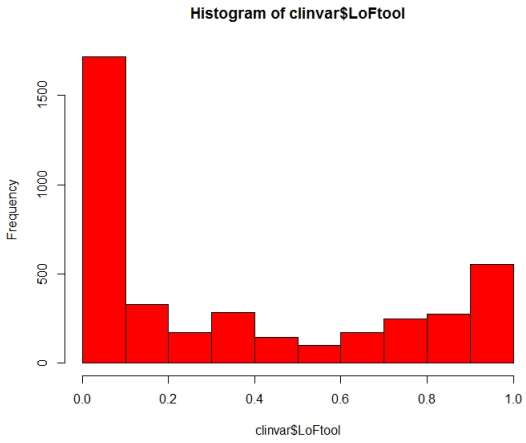
Caractéristiques numériques

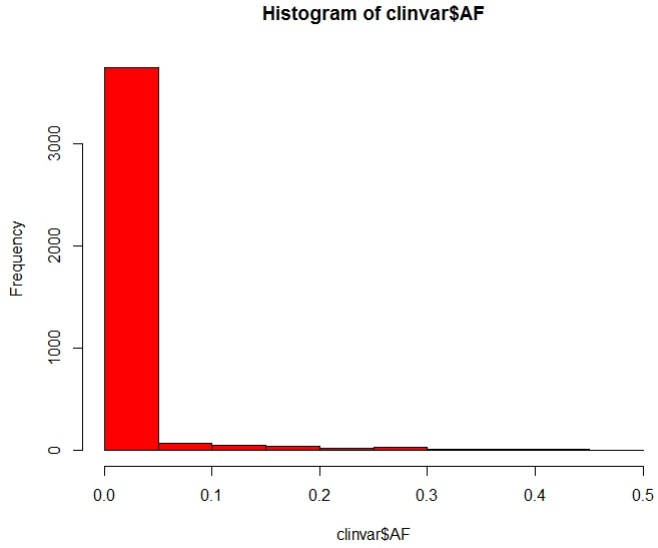
Les corrélations entre les variables numériques sont illustrées dans la figure suivante : Les deux types de score évaluant le caractère délétère d'un variant "CADD" sont très corrélés, de même pour le cas des variables liées aux exons/introns qui présentent aussi une corrélation. Nous avons donc réduit d'avantages le nombre de variables en enlevant l'une entre chacune des deux variables corrélées.
Tous les données numériques n'étaient pas gaussiennes, nous avons confirmé cela via le test de Shapiro-Wilk.

textrmLa seule variable numérique qui nous paraîssait très liée à la classification des

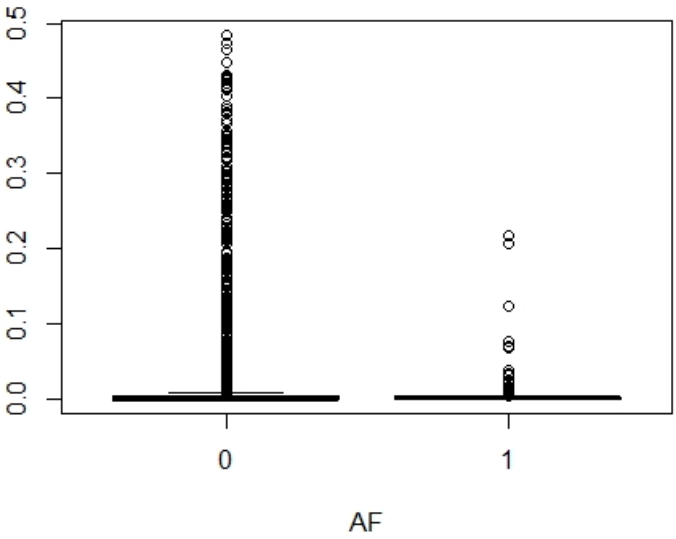

CADD_PHRED
CADD_RAW
EXON_POS
EXON_LENGTH
INTRON_POS
INTRON_LENGTH
LoFtool
AF
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


Histogram of clinvar$CADD_RAW
Frequency
0
500
1000
1500
-5
0
5
10
15
20
25
clinvar$CADD_RAW


Histogram of clinvar$LoFtool
Frequency
0
500
1000
1500
0.0
0.2
0.4
0.6
0.8
1.0
clinvar$LoFtool



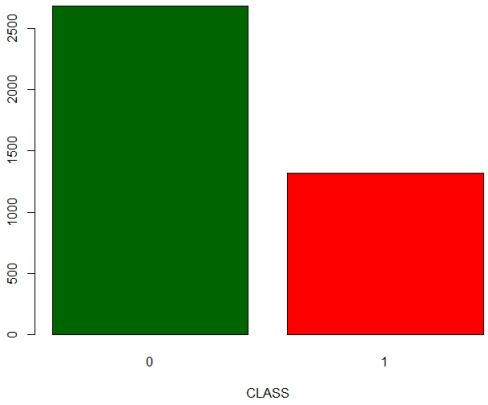
variants est "AF" : les fréquences alléliques des variants.



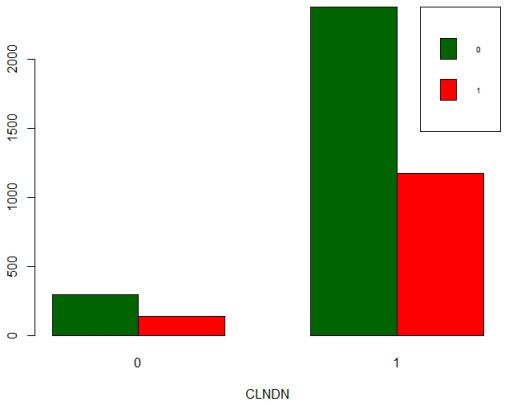
Caractéristiques catégoriques

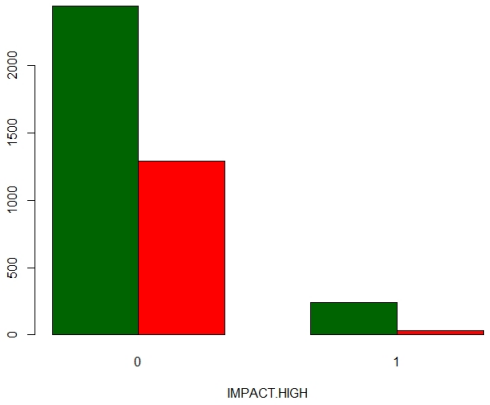
Nous avons remarqué que les deux sous groupes de variants (classés cohéremment et classés conflictuellement) sont déséquilibrés. les variants conflictuellement classés occupe

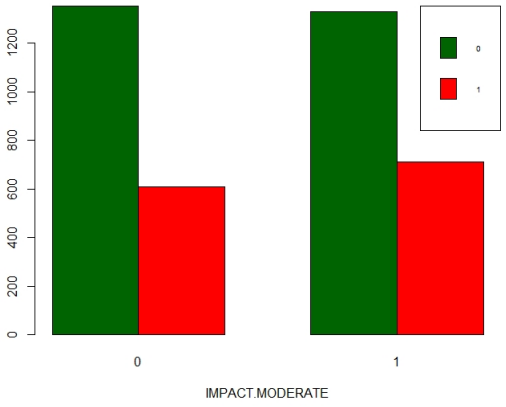
une proportion de 25%. Nous avons ensuite essayé de repérer les variables catégoriques



qui étaient les plus liées à notre variable cible ”CLASS”.




0
500
1000
1500
2000
0
1
IMPACT.HIGH


0
200
400
600
800
1000
1200
0
1
IMPACT.MODERATE

CHAPITRE 2

MÉTHODES

2.1 Sélection de variables

Pour choisir les variables les plus importantes dans le sens où elles sont fortement liées à la classification conflictuelle des variants, nous avons utilisés trois approches différentes :

— Stepwise model selection.

— Tests statistiques.

— Modèle de régression logistique pénalisée.

