ALLEMAND Fabien
SIX Valentin

Apprentissage, Classification Automatique, Data Mining

TP - Algorithmes d'apprentissage ensembliste

1 Introduction

À une époque où l'on parle souvent de *machine learning*, il est important de savoir qu'il existe plusieurs types d'apprentissages automatiques : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé et l'apprentissage par renforcement. Pouvant chacun utiliser différentes méthodes, la plus connue de nom étant l'apprentissage profond.

Cette étude porte sur les méthodes d'apprentissage supervisé basées sur l'apprentissage ensembliste. L'apprentissage ensembliste repose sur le principe de la sagesse des foules, c'est-à-dire que (sous certaines conditions) la décision d'un groupe est meilleure que la décision d'un seul. On utilisera donc plusieurs classifieurs afin de décider de la classe d'un individu.

Les classifieurs de bases utilisés pour l'apprentissage ensembliste peuvent en théorie être quelconques. Nous utiliserons pour ce travail des arbres de décisions ainsi nous pourrons comparer les résultats de classification avec un ou plusieurs classifieurs entraînés de différentes façons (*random forests*, *bagging* et *boosting*) sur différents jeux de données (*Iris flower data set*, *diabetes.csv*, *UCI ML handwritten digits dataset* et *UCI ML Breast Cancer Wisconsin (Diagnostic) dataset*).

L'objectif de notre travail est de comprendre et de mettre en pratique quelques algorithmes d'apprentissage supervisé basés sur l'apprentissage ensembliste. Dans un premier temps, nous détaillerons l'intérêt de l'apprentissage ensembliste et présenterons les algorithmes et les bases de données utilisés par la suite. Nous procéderons ensuite à une comparaison des arbres de décision et des forêts aléatoires sur différentes données. Enfin, nous analyserons de façon détaillée les méthodes de bagging et de boosting en portant une attention particulière aux avantages et aux limites de chaque algorithme ou méthode utilisée.

2 Apprentissage Ensembliste

Dans cette section, nous détaillons le fonctionnement de l'apprentissage ensembliste et des algorithmes qui reposent sur cette méthode.

2.1 Principe de Fonctionnement

Dans différents contextes, on préfère utiliser plusieurs entités de décision pour assurer le bon fonctionnement d'un système. On peut distinguer deux cas :

- La redondance : plusieurs individus effectuent le même travail dans l'éventualité où un autre individu devient défaillant afin d'assurer la sûreté et la fiabilité du système.
- Le partage des tâches : plusieurs individus coopèrent pour assurer le bon fonctionnement du système que ce soit en capacité, en rapidité ou bien en précision.

Pour l'apprentissage ensembliste, le principe est de partager le travail afin d'obtenir une meilleure prise de décision (une meilleure précision) lors de la construction des groupes. L'intuition donnée par le concept de "la sagesse des foules" montre que les décisions prises par plusieurs classifieurs permettent d'obtenir une meilleure classification. En effet, il est possible de faire voter les classifieurs quant à la classification d'un individu. Il faut cependant prendre garde à respecter certaines conditions pour que ce principe soit respecté en pratique :

- Diversité : les classifieurs utilisés doivent être suffisamment divers pour que la décision de groupe soit intéressante. La décision issue d'un vote de classifieurs peu compétents a peu d'intérêt.
- Compétence : les classifieurs doivent être suffisamment compétents (taux de bonne classification supérieur à 50%). La décision d'un ensemble de classifieurs regroupant tous les données de "la mauvaise façon" a peu d'intérêt.
- Nombre : le nombre de classifieurs doit être suffisamment important pour que la décision prise par l'ensemble soit meilleure que la décision d'un seul classifieur (loi des grands nombres). Il faut être sûr que la décision issue du vote des classifieurs ne soit pas due au hasard (deux mauvais classifieurs contre un seul bon...).

2.2 Algorithmes

L'apprentissage ensembliste repose sur les décisions prises par plusieurs classifieurs. Nous allons présenter dans un premier temps le classifieur de base utilisé dans cette étude (arbre de décision) puis nous étudierons les différentes stratégies (algorithmes) qui existent pour créer une population de classifieurs pertinente.

2.2.1 Arbres de Décision

Dans cette étude nous utilisons comme classifieurs de base des arbres de décision. Un arbre de décision est un ensemble de règles de classification (des tests sur les attributs des individus) arrangées selon la forme d'un arbre. Une fois l'arbre construit, il suffit de suivre les branches de l'arbre en se laissant guider par les tests afin de classer un individu dans une feuille.

Ce classifieur a l'avantage d'être facilement interprétable et rapide lors de l'inférence. Il permet aussi de traiter directement des données qualitatives et/ou quantitatives (non normalisées[1]) pour des problèmes de classification binaire ou multiclasse.

Cependant, il faut aussi remarquer qu'il n'est pas évident de construire un arbre de décision. L'objectif est d'atteindre le plus rapidement possible (en une profondeur minimale) des feuilles pures (contenant uniquement des individus de la même classe): il faut pour cela choisir un critère de séparation (entropie ou indice de Gini).

[1] Normalisation inutile car les attributs sont traités indépendamment.

Il faut aussi prendre garde à ne pas effectuer de sur-apprentissage. Deux choix s'offrent à nous : élaguer l'arbre (supprimer les branches qui apportent le moins d'information) après sur-apprentissage ou bien définir à l'avance différents hyper paramètres (profondeur maximale de l'arbre, nombre minimal d'individus dans une feuille...) afin de se protéger du sur-apprentissage. Cela rend les arbres de décisions lents lors de l'apprentissage et inefficaces lorsqu'il y a trop d'attributs. Ils sont aussi instables lors de l'ajout de nouvelles données ou lorsque les données changent : l'arbre classe en fonction de ce qu'il a appris, si par la suite les données changent (iris plus grands ou plus petits à cause du climat ou autre) la classification sera moins bonne. Il faut aussi être vigilant lors de l'utilisation de nouvelles données car la classification peut perdre tout son sens : par exemple les unités jouent un rôle crucial, si on entraîne un arbre avec des longueurs de pétales exprimées en centimètres il ne faut pas présenter de nouvelles données avec ces mêmes longueurs exprimées en millimètres. L'arbre n'ayant pas conscience des attributs qu'il manipule, il n'effectuera pas la classification d'un iris avec des pétales de 30 millimètres mais un iris avec des pétales de 30 centimètres.

2.2.2 Bagging

Bagging Théorique Le bagging est une stratégie d'apprentissage ensembliste basée sur l'aléatoire. Le mot bagging provient de la combinaison des mots *bootstrap* et *aggregating* qui décrivent les deux étapes majeures de la stratégie :

- Création d'échantillons issus de la population totale par tirage aléatoire (généralement avec remise).
- Agrégation des résultats des différents modèles entraînés sur les échantillons (par exemple : moyenne pour une tâche de régression ou vote lorsqu'il s'agit de classification).

Le but est donc d'entraîner en **parallèle** différents modèles sur des sous-ensembles des données puis de combiner leurs résultats pour obtenir un meilleur résultat final.

En théorie les classifieurs utilisés peuvent être complètement différents : on pourrait imaginer combiner les résultats de plusieurs arbres de décisions avec ceux de plusieurs réseaux de neurones. En pratique, notamment dans le cadre de ce travail, on se limite souvent à des modèles similaires comme les arbres de décisions. De la même façon, on pourrait utiliser des arbres de décisions avec différents hyper-paramètres mais la bibliothèque Python scikit-learn crée par défaut des classifieurs de base avec les mêmes hyper-paramètres (seul les données d'entraînement influencent le classifieur).

Forêts Aléatoires La partie précédente décrit le fonctionnement théorique du bagging. En pratique, rien n'empêche de créer des variantes de cette méthode pour (espérer) obtenir de meilleurs résultats pour un problème donné. Ainsi, la forêt aléatoire est une forme particulière de bagging pour laquelle, lors de la création des sous-ensembles de données, il y a un tirage aléatoire des individus mais aussi des attributs utilisés. Intuitivement, limiter les attributs accessibles par chaque classifieur amplifie la diversité de ces classifieurs (décorrélation ou indépendance des classifieurs) limitant le sur apprentissage au détriment d'un temps d'apprentissage légèrement plus long.

Les forêts aléatoires sont généralement composées d'arbres de décisions mais il n'est pas impossible d'imaginer une forêt aléatoire de classifieurs d'un ou plusieurs autres types.

2.2.3 Boosting

Boosting Théorique Le boosting est une stratégie d'apprentissage ensembliste dite adaptative. Contrairement au bagging qui se base sur l'aléatoire et des classifieurs indépendants, le boosting vise à entraîner en **série** plusieurs classifieurs de base. Chaque classifieur recevant un *feedback* du précédent afin qu'il puisse corriger les erreurs de ce dernier.

Comme précédemment, les classifieurs de bases peuvent en théorie être quelconques. En pratique on préfèrera des *weak-learners* car on cherche des classifieurs avec une faible variance sans se soucier du biais qui sera réduit lors de l'aggrégation des résultats (contrairement au bagging où les classifieurs on un biais faible mais un risque de forte variance réduit par l'aggrégation). Une fois de plus dans cette étude nous utiliserons des arbres de décisions de faible profondeur (arbres de décision avec faible capacité d'apprentissage).

AdaBoost Adaboost est un algorithme qui repose sur le principe du boosting. Il utilise généralement des *stumps* (arbres de décision de profondeur 1) comme classifieur de base. Chacun de ces weak-learners indique au suivant ce qu'il n'a pas pu bien classer pour qu'il s'en occupe : cela se traduit par une augmentation des poids des données mal classées.

D'autres algorithmes utilisent une descente de gradient pour effectuer le boosting (gradient boosting).

3 Bases de Données

L'objectif de cette section est de décrire au préalable les différentes bases de données utilisées dans la suite de ce travail.

3.1 Iris flower data set

Dans un premier temps nous souhaitons prendre en main et comparer les résultats des arbres de décision et des forêts aléatoires sur une base de données connue (description et qualité des données), simple (peu d'attributs et d'individus) et facile à interpréter (données concrètes et facilement visualisables). Nous utilisons pour cela la base de données *Iris flower data set*[2]. Cette base de données recense 150 individus de fleurs de 3 espèces d'iris différentes représentées par 4 grandeurs physiques exprimées en centimètre (longueur

[2] https://en.wikipedia.org/wiki/Iris_flower_data_set

et largeur des pétales et des sépales). Cette petite base de données est équilibrée (50 individus par classe) et présente des données de bonne qualité (pas de valeurs manquantes) pour effectuer une tâche de classification multiclasse sur des attributs quantitatifs.

Cette base de données étant simple, l'étape de visualisation donne déjà des intuitions sur les résultats attendus. On peut visualiser les données en deux dimensions en sélectionnant deux à deux les attributs et en colorant les points selon leur classe[3](Figure 1). On observe déjà que certains attributs pourraient être utilisés pour premier test d'un arbre de décision afin d'obtenir une première séparation homogène (par exemple : la longueur des pétales permet d'isoler les iris setosa).

Remarque : Il faut penser à mélanger les données avant l'apprentissage car, par défaut, les fleurs sont listées par classe. Dans la suite, on prendra soin de mélanger ces données avant de les séparer en deux groupes : les données d'entraînement (70%) et les données de test (30%).