2.1.1 Stepwise model selection

Basée sur l'optimisation de l'un de ces existants critères de choix de modèles, la méthode permet de sélectionner automatiquement un modèle constitué de prédicteurs qui minimisent un critère dit : "Akaike Information Criterion" (le choix d'utilisateur par défault). Stepwise regression propose de choisir parmis trois d'approches de sélecyion de modèles : "Both elimination", "Forward selection" ou "Backward elimination".
Forward selection, l'itération initiale ne contient aucune variable et l'approche teste l'ajout de chaque variable à l'aide du critère choisi par défaut : "AIC".
Backward elimination, à l'itération initiale l'approche construit un modèle explicatif avec tous les prédicteurs. À chaque itération, elle teste l'élimination de chaque variable à l'aide du critère "AIC".

Both elimination, ce n'est qu'une combinaison des deux approches précédentes, testant donc à chaque itération les variables à inclure ou à exclure.
Dans les trois cas de la méthode *Stepwise regression*, l'approche choisie s'arrête lorsque le critère "AIC" est optimal ce qui veut dire que rajouter, éliminer ou rajouter/éliminer une variable entraînera l'augmentation de la valeur de l'AIC.

2.1.2 Tests statistiques

Les tests d'hypothèses statistiques nous permettent de déceler l'existence d'un potentiel lien significatif entre une variable quelconque et la variable cible "CLASS". Le test est choisie selon le type des variables et selon la nature de l'hypothèse H_0.
Les données numériques n'étaient pas gaussiennes. Pour voir s'il y a une différence significative entre les deux sous-groupes de la variable "CLASS", nous avons donc choisi le test de rangs signés de Wilcoxon-Mann-Whitney qui permet de comparer les médianes de ces deux sous-groupes.
Concernant les données catégorielles, les conditions pour appliquer le test de khi-deux étaient vérifiées. Naturellement, l'objectif est de vérifier à partir de tables de contingences s'il y a une différence significative entre les proportions des deux sous-groupes. Si elle existe, cette différence significative reflète un potentiel lien loin d'être dû au hasard entre une variable catégorielle et notre variable cible.

2.1.3 Modèle de régression logistique pénalisée

Nous avons trouvé dans la littérature que la méthode de régression logistique pénalisée avec l'une des approches : LASSO, RIDGE... pourrait être utilisée dans un contexte de sélection de variables pertinentes. Nous nous sommes appuyé sur la validation croisée pour construire notre modèle pénalisé avec l'approche LASSO, les étapes nécéssaires pour trouver le coefficient de pénalisation qui minimise l'erreur moyenne de validation croisée (CVAE) sont données comme suit :

1. Sélection aléatoire par défaut de 10 échantillons de validation croisée (*10 folds cross-validation*).

2. Fixer le critère d'erreur (*type measure*) qui est le plus spécifique aux régressions logistiques, dans notre cas c'est la mesure AUC qui indique l'aire sous la courbe ROC (à maximiser).
3. Le coefficient de pénalisation $\lambda_{min(CVAE)}$ est obtenu en cherchant à maximiser l'AUC (respectivement en minimisant le CVAE).

Les coefficients optimaux des variables sont ensuite obtenus en fixant $\lambda_{min(CVAE)}$ dans les contraintes. Les coeffcients nulles ou très proches de zéro sont non pertinents.

2.2 Méthodes de classification

Les méthodes choisies rentrent dans le cadre de l'apprentissage machine supervisé, qui se basent sur la sélection d'un échantillon d'apprentissage de taille assez suffisante (par exemple 75% de la taille totale) pour construire ensuite un modèle de prédiction sur ce dernier. L'échantillon restant est appelé échantillon de test qui sert à tester la performance des modèles construits.
La nature des données qui sont majoritairement catégorielles nous laissent peu de chances que certaines des méthodes de classifications puissent fournir des résultats cohérents.

2.2.1 Arbres de décision

Cette méthode permet de repérer, parmis les variables choisies, celles qui rapportent plus de discrimination entre les variants au sens de pouvoir les classer correctement. La méthode applique des tests successives à chaque observation en cherchant par exemple à vérifier si pour une certaine variable, sa valeur est inférérieur ou supérieur à un certain seuil. Les vérifications se réalisent au niveau de chaque branche de l'arbre jusqu'à arriver à l'une des extrémités de l'arbre dites : feuilles où les individus seront classés soit en classe : 0 soit en classe : 1.

2.2.2 Méthode des k plus proches voisins

L'algorithme des k plus proches voisins est une méthode non paramétrique utilisée majoritairement pour la classification. Il permet de classer une nouvelle entrée dans la classe

à laquelle appartient ces k plus proches voisins définis dans l'espace des caractéristiques (prédicteurs pertinents).
Le mécanisme de cette méthode est le plus simple en apprentissage machine, soit k un entier strictement positif et d une distance. Pour chaque nouvelle entrée, la classe qui lui sera attribuée est choisie dépendamment des k observations qui lui sont très plus proches en respectant la distrance d. Bien entendu, le choix de la métrique de la distance et le nombre de voisins k, joue un rôle primordial dans l'amélioration des taux de prédiction. Pour des données non continues (notre cas), on priviligie d'utiliser la métrique de distance de Hamming dite : métrique de recouvrement. En parallèle, le choix de k doit être optimisé car pour un nombre réduit ou très grand de voisins, la classification est moins précise entraînant par conséquent des taux d'erreur de prédiction élevés.

2.2.3 Méthode des machines à vecteurs de support

Les machines à vecteurs de support ou séparateurs à vastes marges sont des classificateurs qui reposent sur deux principes clés. Le premier principe est la notion de marge maximale, elle représente la distance maximale entre la frontière de séparation et les observations les plus proches qui permettent d'obtenir les vecteurs de support. Dans le cas d'un problème linéairement séparable, l'objectif est assez similaire comparée à celui de l'analyse discriminante linéaire, qu'est de trouver une fonction discriminante linéaire construite à partir d'une combinaison linéaire du vecteur d'entrées avec un vecteur de poids.
Afin de pouvoir traiter des cas plus complexes où les données ne sont pas linéairement séparables, le deuxième principe des SVM intervient. Il consiste à transformer l'espace de représentation des données d'entrées en un espace de plus grande dimension dans lequel il est très probable qu'il existe une séparation linéaire. Ceci est réalisé grâce à une fonction noyau qui permet de transformer un produit scalaire dans un espace de grande dimension.

Choix de la fonction noyau

Le théorème de Mercer exige des conditions pour qu'une fonction soit considérée : fonction noyau. Elle doit être symétrique et semi définie positive.

L'exemple le plus simple de fonction noyau est le noyau linéaire :

$$K(x_i, x_j) = x_i^t.x_j$$

On se ramène donc au cas d'un classifieur linéaire sans changement d'espace. Ce qui également peut paraître une excellente idée pour évaluer la difficulté d'un problème de classification.

Il existe plusieurs noyaux, nous allons en citer trois autres très usuellement utilisés :

— Le noyau gaussien : $K(x, z) = \exp(\frac{-\|x-z\|^2}{2\sigma^2})$

— Le noyau polynomial : $K(x, z) = (x^T.z + 1)^d$

— Le noyau sigmoïde : $K(x, z) = \tanh(\alpha x^t z + c)$

— Le noyau Radial : $K(x, z) = \exp(-\gamma\|x - z\|^2)$

2.2.4 Méthode des réseaux de neurones

Un réseau de neurone artificiel est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques. En effet, un neurone artificiel est conçu comme un automate doté d'une fonction de transfert qui transforme ses entrées en sortie selon des règles précises. Par exemple, un neurone somme ses entrées, compare la somme résultante à une valeur seuil et répond en émettant un signal si cette somme est supérieure ou égale à ce seuil. L'efficacité de la transmission des signaux d'un neurone à l'autre peut varier : on parle de "poids synaptiques". Ces derniers peuvent être optimisés par des règles d'apprentissage de telle sorte que le label prédit correspond au réel label.

Perceptron simple monocouche

Un réseau de neurone artificiel qui permet de traiter des problèmes de classification à deux classes possède naturellement un seul neurone de sortie. Le perceptron est associé à une règle d'apprentissage qui lui permet de déterminer automatiquement à travers un certain nombre d'itérations, les poids synaptiques optimaux qui assurent la séparation linéaire d'un problème d'apprentissage supervisé. Étant la forme la plus simple d'un réseau de neurones, si le problème n'est pas linéairement séparable, la structure du perceptron simple ne permet pas de trouver une séparatrice entre les deux classes.

Perceptron multicouches

Contrairement à son analogue monocouche, le perceptron multicouches est capable via son mécanisme de rétropropagation du gradient de l'erreur, de séparer non linéairement deux classes. Les neurones d'une couche sont reliés à la totalité des neurones des couches adjacentes. Ces liaisons sont soumises à un coefficient altérant l'effet de l'information sur le neurone de destination. Ainsi, le poids de chacune de ces liaisons est l'élément clef du fonctionnement du réseau. Bien évidemment, la mise en place d'un perceptron multicouches pour résoudre un problème requiert la détermination des poids synaptiques optimaux applicables à chacune des connexions inter-neuronales.

2.2.5 Analyse discriminante

L'analyse discriminante est une technique statistique qui vise à expliquer et, dans notre contexte, prédire l'appartenance des observations (les variants génétiques) à l'une des deux classes : "conflictuelle" ou "cohérente" en se basant sur leurs caractéristiques les plus importantes.

Analyse discriminante linéaire

En se basant sur l'hypothèse d'homoscédasticité des données, ce qui veut dire que les deux groupes d'observations sont supposés d'avoir la même matrice de variances-covariances et qui peut être également reflétés par la présence de deux nuages de points de même forme et volume dans l'espace de représentation.
L'hypothèse d'homoscédasticité, la méthode d'analyse discriminante linéaire permet de fournir une séparation linéaire entre les deux classes en maximisant la distance entre le séparateur linéaire et les barycentres des classes.

Analyse discriminante quadratique

Contrairement à son analogue mentionné précédemment (LDA), la méthode d'analyse discriminante quadratique (QDA) suppose que chaque classe a sa propre matrice de variances-covariances. En d'autres termes, les variables prédictives ne sont pas supposés

avoir la variance commune à chacun des deux classes dans la variable cible. Par conséquent, la méthode QDA dépasse les limites de décision linéaires car elle est capable de capturer les différentes variances-covariances et de fournir une classification non forcément linéaire.

2.2.6 Régression logistique

La régression logistique est un modèle de régression binomiale gérant uniquement les problèmes à deux classes. Le modèle cherche à associer à un vecteur d'entrées mesurées sur l'ensemble des prédicteurs, des coefficients optimaux de telle sorte que pour chaque nouvelle entrée, sa classe sera prédite. En fixant un seuil de décision τ et pour pouvoir prédire la classe d'une entrée, le modèle utilise la fonction de répartition d'une loi logistique standard de support infini. L'objectif est de trouver la valeur du seuil de décision qui améliore le taux de prédiction.

CHAPITRE 3

RÉSULTATS

3.1 La combinaison des variables pertinentes

La méthode de *Stepwise model selection* a convergé vers un modèle presque identique dans les trois cas de direction choisies. De plus, la valeur d'AIC obtenu était identique (AIC = 4678.02845), la combinaison des variables trouvée est donnée comme suit :

```
1  CHROM.11  CHROM.12  CHROM.16
2  CHROM.17  CHROM.19  CHROM.2  CHROM.3  CHROM.5
3  ALT.A  ALT.C  ALT.G  ALT.T
4  CLNVC.Deletion  CLNVC.single.nucleotide.variant
5  Consequence.frameshift.variant  Consequence.intron.variant
6  Consequence.synonymous.variant
7  IMPACT.HIGH  IMPACT.MODERATE
8  PolyPhen.benign  PolyPhen.possibly.damaging
9  AMINO.ACID.ALT.A  AMINO.ACID.ALT.E  AMINO.ACID.ALT.G
10 AMINO.ACID.ALT.M  AMINO.ACID.ALT.N
11 AMINO.ACID.ALT.NS  AMINO.ACID.ALT.R
12 STRAND
13 CADD.RAW
14 AF
```

```
15 EXON_LENGTH
```

En ce qui concerne les tests d'hypothèses statitiques, nous avons fixé un seuil de signification de 5% et nous avons obtenus les variables les plus significativement liées à la classification des variants :

ID_VAR	NAMES_VAR	P_VALUE
1	CLASS	0.000000e+00
37	IMPACT.HIGH	1.114689e-17
32	Consequence.frameshift_variant	1.033907e-10
46	PolyPhen.probably_damaging	1.555198e-09
35	Consequence.stop_gained	7.251183e-08
47	AMINO_ACID_ALT...1	7.251183e-08
71	AF	3.347042e-07
44	PolyPhen.benign	5.240703e-06
66	AMINO_ACID_ALT.X	9.178258e-05
5	CHROM.11	2.042996e-04
31	CLNVC.single_nucleotide_variant	3.010166e-03
15	CHROM.5	3.045971e-03
60	AMINO_ACID_ALT.P	5.938509e-03
11	CHROM.17	6.684985e-03

ID_VAR	NAMES_VAR	P_VALUE
7	CHROM.13	9.680803e-03
8	CHROM.14	1.050328e-02
39	IMPACT.MODERATE	1.057124e-02
69	CADD_RAW	1.495613e-02
34	Consequence.missense_variant	1.546702e-02
17	CHROM.7	1.634286e-02
59	AMINO_ACID_ALT.NS	1.862713e-02
63	AMINO_ACID_ALT.S	1.952596e-02
9	CHROM.15	2.039094e-02
72	EXON_LENGTH	2.756295e-02
62	AMINO_ACID_ALT.R	3.014296e-02
67	STRAND	3.035430e-02
30	CLNVC.Deletion	3.161104e-02
24	REF.MN	3.463294e-02

Après analyse des résultats retournés par le modèle de régression logistique pénalisée, nous avons fixé une valeur absolue de seuil de coefficient à 0.5. L'idée de fixer une si petite valeur de seuil était justifiée par le fait que nous ne voulions pas perdre beaucoup de variables. En se basant sur ce qui est indiqué précédemment, les prédicteurs les plus pertinents sont donnés comme suit :

```
1 AF
2 PolyPhen.probably.damaging
3 IMPACT.MODIFIER    IMPACT.HIGH    IMPACT.LOW
4 Consequence.frameshift.variant      Consequence.intron.variant
5 Consequence.synonymous.variant
6 CLNVC.single.nucleotide.variant
7 CHROM.5  CHROM.17  CHROM.11
```

L'unique variable qui possède une valeur de coefficient dominante est la variable "AF"

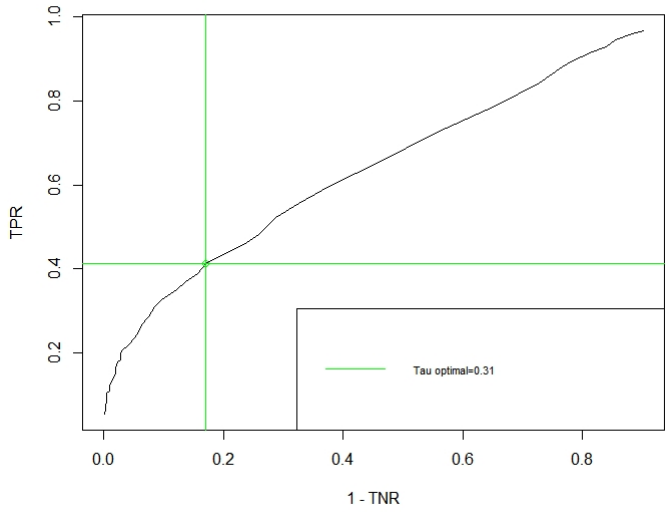
(coef = -31.2) ce qui semble paraître comme étant un excellent prédicteur pour nos modèles de classification. Par contre, les contributions du restant des variables sont assez équilibrées entre elles car leurs coefficients estimés ne dépassent pas en valeur absolue la valeur de 1.