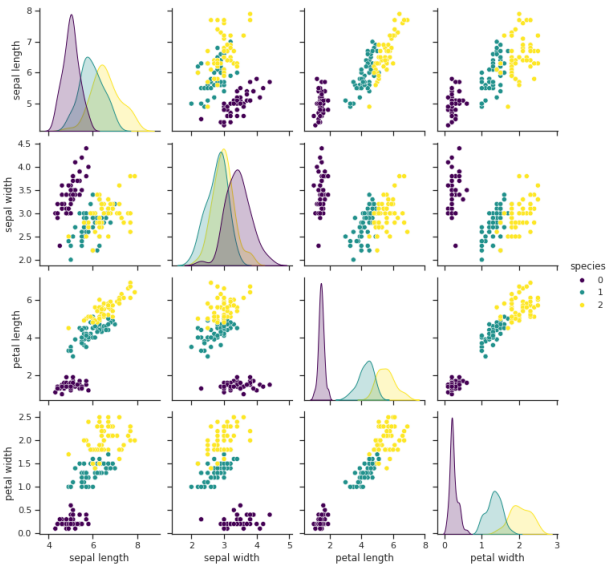


Figure 1: Pair plot de la base de données *Iris flower data set*

3.2 Diabetes

La base de données contenue dans le fichier `diabetes.csv` contient 8 attributs quantitatifs correspondant à des caractéristiques (âge...) ou mesures physiologiques (pression sanguine, épaisseur de la peau...) de 768 individus ainsi qu'une étiquette décrivant si un individu souffre de diabète. On ne constate aucune valeur manquante mais cette base de données n'est pas tout à fait équilibrée : on compte 500 individus sains contre 268 patients malades ce qui pourrait impacter les résultats de la classification binaire.

[3]Lorsqu'on effectue cela pour tous les couples possibles d'attributs (possible ici car il y en a peu) cela s'appelle un pair plot. L'implémentation de la bibliothèque Python seaborn présente en double chaque graphe : il y a une symétrie par rapport à la diagonale du tableau de graphes sur laquelle on retrouve la distribution de chaque attribut au sein de chaque classe.

Le nombre d'attributs est encore assez limité pour faire un pair plot relativement facile à analyser cependant on ne remarque pas de couple d'attributs permettant d'isoler les deux classes.

3.3 UCI ML Handwritten Digits Dataset

La troisième base donnée que nous utilisons est nommée *UCI ML handwritten digits dataset*[4]. Cette base de données libre d'accès présente de nombreuses données de bonne qualité et utilisables sans pré-traitement.

Cette base de données contient 1797 images en niveaux de gris, représentant chacune un chiffre manuscrit (0 jusqu'à 9). La résolution de chaque image est de 8 x 8 pixels, ce qui fait 64 pixels par image. On remarque sur la Figure 2 que les images sont beaucoup moins lisibles pour un humain mais sont beaucoup plus faciles à manipuler pour un ordinateur (moins de valeurs) que celle de la base de données *MNIST_784 Handwritten Digits*.

Ces chiffres manuscrits ont été écrits à la main par différentes personnes qui ont sensiblement une écriture différente ; ainsi les images sont toutes uniques et on observe une certaine hétérogénéité dans la forme et la taille des chiffres manuscrits. Il est important de noter que pour chaque image de la base données, le chiffre présent sur l'image est entièrement visible, bien centré dans l'image et bien orienté : il n'y a donc pas de travail de prétraitement des données à effectuer pour obtenir des échantillons de qualité.

À chaque image est également associé un *label*, c'est-à-dire une étiquette qui correspond au chiffre représenté sur l'image.

Une dernière remarque concernant la base de données : on peut vérifier à l'aide des étiquettes des images, que les données sont bien équilibrées (autrement dit, tous les chiffres sont représentés dans un nombre comparable d'images, environ 180). Cet aspect d'équilibre dans la base de données est très important car cela assure qu'un chiffre ne sera pas sur-représenté ou au contraire sous-représenté au sein de la base de données. Avoir un jeu de données équilibré permet d'obtenir de meilleurs résultats de manière générale car l'apprentissage se fera uniformément sur toutes les classes.

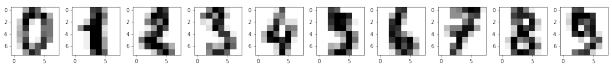


Figure 2: Visualisation d'un individu de chaque classe de la base de données *UCI ML handwritten digits dataset*

[4]`https://archive.ics.uci.edu/dataset/80/optical+recognition+of+handwritten+digits`

3.4 UCI ML Breast Cancer Winsconsin (Diagnostic) Dataset

La base de données *UCI ML breast cancer Wisconsin (diagnostic) dataset*[5] est une base de données décrivant 569 individus avec 30 attributs (un identifiant et un diagnostic qui sont des variables catégorielles et 28 attributs continus correspondant à des mesures physiologiques). On remarque que cette base de donnée n'est pas bien équilibrée : on observe 212 tumeurs malignes contre 357 tumeurs bénignes.

Cette base de données présente trop d'attributs pour faire un pair plot utile mais on peut tout de même visualiser les données en sélectionnant les attributs deux par deux comme par exemple sur la Figure 3.

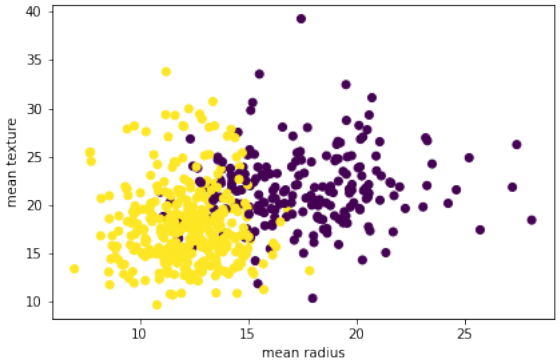


Figure 3: Visualisation des attributs *mean radius* et *mean texture* pour les individus de la base de données *UCI ML handwritten digits dataset*.