Selon les résultats obtenus, nous nous sommes mis d'accord de choisir les variables les plus pertinentes et en commun entre les trois approches de sélection de modèle. La première variable qui était candidate est bien évidemment la variable fréquence allélique des variants "AF". Puisque certaines de ces variables étaient des modalités pour les variables sources (initiales, avant dichotomisation), il était nécessaire de fournir aussi leurs interprétations. Le tableau suivant résume les variables pertinentes choisies est leurs significations :

3.2 Application des méthodes de classification

3.2.1 Régression logistique

L'objectif initiale était d'utiliser les courbes ROC pour trouver la valeur optimale du seuil de décision τ_{opt}. En effet, chercher à maximiser la sensibilité tout en minimisant l'inverse de la spécificité revient à chercher à minimiser le taux d'erreur. La figure suivante nous indique la valeur optimale de τ trouvée : La règle de prédiction est donnée comme



Variables AF	Significations Fréquences Alléliques des variants
PolyPhen.probably.damaging	L'outil Polymorphism Phenotyping est un prédicteur d'impact d'une substitution d'acides aminées sur la structure et la fonction d'une protéine humaine.
CADD.RAW	Score brut de prédiction du caractère délétère des variants.
CHROM.17	Chromosome 17
CHROM.5	Chromosome 5
CHROM.11	Chromosome 11
IMPACT.HIGH	Variable qui présume l'impact d'un variant basée sur l'étude ontologique de leurs séquences nucléotidiques.
Consequence.frameshift.variant	Un variant de séquence qui perturbe le cadre de lecture de la traduction, car le nombre de nucléotides insérés ou supprimés n'est pas un multiple de trois.
CLNVC.single.nucleotide.variant	Le type de variant. (survenu par un changement d'un seul nucléotide)

TABLE 3.1 –

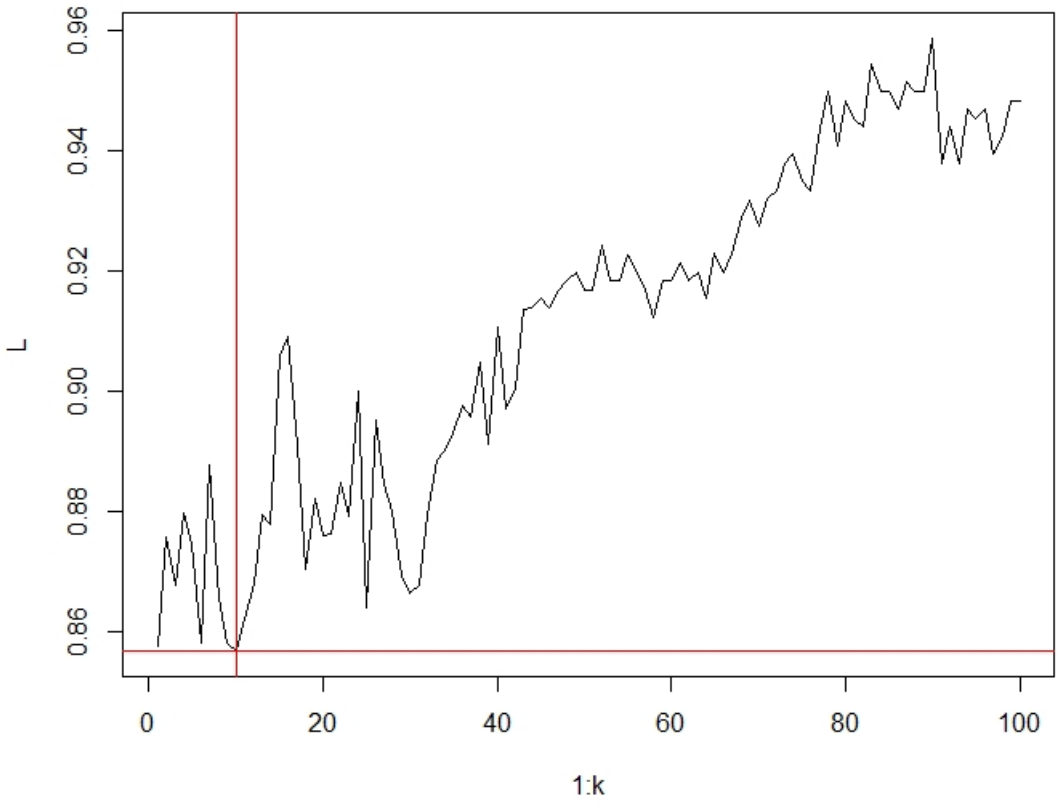
suit :

$$F(\beta^t X^i) > 0.31 \Rightarrow \hat{Y}_i = 1; F(\beta^t X^i) \leq 0.31 \Rightarrow \hat{Y}_i = 0.$$

telle que : $F(x) = \frac{e^x}{1+e^x}$

3.2.2 La méthode des k plus proches voisins

Avant d'appliquer la méthode par défaut (un seul voisin), nous voulions trouver la valeur optimale de k. La démarche restait la même que celle utilisée lors de la régression logistique qu'est d'optimiser la courbe ROC. Néanmoins, cette fois ci nous avons choisi d'illustrer l'evolution du taux d'erreur en fonction de k : Nous pouvons conclure que

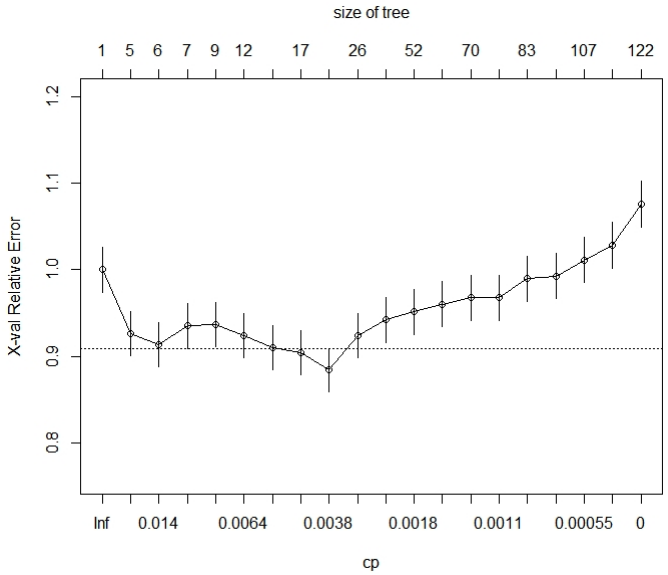


pour optimiser le taux de prédiction (minimiser le taux d'erreur) il faudrait considérer 10 voisins. Nous avons donc appliqué la méthode KNN avec $k = 10$.

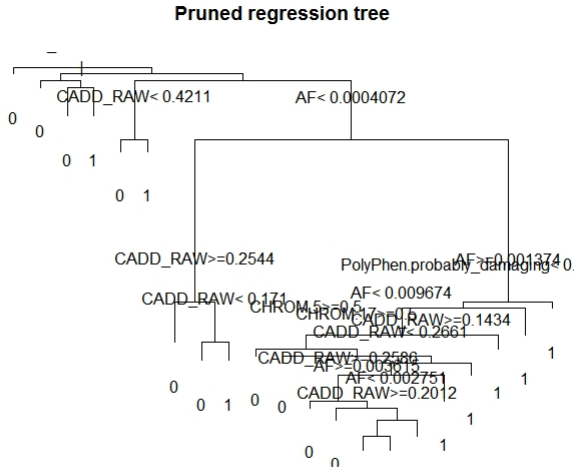
3.2.3 Les arbres décisionnels

L'arbre maximal (obtenu par un coefficient de complexité égal à zero) n'était pas celui qui fournit de meilleurs résultats. Néanmoins, nous avons pu voir, à travers la fonction *summary*, que les variables les plus importantes étaient "AF" et "CADD.RAW" utilisées principalement dans la construction de l'arbre maximal. D'après quelques articles trouvés dans la littérature, il est possible que l'arbre maximal ne soit pas le meilleur fournisseur de résultats s'il est généré par un surapprentissage de l'échantillon d'antrainement. De plus, l'arbre maximal était pratiquement illisible.

Une solution pour détourner ce genre de problème est de couper l'arbre maximal en cherchant la valeur optimale du paramètre de complixité *cp* qui minimise l'erreur de validation croisée (*xerror*). Autrement dit, la procédure permet simplement de couper certaines branches qui ne sont pas significatives dans le tri des variants ce qui pourrait améliorer d'un côté la lisibilité de l'arbre et d'un autre côté, le problème de surapprentissage pourrait être résolu. Le coefficient de complexité optimal etait égal à : 0.003. Il nous a permis



d'obtenir un arbre coupé plus lisible et plus performant.



3.2.4 Les machines à vecteurs de support

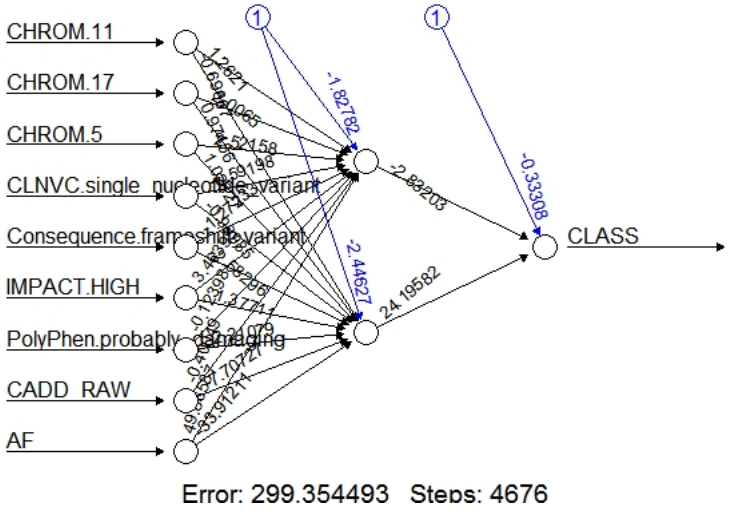
Nous avons remarqué d'après les résultats obtenus durant l'application du modèle LDA, que les données sont loins d'être linéairement séparables. Ce qui pourrait éloigner l'hypothèse que SVM pourrait utiliser le premier principe pour traiter ce genre de problèmes. Nous nous sommes intérrogés donc sur quelle noyau nous pourrions construire notre modèle. Pour cela, nous avons chercher à maximiser le taux de prédiction grâce à un petit programme qui parcours les noyaux qui sont proposés dans la partie méthodes. Pour plusieurs valeurs de coûts, nous retenons à chaque sous itération la valeur du taux d'erreur global pondéré. Les résultats peuvent être résumés dans les tableaux suivants :

	linear	radial	polynomial	sigmoid
1	0.75	0.6576515792	0.6549382136	0.5059156966
2	0.75	0.6561432836	0.6553152875	0.4826031303
3	0.75	0.6583688029	0.6549382136	0.4974276175
4	0.75	0.6572745053	0.6530897682	0.5635285614
5	0.75	0.6561802078	0.6534668421	0.5647783880
6	0.75	0.6569343556	0.6534668421	0.5688523526
7	0.75	0.6554629841	0.6534668421	0.5669669831
8	0.75	0.6554629841	0.6534668421	0.5688523526
9	0.75	0.6554629841	0.6534668421	0.5673440570
10	0.75	0.6558400580	0.6538439160	0.5717950956

Le noyau choisi était : "sigmoid" avec un coût égal à 2.

3.2.5 Le perceptron multicouches

Au niveau des couches d'entrées du perceptron multicouches, on voit que le plus grand poids synaptique est attribué à la variable "AF". La figure suivante illustre le schéma du perceptron qui a été utilisé :



Le perceptron illustré est composé de deux couches : une couche cachée composée de deux neurones et une couche de sortie qui retourne le label prédit. Initialement, les poids synaptiques sont fixés aléatoirement dans les deux couches. Chaque neurone dans une couche quelconque associe un poids synaptique à la variable correspondante. Ensuite, après sommation de toutes les entrées pondérés, il exécute la fonction d'activation (par défaut : la fonction sigmoïde) qui retourne une valeur. Cette dernière correspond au label prédit si le neurone en question le neurone de sortie. Pour trouver la combinaison des poids optimaux qui maximise le taux de prédiction, le perceptron multicouche utilise un algorithme dit de rétropropagation des erreurs. Il consiste à calculer à chaque itération l'erreur de prédiction au niveau du neurone de sortie ensuite cette erreur est rétropropagé aux neurones précédents (neurones se trouvant dans les couches cachées) pour pouvoir permettre le calcul de l'erreur de prédiction au niveau de chaque neurone de couches cachées. Une fois les erreurs sont calculées, les poids sont corrigés par la méthode de descente du gradient en utilisant un taux d'apprentissage et les erreurs correspondantes.

3.3 Résultats des méthodes de classification

Méthodes	Taux d'erreur pondéré	erreur $2^{\text{ème}}$ espèce	Temps de calcul
Logistic Regression	0.68	0.90	0.09
LDA	0.67	0.87	0.05
QDA	0.25	0.09	0.05
Les plus proches voisins	0.54	0.64	0.22
Arbres décisionnels	0.44	0.35	0.34
SVM	0.48	0.52	5.5
Perceptron multicouches	0.42	0.49	116

TABLE 3.2 –

Malgré l'injection des variables les plus significativement pertinentes dans nos modèles de classification, nous avons remarqué que le taux d'erreur est assez élevé dans la plupart des méthodes, notamment celle de la régression logistique. La seule analyse qui pourrait nous aider à trouver un modèle plus adéquat est celle de chercher celui qui offre un taux d'erreur global pondéré ainsi qu'une erreur de $2^{\text{ème}}$ espèce minimaux. Le meilleur candidat était le modèle d'analyse discriminante quadratique. Néanmoins, malgré la grande taille d'échantillon, il s'avère peu fiable de l'utiliser car à la base les modèles de LDA et QDA sont utilisés dans un contexte où l'on a des données gaussiennes. Les modèles qui peuvent devenir aussi de potentiels candidats sont : Le perceptron multicouche et les arbres décisionnelles. Ces derniers n'avancent aucun a priori sur la nature des données et il est également possible de pouvoir améliorer leur prédiction en modifiant leur structure. Le désavantage lié au perceptron multicouches est l'énorme temps de calcul qu'il met pour exécuter.

CONCLUSION

L'objectif principal de cette étude était d'essayer de déchiffrer avec des méthodes d'apprentissage machine le phénomène qui pousse les généticiens à tomber dans un conflit de classification des variants génétiques humains. Nous avons vu initialement grâce à l'analyse descriptive que la fréquence allélique des variants pourrait être le principal facteur créant confusion chez ces spécialistes, car la plupart des variants conflictuellement classés possèdent une fréquence très proches de zéro ce qui laisse à penser que ces derniers sont extrêmement rares et que les outils d'interprétation utilisés ne suffisent pas à aider dans la prise de décision correcte et commune entre tous les laboratoires. Le test de Mann-Whitney et les méthodes de régression appliquées dans un contexte de sélection de variables étaient tous une confirmation pour nous que la variable fréquence allélique (AF) était certainement celle la plus significative. De plus, il y avait aussi d'autres variables significatives qui sont de potentiels prédicteurs pour la classification conflictuelle, notamment le score délétère des variants (CADD) et la traduction de la forme phynotypique du protéine associée au variant(PolyPhen). Plusieurs méthodes d'apprentissage machine ont été appliquées dans le but d'essayer de prédire à base des variables significatives obtenues, si une classification conflictuelle sera attribuée ou pas à un nouvel variant. Malheureusement, la source de données était assez complexe à traiter et non exploitable sur logiciel R car le niveau des données était énorme. Nous avons posés trois hypothèses sur la qualité des résultats de prédiction obtenus qui été loins d'être satisfaisant, les hypothèses sont données comme suit :

— Lors du traitement de la base de données et durant la modification des variables, de potentielles informations pourraient être disparues.
— Selon leurs dispersions, les données manquantes été loin d'être dû au hasard ce qui crée des biais lors de l'imputation multiple par équations chaînées.
— l'énorme déséquilibre entre les deux groupes pourrait provoquer un phénomène de surapprentissage suspecté à partir des résultats liées aux arbres décisionnels.