4 Comparaison Arbres de Décision et Forêts Aléatoires

Cette section a pour objectif de comparer les arbres de décisions et les forêts aléatoires sur différents jeux de données afin d'apprécier les capacités, les avantages et les inconvénients de ces deux classifieurs.

4.1 Arbres de Décision

Nous procédons tout d'abord à une analyse des résultats obtenus avec les arbres de décisions sur deux bases de données.

4.1.1 Iris flower data set

On souhaite effectuer une classification multiclasse de données quantitatives en utilisant la base de données Iris de Fisher.

On commence par construire un arbre sans spécifier les hyper paramètres (Figure 4) ce qui implique qu'il n'y a pas de limites sur la profondeur ni sur la taille des groupes formés. On observe alors un arbre de décision relativement profond (profondeur de 4). On peut aisément suivre cet arbre de décision pour classer un individu. Prenons par exemple un iris ayant une longueur de sépale de 5,1 cm, une largeur de sépale 3,5 cm et des pétales de 1,4 cm de long et 0,2 cm de large. **x[3]** correspond à la largeur des pétales, l'individu étudié a cet attribut inférieur à 0,75 donc le test est valide, on descend dans ce cas dans la feuille de gauche qui est une feuille pure correspondant aux iris setosa. L'inférence s'arrête donc là et cela correspond bien à la vérité terrain.

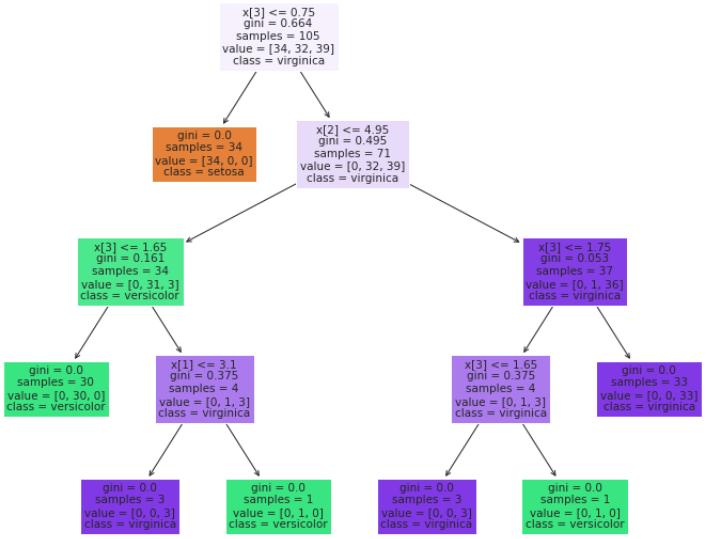


Figure 4: Arbre de décision pour classer les individus de la base de données *Iris flower data set*

La bonne classification de cet individu est attendue car cet arbre de décision présente un pourcentage de bonne classification de 100% sur les données d'entraînement et de 98% sur les données de test. La plupart des individus de cette base de données sont bien classés par cet arbre. D'autres indicateurs mettent en valeur les bonnes performances de ce classifieur pour cette tâche : notamment le taux de bonne classification par classe (pondérée par le nombre d'individus dans chaque classe) est de 98% et la précision (pourcentage de bonne classification parmi les individus classés dans une classe) et le rappel (le pourcentage de bonne classification pour une classe réelle) par classe sont tous supérieurs à 90% entraînant un score f1 supérieur à 0.95 pour les trois classes d'iris. On peut également visualiser le résultat de la classification du jeu de test sur la Figure 5 avec un tableau (appelé matrice de confusion). On voit qu'un seul individu des données de test n'est pas classé correctement.

[5] `https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic`

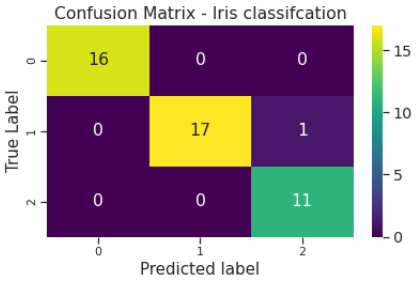


Figure 5: Matrice de confusion de la classification les individus tests de la base de données *Iris flower data set* avec un arbre de décision

Bien que les résultats de ce classifieur soient très bons, on peut se demander si la profondeur de l'arbre est justifiée. Certes toutes les feuilles sont pures mais on observe plusieurs nœuds presque purs (un seul individu d'une classe) qui donnent naissance à deux nouvelles feuilles pures. Cela augmente le temps d'entraînement et d'inférence et réduit l'interprétabilité de l'arbre mais surtout cela peut mener à du sur apprentissage des données d'entraînement. Le classifieur s'adapte trop aux données qu'il utilise lors de la phase d'entraînement et perd en capacité de généralisation. Il est trop spécifique pour donner de bons résultats sur d'autres données.

Une méthode pour pallier ce problème serait de demander à un expert quelles branches sont inutiles ou moins importantes et procéder à un élagage de l'arbre. Par exemple, dans ce cas, un expert pourrait considérer que pour cette application ce n'est pas grave de confondre quelques iris versicolor avec des iris virginica et supprimer toutes les feuilles du niveau 4 de l'arbre.