La méthode des réseaux de neurones pourrait être la plus adaptée à ce genre de problème, car elle possède deux particularités essentielles : elle permet de traiter n'importe quelle type et taille de données, elle offre comme deuxième particularité, plus de flexibilité comparée aux autres méthodes car l'utilisateur peut modifier la structure de son réseaux de neurones à sa guise et selon la forme des données. Le seul désavantage de cette méthode est qu'elle considère un temps de calcul considérable.

Annexe

Redéfinition de la base de données

R

```
############ LIBRARIES USED ###########
library(corrplot)
library(questionr)
library(glmnet)
library(mice)
library(DescTools)
library(MASS)
library(data.table)
library(e1071)
library(dplyr)
library(corrplot)
library(DMwR)
library(rpart)
library(neuralnet)
library(class)
############## IMPORTING CLINVAR DATA SET ##########

setwd(dir = "D:/M2 Project")
clinvar = read.csv("clinvar.csv",
na.strings=c("","NA"),header=T,stringsAsFactors = F)
clinvar = subset(clinvar,clinvar$CHROM!="MT")
summary(clinvar)

```

```
24 ############### CLEANING DATA SET ##################
25 r = c()
26 for (i in 1:length(clinvar)){
27   if(sum(is.na(clinvar[i])) > 65000){
28     print(sum(is.na(clinvar[i])))
29     r=c(r,i)
30   }
31 }
32 clinvar = clinvar[-r]
33
34 # Checking for redundant non useful variables
35 n=c("POS","CLNDISDB","CLNHGVS","CLNVI",
36 "MC","Allele","SYMBOL","Feature","Feature_type")
37 l=c()
38 k=1
39 while(k<length(n)+1){
40 for(i in 1:length(clinvar)){
41   if (names(clinvar[i])==n[k]){
42     l=c(l,i)
43   }
44 }
45   k=k+1
46 }
47 clinvar = clinvar[-l]
48
49 ############ MODALITIES ISSUE #############
50
51  for(j in c(2,3)){
52  for(i in 1:dim(clinvar)[1]){
53    if(nchar(clinvar[i,j])>1){
```

```
    clinvar[i,j]="MN"
     }
  }
  }
AF = (clinvar$AF_ESP+clinvar$AF_EXAC+clinvar$AF_TGP)/3
clinvar = clinvar[,-c(4,5,6)]

 for(i in 1:dim(clinvar)[1]){
 if(clinvar[i,"CLNDN"]=="not_specified" |
 clinvar[i,"CLNDN"]=="not_provided" |
    clinvar[i,"CLNDN"]=="not_specified|not_provided"|
        clinvar[i,"CLNDN"]=="not_provided|not_specified"){
      clinvar[i,"CLNDN"]="0"
    }else{
      clinvar[i,"CLNDN"]="1"
    }
 }

clinvar$Consequence[grepl("synonymous_variant",
clinvar$Consequence) == T] = "synonymous_variant"
clinvar$Consequence[grepl("missense_variant",
clinvar$Consequence) == T] = "missense_variant"
clinvar$Consequence[grepl("intron_variant",
clinvar$Consequence) == T] = "intron_variant"
clinvar$Consequence[grepl("3_prime_UTR_variant",
clinvar$Consequence) == T] = "3_prime_UTR_variant"
clinvar$Consequence[grepl("5_prime_UTR_variant",
clinvar$Consequence) == T] = "5_prime_UTR_variant"
clinvar$Consequence[grepl("frameshift_variant",
clinvar$Consequence) == T] = "frameshift_variant"
```

```
84  clinvar$Consequence[grepl("inframe_deletion",
85  clinvar$Consequence) == T] = "inframe_deletion"
86  clinvar$Consequence[grepl("inframe_insertion",
87  clinvar$Consequence) == T] = "inframe_insertion"
88  clinvar$Consequence[grepl("intron_variant",
89  clinvar$Consequence) == T] = "intron_variant"
90  clinvar$Consequence[grepl("splice_acceptor_variant",
91  clinvar$Consequence) == T] = "splice_acceptor_variant"
92  clinvar$Consequence[grepl("splice_donor_variant",
93  clinvar$Consequence) == T] = "splice_donor_variant"
94  clinvar$Consequence[grepl("start_lost",
95  clinvar$Consequence) == T] = "start_lost"
96  clinvar$Consequence[grepl("stop_gained",
97  clinvar$Consequence) == T] = "stop_gained"
98  clinvar$Consequence[grepl("stop_lost",
99  clinvar$Consequence) == T] = "stop_lost"
100
101
102  EXON_POS = sub("\\/.*", "", clinvar[,"EXON"])
103  EXON_LENGTH = sub(".*\\/", "", clinvar[,"EXON"])
104
105  INTRON_POS = sub("\\/.*", "", clinvar[,"INTRON"])
106  INTRON_LENGTH = sub(".*\\/", "", clinvar[,"INTRON"])
107
108  EXON_POS=ifelse(is.na(EXON_POS),"0",EXON_POS)
109  EXON_LENGTH=ifelse(is.na(EXON_LENGTH),"0",EXON_LENGTH)
110  INTRON_POS=ifelse(is.na(INTRON_POS),"0",INTRON_POS)
111  INTRON_LENGTH=ifelse(is.na(INTRON_LENGTH),"0",
112  INTRON_LENGTH)
113  EXON_POS = as.integer(EXON_POS)
```

```
EXON_LENGTH = as.integer(EXON_LENGTH)
INTRON_POS = as.integer(INTRON_POS)
INTRON_LENGTH = as.integer(INTRON_LENGTH)

CODON_ALT = sub(".*\\/", "", clinvar[,"Codons"])
CODON_REF = sub("\\/.*", "", clinvar[,"Codons"])
CODON_REF[clinvar$REF=="MN"]="MC"
CODON_ALT[clinvar$ALT=="MN"]="MC"


AMINO_ACID_ALT = sub(".*\\/", "", clinvar[,"Amino_acids"])
AMINO_ACID_ALT = ifelse(nchar(AMINO_ACID_ALT)>1,
"NS",AMINO_ACID_ALT)


r=c("EXON","INTRON","cDNA_position","Amino_acids","Codons")
l=c()
k=1
while(k<length(r)+1){
  for(i in 1:length(clinvar)){
    if (names(clinvar[i])==r[k]){
      l=c(l,i)
    }
  }
  k=k+1
}
clinvar = clinvar[-l]


```

```
144
145 ############ DEALING WITH MISSING DATA ###############
146
147  md.pattern(clinvar)
148  imputed = mice(clinvar, method="pmm", m=2, maxit=2)
149  imputed_clinvar = complete(imputed)
150  md.pattern(imputed_clinvar)
151  sapply(imputed_clinvar, function(x) sum(is.na(x)))
152  for(i in 1:length(imputed_clinvar)){
153     print(class(imputed_clinvar[,i]))
154  }
```

Sélection des variables et application des méthodes de classification

R

```
1  #RANDOM
2  set.seed(1)
3  temp1 = clinvar[clinvar$CLASS==1,]
4  temp2 = clinvar[!clinvar$CLASS==1,]
5  sm1 = sample_n(temp1,1320)
6  sm2 = sample_n(temp2,2680)
7  clinvar_smp = rbind(sm1,sm2)
8   library(ade4)
9   dichot = acm.disjonctif(subset(clinvar, select=1))
10  genvar = cbind(clinvar[,-1], dichot)
11  library(corrplot)
12 clinvar[,1:4]=clinvar[,1:4]-1
13
```

```
14 X = as.matrix(clinvar[,c(126:131)])
15 mcor = cor(X,method = "spearman")
16 corrplot(mcor, type="upper",
17 order="hclust", tl.col="black", tl.srt=45)
18 #### MODIFICATION DE LA NOUVELLE BASE
19 j=c()
20 for(i in 1:124){
21   if(sum(clinvar[,i])<100 |
22   sum(clinvar[,i])>3900){
23     j=c(j,i)
24   }
25
26 }
27 clinvar = clinvar[,-j]
28
29
30 for(i in 1:length(clinvar)){
31   clinvar[,i]=as.numeric(clinvar[,i])
32 }
33 for(i in 1:length(clinvar)){
34   print(class(clinvar[,i]))
35 }
36
37 X = as.matrix(clinvar[,67:73])
38 mcor = cor(X,method = "spearman")
39 corrplot(mcor, type="upper", order="hclust",
40 tl.col="black", tl.srt=45)
41
42 ### Selection de variables
43 ########### LASSO
```

```
y=clinvar[,"CLASS"]
x=model.matrix(~.,data = clinvar[,-1])

model_lasso = cv.glmnet(x, y, family="binomial",
 alpha = 1, nfolds = 10, type.measure = "auc")
cf_lasso = coef(model_lasso, s=model_lasso$lambda.min)
s = summary(cf_lasso)

##### LOGISTIC REGRESSION
logis_model = glm(as.factor(CLASS)~.,
family=binomial(link = "logit"),data=clinvar)
summary(logis_model)

##### STEP AIC
step.both = stepAIC(logis_model, direction = "both")
step.both$anova
step.both$aic


step.backward=stepAIC(logis_model,direction="backward")
step.backward$anova
step.backward$aic

logis_model1 = glm(as.factor(CLASS)~1,
 data=clinvar, family=binomial(link = "logit"))
step.forward=stepAIC(logis_model1,
direction="forward",scope=list(upper=logis_model,
lower=logis_model1))
step.forward$anova
step.forward$aic
```

```
74
75 #TESTS
76 num=67:73
77 fac=1:66
78 test_data = data.frame()
79 df1 = data.frame(ID_VAR=num,NAMES_VAR=names(clinvar[num]),
80 P_VALUE=numeric(length(num)),stringsAsFactors = FALSE)
81 df2 = data.frame(ID_VAR=fac,NAMES_VAR=names(clinvar[fac]),
82 P_VALUE=numeric(length(fac)),stringsAsFactors = FALSE)
83 w=k=c()
84 for(i in num){
85   w=c(w,wilcox.test(clinvar[,i]~as.factor(clinvar[,1])
86   ,data = clinvar)$p.value)
87 }
88 df1[,3]=w
89 for(i in fac){
90   k=c(k,chisq.test(table(as.factor(clinvar[,i]),
91   as.factor(clinvar[,1])))$p.value)
92 }
93 df2[,3]=k
94
95 test_data = rbind(df1,df2)
96 View(test_data[test_data$P_VALUE<0.05,])
97
98 #### Machine Learning
99 #Training and Test Data
100 rownames(clinvar) = 1:4000
101 set.seed(1)
102 train.data = sample_n(clinvar[,c("CLASS","CHROM.11",
103 "CHROM.17","CHROM.5","CLNVC.single_nucleotide_variant",
```

```
  "Consequence.frameshift_variant","IMPACT.HIGH",
  "PolyPhen.probably_damaging",
  "CADD_RAW","AF")],3000)
test.data = clinvar[-as.integer(rownames(train.data)),
c("CLASS","CHROM.11","CHROM.17","CHROM.5",
"CLNVC.single_nucleotide_variant",
"Consequence.frameshift_variant","IMPACT.HIGH",
"PolyPhen.probably_damaging",
"CADD_RAW","AF")]



standardize = function(x){ #re-scaling to [0,1]
  x = ((x-min(x))/(max(x)-min(x)))
}
train.clinvar_scaled =
as.data.frame(apply(train.data,2,standardize))
test.clinvar_scaled =
as.data.frame(apply(test.data,2,standardize))

### NEURAL NETWORK
ti = proc.time()
nn = neuralnet(CLASS~CHROM.11 + CHROM.17 + CHROM.5 +
CLNVC.single_nucleotide_variant +
Consequence.frameshift_variant + IMPACT.HIGH +
PolyPhen.probably_damaging +
 CADD_RAW + AF,
  data=train.clinvar_scaled,
  hidden=c(2,3), linear.output=FALSE, threshold=0.01)
proc.time() - ti
```

```
#nn = neuralnet(CLASS~AF, data=train.clinvar_scaled,
hidden=2, linear.output=FALSE, threshold=0.01)

plot(nn)
nn$result.matrix
print(nn)

nn.results = compute(nn, test.clinvar_scaled[,"AF"])
results = data.frame(actual = test.clinvar_scaled[,1],
prediction = nn.results$net.result)

roundedresults = sapply(results,round,digits=0)
roundedresultsdf = data.frame(roundedresults)
attach(roundedresultsdf)

table.nn = table(pred = prediction,true = actual)
alpha = table.nn[2,1]/sum(table.nn[,1])
beta = table.nn[1,2]/sum(table.nn[,2])
L_w = 1/4*(alpha) + 3/4*(beta)


########### Support Machine Vector ############

L_min = 2
TPR = matrix(NA, ncol=4, nrow=10)
TNR = matrix(NA, ncol=4, nrow=10)
noyau = c("linear", "radial", "polynomial", "sigmoid")
error = matrix(NA, ncol=4, nrow=10)
colnames(error) = noyau
for (j in 1:4){
```

```
  for (i in 1:10){
    svm.model = svm(as.factor(CLASS)~CHROM.11 +
    CHROM.17 + CHROM.5 + CLNVC.single_nucleotide_variant +
     Consequence.frameshift_variant + IMPACT.HIGH +
     PolyPhen.probably_damaging +
     CADD_RAW + AF, data = train.clinvar_scaled,
     cost = i,kernel = noyau[j])

    svm.pred = predict(svm.model,
    test.clinvar_scaled[,-1])
    T.svm = table(pred = svm.pred,
    true = test.clinvar_scaled[,1])
    TPR[i,j]=T.svm[1,1]/sum(T.svm[,1])
    TNR[i,j]=T.svm[2,2]/sum(T.svm[,2])
    alpha = 1 - TPR[i,j]
    beta = 1 - TNR[i,j]
    L = 1/4*(alpha) + 3/4*(beta)
    error[i,j] = L
    if (L_min > L) {
    L_min = L
    alpha_min = alpha
    beta_min = beta
    T_opt = T.svm
    kern_opt = noyau[j]
    cout_opt = i}
  }
}
which.min (error)
TPR + TNR
alpha_min
```

```
beta_min
L_min
T_opt
kern_opt
cout_opt
################## Support Machine Vector #################
ti = proc.time()
svm.model = svm(as.factor(CLASS)~CHROM.11 + CHROM.17 +
CHROM.5 + CLNVC.single_nucleotide_variant +
Consequence.frameshift_variant + IMPACT.HIGH +
PolyPhen.probably_damaging +
CADD_RAW + AF, data = train.clinvar_scaled,
cost = 2, kernel = "sigmoid")
proc.time() -ti
svm.pred = predict(svm.model, test.clinvar_scaled[,-1])
t = table(pred = svm.pred, true = test.clinvar_scaled[,1])
alpha = t[2,1]/sum(t[,1])
beta = t[1,2]/sum(t[,2])