Il est aussi possible de modifier des hyper paramètres ajoutant des contraintes lors de la création de l'arbre. En créant un nouvel arbre avec une profondeur limitée à 2 (Figure 6), on observe un classifieur qui est beaucoup plus facile à interpréter (moins de tests pour une inférence) donnant des résultats qui peuvent encore être acceptables pour cette tâche : 96% et 91% de bonne classification respectivement sur les données d'entraînement et de test. La matrice de confusion obtenue sur les données de tests (Figure 7) montre aussi des résultats tout à fait acceptables : seulement 4 individus sur 45 sont mal classés.

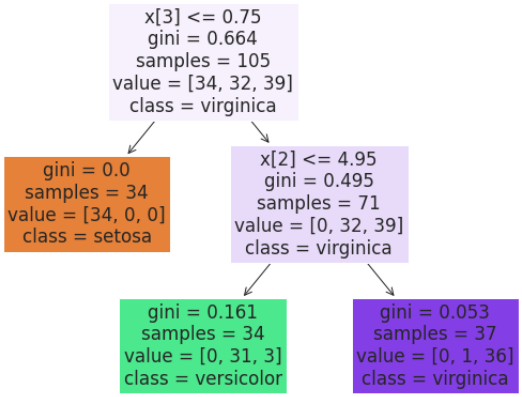


Figure 6: Arbre de décision avec profondeur maximale limitée pour classer les individus de la base de données *Iris flower data set*

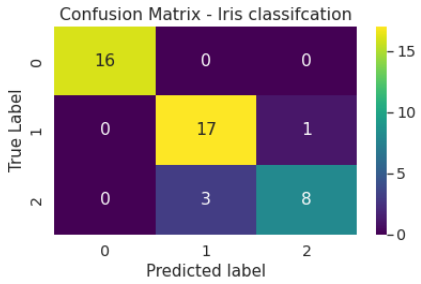


Figure 7: Matrice de confusion de la classification les individus tests de la base de données *Iris flower data set* avec un arbre de décision ayant une profondeur maximale limitée

4.1.2 Diabetes

La base de données diabètes permet de tester les performances des arbres de décision pour la classification sur des données plus complexes (plus d'attributs).

De la même façon que pour les iris de Fisher, on crée dans un premier temps un arbre sans contraintes. On obtient alors des résultats de classification qui semblent bons au premier abord : respectivement 100% et 74% de bonne classification sur les données d'entraînement et de test. Comme on pouvait s'y attendre, la complexité des données rend la tâche plus difficile et les résultats sont moins bons que pour la classification des iris. Cependant, on s'aperçoit rapidement que le classifieur est compliqué à interpréter (Figure 8) et l'étude de la matrice de confusion, présentée en Figure 9, montre une précision (0.82 et 0.61 respectivement pour les classes non-diabétique et diabétique) et un rappel (0.82 et 0.62 respectivement pour les classes non-diabétique et diabétique) potentiellement trop faibles pour une application dans le domaine médical qui présente de fort risques (traitement d'un patient sain ou non traitement d'un patient malade).

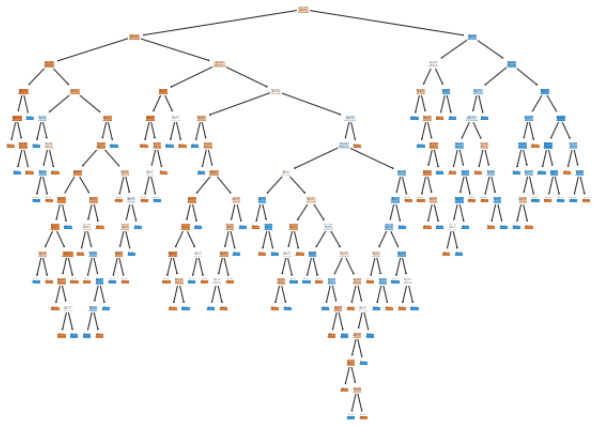


Figure 8: Arbre de décision pour classer les individus de la base de données *Diabetes*

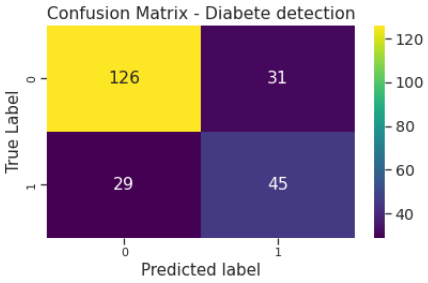


Figure 9: Matrice de confusion de la classification les individus tests de la base de données *Diabetes* avec un arbre de décision

Il est aussi possible de limiter la profondeur de l'arbre comme expliqué précédemment mais là encore les performances sont dégradées au profit d'une meilleure interprétabilité de l'arbre de décision. On comprend facilement larbre de la Figure 10 mais les performances de l'arbres de décision sur la tâche de classifications sont désormais insuffisantes pour une applicatoin médicale sérieuse.

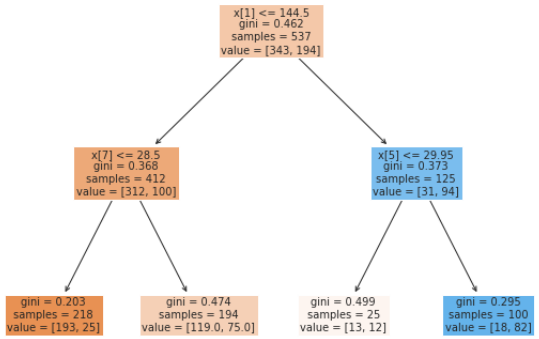


Figure 10: Arbre de décision avec profondeur maximale limitée pour classer les individus de la base de données *Diabetes*

4.2 Forêts Aléatoires

En suivant la même méthodologie que pour les arbres de décisions, nous analysons les résultats des forêts aléatoires sur les deux premières bases de données.

4.2.1 Iris flower data set

Appliquons l'algorithme de forêt aléatoire pour effectuer une classification multi-classes des données. On crée un classifieur en utilisant les hyper paramètres par défaut (n_estimators = 100 et min_samples_split = 2). Après les phases d'entraînement et de prédiction, on obtient les résultats suivants pour la forêt aléatoire :

- 100% de bonne classification sur les données d'entraînement.
- 97,78% de bonne classification sur les données de test.

La Figure 11 montre la matrice de confusion qui apporte des précisions sur les erreurs de classification lors de la phase de prédiction test. La forêt aléatoire ne fait qu'une seule erreur sur les 45 échantillons. Les résultats sont satisfaisants.

Figure 11: Matrice de confusion de la classification les individus tests de la base de données *Diabetes* avec une forêt aléatoire

L'algorithme de forêt aléatoire permet d'identifier les attributs les plus importants pour la classification, c'est

à dire, les *features* qui permettent de mieux séparer les individus des différentes classes. On observe sur la Figure 12 que dans ce jeu de données certains attributs ont une importance bien plus élevée que les autres (petal_length et petal_width). Ces deux attributs combinés ont une importance combinée de presque 0.9 (90% des informations utiles à la classification sont contenus dans ces deux attributs). On peut supposer qu'une classification des individus basée sur ces deux seuls attributs donnerait des résultats très convenables.

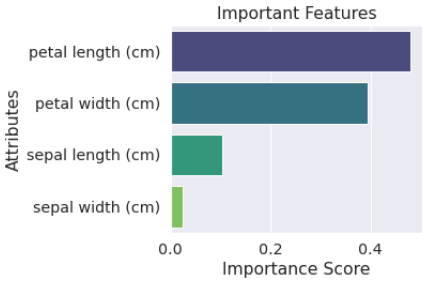


Figure 12: Importance des attributs des données de *Iris flower data set* pour effectuer une classification multi-classes

Pour illustrer l'importance des différents attributs, on peut considérer les attributs séparément et réaliser une classification des individus basée sur chaque attribut isolé (dans ce cas, étant donnée qu'il n'y a qu'un seul attribut, cela revient à créer un arbre de décision) :

- En ne considérant que la longueur des pétales : 91.11% de bonne classification sur les données de test.
- En ne considérant que la largeur des pétales : 97.78% de bonne classification sur les données de test.
- En ne considérant que la longueur des sépales : 60% de bonne classification sur les données de test.
- En ne considérant que la largeur des sépales : 53.33% de bonne classification sur les données de test.

Les résultats obtenus sont cohérents avec l'importance des attributs. La matrice de confusion dans le pire des cas (Figure 13) montre beaucoup plus d'erreurs que ce que nous avons pu voir jusqu'à présent. L'attribut sepal_width ne permet pas de bien classifier les fleurs. En revanche, utiliser les deux attributs avec les meilleurs scores d'importance donne les mêmes résultats que ceux obtenus avec l'ensemble des données (97,78% d'accuracy) ce qui montre bien que les deux attributs restants sont peu utilisés pour la classification (cela se voit aussi sur le pair plot en Figure 1, aucun de ces deux attributs ne permettent des distinguer des classes).

Utiliser deux attributs ayant un score d'importance élevés n'améliore pas toujours les résultats. Si ces attributs sont corrélés, il y a juste une redondance dans les données et il vaudrait mieux remplacer l'un des deux attributs par un troisième non corrélé avec les premiers.

Il existe différentes méthodes pour sélectionner les variables les plus pertinentes. Il est possible d'ajouter less variables une à une et de sélectionner seulement celles qui apportent une augmentation significative des performances. Une autre méthode consiste à ajouter à plusieurs reprises une variable fictive correspondant à du bruit et à délaisser les attributs qui sont généralement moins importants que la fausse variable. Cette méthode est dite "méthode de la variable sombre". Dans les deux cas, rien ne certifie que les variables conservées sont réellement pertinentes et il faut parfois demander confirmation à un expert si possible.

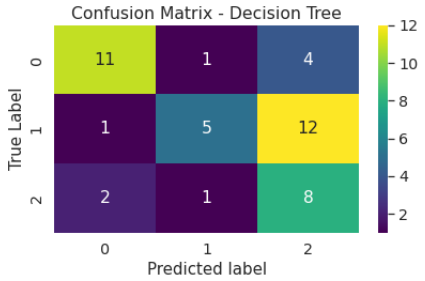


Figure 13: Matrice de confusion de la classification les individus tests de la base de données *Iris flower data set* réduite à l'attribut sepal_width avec une forêt aléatoire

4.2.2 Diabetes

Nous utilisons maintenant les forêts aléatoires sur la base de données du diabète qui est plus complexe et nous augmentons le nombre d'arbres de décision à 300. Nous obtenons 78,78% de bonne classification sur les données de test. Ce qui est nettement moins élevé que les résultats obtenus sur la base de données précédente et montre que les forêts aléatoires sont sensibles à la complexité des données et en particulier au nombre d'attributs.

4.3 Comparaison des Deux Classifieurs

Nous procédons dans cette partie à une comparaison des deux algorithmes.

4.3.1 Comparaison sur les Résultats Précédents

Jusqu'à présent nous avons appliqué les deux algorithmes sur deux bases de données (*Iris flower data set* et *diabetes*). Cela permet de tester les algorithmes dans des conditions différentes et déceler des particularités.

En particulier, nous avons vu que les deux méthodes donnent des résultats similaires : 91% de bonne classification en phase de test pour un arbre de décision raisonnable et 98% pour un arbre de décision plus détaillé (avec un risque de sur-apprentissage) contre 98% de bonne classification de test avec une forêt aléatoire. Dans le cadre de cette application, l'augmentation des performance n'est peut être pas assez significative pour justifier la légère augmentation du temps de calcul et la perte d'interprétabilité des résultats. Sur des données simples comme la base de données des iris, les arbres de décision peuvent être suffisants.