L_w = 1/4*alpha + 3/4*beta



################ LDA ##################

 ti = proc.time()
 clinvar.lda=lda(CLASS~CHROM.11 + CHROM.17 +
CHROM.5 + CLNVC.single_nucleotide_variant +
 Consequence.frameshift_variant + IMPACT.HIGH +
 PolyPhen.probably_damaging +
CADD_RAW + AF, data = train.clinvar_scaled)
```

```
proc.time() - ti
clinvar.pred = predict(clinvar.lda,
 test.clinvar_scaled[,-1])
table.lda = table(pred = clinvar.pred$class,
true = test.clinvar_scaled[,1])
alpha = table.lda[2,1]/sum(table.lda[,1])
beta = table.lda[1,2]/sum(table.lda[,2])
L_w = 1/4*alpha + 3/4*beta


#################### QDA ################

ti = proc.time()
clinvar.qda=qda(CLASS~CHROM.11 + CHROM.17 +
CHROM.5 + CLNVC.single_nucleotide_variant +
Consequence.frameshift_variant + IMPACT.HIGH +
PolyPhen.probably_damaging +
CADD_RAW + AF, data = train.clinvar_scaled)
proc.time() - ti
clinvar.pred = predict(clinvar.qda,
test.clinvar_scaled[,-1])
table.qda = table(pred = clinvar.pred$class,
true = test.clinvar_scaled[,1])
alpha = table.qda[2,1]/sum(table.qda[,1])
beta = table.qda[1,2]/sum(table.qda[,2])
L_w = 1/4*alpha + 3/4*beta


#################### KNN
k = 100
```

```
L = rep(NA,k)
L.min = 2
k.opt=NA
for (i in 1:k)
{
  class.knn = knn(train.clinvar_scaled[,-1],
  test.clinvar_scaled[,-1],
  train.clinvar_scaled[,1],k = i)
  T.knn = table(pred = class.knn,
  true = test.clinvar_scaled[,1])
  alpha = T.knn[2,1]/sum(T.knn[,1])
  beta = T.knn[1,2]/sum(T.knn[,2])
  L[i] = alpha + beta
  if (L.min > L[i]){
    k.opt = i
    T.knn.opt = T.knn
    L.min = L[i]
    alpha.min = alpha
    beta.min = beta}
}

plot(1:k,L,type='l')
abline(h=L.min,v=k.opt,col="red")

et = proc.time()
class.knn = knn(train.clinvar_scaled[,-1],
test.clinvar_scaled[,-1],
train.clinvar_scaled[,1],k = k.opt)
proc.time() - et
T.knn = table(pred = class.knn,
```

```
284  true = test.clinvar_scaled[,1])
285  alpha = T.knn[2,1]/sum(T.knn[,1])
286  beta = T.knn[1,2]/sum(T.knn[,2])
287  L_w = 1/4*alpha + 3/4*beta
288
289
290  ########## Maximal tree ###########
291  sctrl = rpart.control(cp=0)
292  et = proc.time()
293  tree.max = rpart(as.factor(CLASS) ~ CHROM.11 +
294  CHROM.17 + CHROM.5 + CLNVC.single_nucleotide_variant +
295  Consequence.frameshift_variant + IMPACT.HIGH +
296  PolyPhen.probably_damaging +
297  CADD_RAW + AF, data = train.clinvar_scaled,
298  method="class",control=sctrl)
299  proc.time() - et
300  print(tree.max)
301  plot(tree.max)
302  text(tree.max)
303
304  summary(tree.max)
305  # Examine the complexity plot
306  printcp(tree.max)
307  plotcp(tree.max)
308  # Compute the accuracy of the max tree
309  test = predict(tree.max,
310  test.clinvar_scaled[,-1], type = "class")
311  accuracy = mean(test == test.clinvar_scaled[,1])
312
313  P = predict(tree.max,
```

```
314  test.clinvar_scaled[,-1], type = "class")
315  T.maximal = table(pred = P,
316  true = test.clinvar_scaled[,1])
317  alpha=T.maximal[2,1]/sum(T.maximal[,1])
318  beta=T.maximal[1,2]/sum(T.maximal[,2])
319  L_w = 1/4*alpha + 3/4*beta
320
321
322  ############ Pruned trees ###################
323  tree.pruned=prune(tree.max,cp=0.0030518820)
324  printcp(tree.pruned)
325  plot(tree.pruned, main = "Pruned regression tree")
326  text(tree.pruned)
327
328  P.pruned = predict(tree.pruned, test.clinvar_scaled[,-1], type = "class"
329  T.pruned = table(pred = P.pruned, true = test.clinvar_scaled[,1])
330  alpha=T.pruned[2,1]/sum(T.pruned[,1])
331  beta=T.pruned[1,2]/sum(T.pruned[,2])
332  L_w = 1/4*alpha + 3/4*beta
333
334  ############ LOGISTIC REGRESSION ############
335
336  et = proc.time()
337  logis.model = glm(as.factor(CLASS) ~ CHROM.11 + CHROM.17 +
338  CHROM.5 + CLNVC.single_nucleotide_variant +
339  Consequence.frameshift_variant + IMPACT.HIGH +
340  PolyPhen.probably_damaging +
341  CADD_RAW + AF, data = train.clinvar_scaled,
342  family=binomial(link = "logit"))
343  proc.time() - et
```

```
344
345 TPR = rep(NA,50) # (sensitivity)
346 TNR = rep(NA,50) # (specificity)
347 R=0
348
349 for (t in 1:50)
350 {
351   tau=t/100
352   logis.class = as.numeric(fitted(logis.model)>=tau)
353   T.log = table(predict = logis.class,
354   true = train.clinvar_scaled[,1])
355   TPR[t]= T.log[1,1]/sum(T.log[,1])
356   TNR[t]= T.log[2,2]/sum(T.log[,2])
357   if (R < TNR[t] + TPR[t]) {
358     tau_opt = tau
359     R = TNR[t] + TPR[t]
360     TPR_opt = TPR[t]
361     TNR_opt = TNR[t]
362     T_opt = T.log}
363 }
364
365 plot(1-TNR,TPR,type='l') # courbe ROC
366 points(1-TNR_opt,TPR_opt,col="green")
367 abline(v=1-TNR_opt,h=TPR_opt,col="green")
368 legend("bottomright","Tau optimal=0.31",
369 col="green",lty=1,lwd="1",cex = 0.7)
370 tau_opt # le tau optimal = 0.31
371 T_opt # Table de confusion optimal
372
373 et = proc.time()
```

```
logis.model = glm(as.factor(CLASS) ~ CHROM.11 + CHROM.17 +
CHROM.5 + CLNVC.single_nucleotide_variant +
Consequence.frameshift_variant + IMPACT.HIGH +
PolyPhen.probably_damaging +
CADD_RAW + AF, data = train.clinvar_scaled,
family=binomial(link = "logit"))
proc.time() - et
variant.class = as.numeric(predict(logis.model,
test.clinvar_scaled[,-1],type="response")>=tau_opt)
glm.T = table(predict = variant.class,
true = test.clinvar_scaled[,1])
TPR_opt=T.log[1,1]/sum(T.log[,1])
TNR_opt=T.log[2,2]/sum(T.log[,2])
## No weights##
alpha = 1 - TPR_opt
beta = 1 - TNR_opt
L = alpha + beta
## With weights ##
L_w = 1/4*alpha + 3/4*beta


## ROCK CLUSTERING
library(cba)
x=model.matrix(~.,data = train.clinvar_scaled)
clus = rockCluster(x, 2, fun = "dist",
funArgs = list(method="binary"), debug = FALSE)


## LEAVE ONE OUT CROSS VALIDATION PREDICTION
glmnet.hat=rep(NA,4000)
```

```
404  for(i in 1:nrow(clinvar)){
405   x=model.matrix(~.,data = clinvar[-i,-1])
406   y=clinvar[-i,1]
407  model_lasso = cv.glmnet(x,y , family="binomial"
408   , type.measure = "auc",nfolds = nrow(clinvar) ,alpha=1)
409  glmnet.hat[i]= predict(model_lasso,clinvar[i,1],type="class")
410
411 }
```