En revanche, pour un travail avec des données plus complexes comme celles sur le diabète, les résultats montrent

que l'apprentissage ensembliste est bénéfique. Dans le meilleur des cas, un simple arbre de décision fournit un pourcentage de classification correcte en test de 74% alors qu'une forêt aléatoire peut presque atteindre 79%.

4.3.2 Comparaison sur un Troisième Jeu de Données

Nous poursuivons la comparaison des deux méthodes avec un dernier jeu de données que nous n'avons pas utilisé jusqu'ici. La base de données *UCI ML handwritten digits dataset* est une base de données permettant d'effectuer une classification multiclasses sur des données encore plus complexes que les précédentes. Les individus (images de chiffres manuscrits) sont interprétés par les arbres de décision comme 64 attributs distincts.

Pour l'arbre de décision, nous avons réalisé plusieurs tirages pour la constitution des ensembles d'apprentissage et d'entraînement, de manière à avoir des résultats sûrs. En moyenne, le pourcentage de bonne classification des chiffres manuscrits est d'environ 82.5%. Pour comparer l'algorithme d'arbre de décision avec l'algorithme de forêt aléatoire, nous avons choisi l'hyperparamètre `n_estimators = 300` (qui correspond au nombre d'arbres pour l'algorithme de forêt aléatoire) et nous avons également réalisé plusieurs tirages pour la constitution des ensembles d'apprentissage et d'entraînement : cette fois-ci, on obtient une moyenne de 95.57%. Dans ce cas là, on observe bien que les performances de classification sont meilleures.

Il est important de comprendre que les résultats obtenus avec l'algorithme de forêt aléatoire dépendent de l'hyperparamètre *n_estimators*. Comme on peut le voir sur la Figure 14, en moyenne, les résultats augmentent avec le nombre d'estimateurs de base[6].

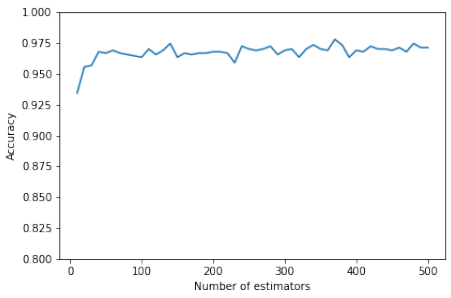


Figure 14

Un indice de performance supplémentaire que nous pouvons analyser est l'écart type de l'exactitude de classification pour les deux algorithmes. Le pourcentage de bonne classification varie environ deux fois plus pour l'algorithme d'arbre de décision (écart type égal à 0.009) que pour l'algorithme de forêt aléatoire (écart type égal à 0.005). On peut donc conclure que l'utilisation d'une forêt aléatoire est plus adaptée pour la classification de chiffres manuscrits car cela permet d'avoir des résultats plus fiables quels que soient les individus utilisés lors de l'apprentissage (en supposant que les classes restent équilibrées dans les données d'entraînement).

Remarque : Nous avons aussi essayé d'utiliser l'algorithme AdaBoost pour réaliser la classification multi-classes pour des chiffres manuscrits. Cet algorithme, utilisé avec 500 arbres de décisions de profondeur maximale de 1, donne un pourcentage de classification correcte de 75.64%. AdaBoost donne donc des résultats bien moins bons que les deux algorithmes précédents sur cette tâche spécifique. Nous verrons cependant par la suite que cet algorithme peut être bien meilleur que les autres sur une autre base de données. Ce phénomène montre à nouveau qu'il n'existe pas un meilleur algorithme pour touts les tâches. Il faut connaitre le fonctionnement des algorithmes qu'il est possible d'utiliser pour une tâche donnée pour sélectionner ceux qui sont suceptibles de donner les meilleurs résultats puis de les comparer et de sélectionner le plus performant avant d'ajuster plus finement ses hyper paramètres.

5 Bagging et Boosting

L'objectif de cette section est de comparer les résultats obtenus par méthodes de bagging et de boosting puis d'étudier l'influence des hyper paramètres de l'algorithme AdaBoost. Pour conclure l'étude d'AdaBoost, nous étudierons aussi la contribution des estimateurs de base. Nous utilisons pour cela la base de données relatives au cancer du sein.

5.1 Comparaison des Deux Algorithmes

Comme mentionné au début de l'étude, nous utilisons des méthodes de bagging et de boosting avec comme classifieur de base des arbres de décision. Nous prenons donc comme valeur de référence les résultats obtenus avec un seul arbre de décision sans spécifier d'hyper paramètres. On obtient un taux de bonne classification sur les données de test de 94%.

Comme prévu, un classifieur ensembliste par bagging utilisant 300 classifieurs de base présente de meilleurs résultats. On observe un taux de bonne classification sur données de test de 96%. La diversité des arbres de décisions créés permet d'améliorer légèrement les résultats pour cette application. L'utilisateur doit choisir si ce gain est suffisamment important pour justifier la perte d'interprétabilité des résultats et le temps de calcul plus élevé.

La forêt aléatoire (correspondant à une version améliorée du bagging simple) fournit aussi une légère augmentation du taux de bonne classification en phase de test (96%) pour le même nombre de classifieurs de base. Cela est dû à la diversité supplémentaire induite par le choix aléatoire des attributs étudiés par chaque arbre.

Pour cette application, la méthode de boosting AdaBoost renvoie les meilleurs résultats avec seulement 50 estimateurs de base ayant une profondeur maximale de 1. On observe un taux de bonne classification de test proche de

[6] Nous avons vu qu'en théorie, les performances peuvent se dégrader si le nombre d'estimateurs de base choisi est trop élevé.

98% pour un temps de calcul inférieur aux méthodes de bagging précédentes. Il semblerait que ce soit le classifieur ensembliste le plus adapté pour répondre à ce problème de classification binaire. Poursuivons donc l'étude de cet algorithme et essayons d'obtenir de meilleurs résultats.

5.2 Etude Approfondie de AdaBoost

Ayant décidé d'utiliser l'algorithme AdaBoost pour résoudre le problème de classification binaire sur les données médicales étudiées (car il présente les meilleurs résultats parmi les méthodes choisies), nous cherchons désormais à obtenir les meilleurs résultats possibles (en supposant que nos données ne contiennent aucune erreur). Pour cela, nous étudierons l'impact de trois hyper paramètres sur les résultats : l'estimateur de base, le nombre d'estimateurs de base et le taux d'apprentissage.

5.2.1 Classifieur de Base

L'estimateur de base utilisé peut grandement influencer les résultats. Conformément aux explications en début d'étude, AdaBoost donne de meilleurs résultats avec des weak learners : toutes autres paramètres inchangés, on observe de meilleurs résultats avec des stumps (98% de pourcentage de bonne classification) qu'avec des arbres de profondeur maximale égale à 10 (93% de pourcentage de bonne classification). De la même façon, un modèle de régression logistique donnera de meilleurs résultats qu'un SVC en tant que classifieur de base pour AdaBoost.

Remarque : Ces deux derniers classifieurs de bases ne sont pas des arbres et nécessitent de normaliser les données.

5.2.2 Nombre de Classifieurs de Base

Ayant compris la base de l'apprentissage ensembliste, on pourrait penser que plus le nombre d'estimateurs de base est élevé, meilleurs sont les résultats. Cependant, cette affirmation n'est vraie que dans une certaine mesure : en augmentant trop le nombre de classifieurs de base, le risque de redondance des classifieurs augmente ; si ces classifieurs sont mauvais, les performances risquent de chuter (manque de diversité).

De plus, il faut également trouver un compromis entre qualité des résultats et temps de calcul. AdaBoost avec des stumps donne instantanément 96% d'accuracy avec 5 estimateurs de base, ce qui est très correct. Si on considère 100 stumps, le temps de calcul reste inférieur à une seconde et l'accuracy augmente jusqu'à 98%, il est pertinent de conserver 100 stumps plutôt que 5. Cependant, si on utilise 10000 classifieurs de base, l'accuracy augmente légèrement (98,8%) mais le temps de calcul dépasse désormais 40 secondes : le coût du temps de calcul dépasse sûrement le gain en accuracy.

5.2.3 Taux d'Apprentissage

Le taux d'apprentissage détermine la capacité ou encore la rapidité d'un système à apprendre pour une tâche donnée.

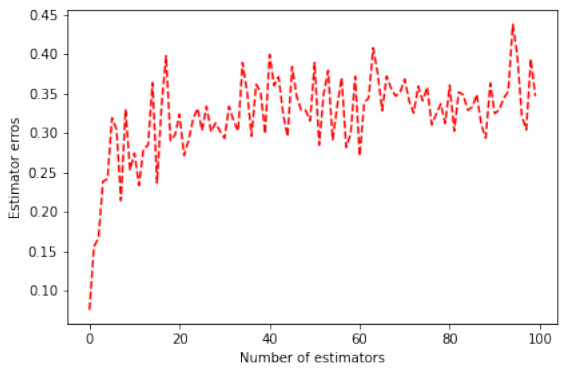
	100	500	1000
0,5	96,491	**98,246**	98,246
1	**98,246**	98,246	98,830
2	69,006	69,006	69,006

Table 1: Table regroupant le pourcentage de bonne classification sur les données de test de l'algorithme AdaBoost pour différents nombres d'estimateurs de base (100, 500, 1000) et taux d'apprentissages (0,5, 1, 2)

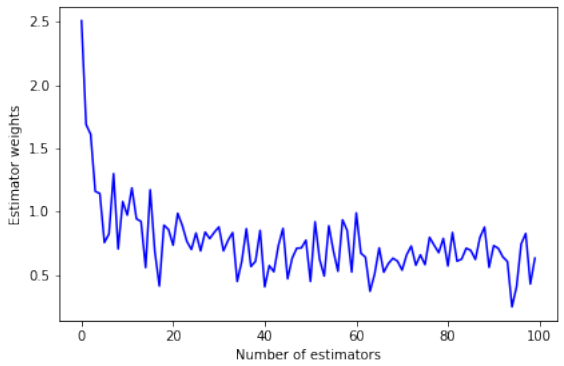
Pour les méthodes d'apprentissage ensembliste, cet hyper paramètre est à étudier en même temps que le nombre d'estimateurs : le fait de ralentir l'apprentissage peut être compensé par une augmentation du nombre de classifieurs de base. Comme on peut le voir dans le Tableau 1, 100 stumps avec un taux d'apprentissage de 1 donne les mêmes résultats que 500 stumps avec un learning rate de 0.5. Une fois de plus, il faut trouver un compromis entre la rapidité d'apprentissage et la quantité de classifieurs de base pour le problème étudié.

5.2.4 Contribution des Classifieurs de Base

On pourrait penser que, chaque classifieur de base apprenant des erreurs des précédents, les derniers classifieurs sont les plus performants et qu'il serait même intéressant de n'utiliser que ces derniers pour la classification finale. Or on observerait des résultats beaucoup moins bons. Le dernier classifieur de base doit s'occuper des individus que tous les précédents n'ont pas réussi à classer correctement. La classification est de plus en plus difficile. Au contraire, les derniers classifieurs de base se voient attribuer un poids moins important : leur décision importe moins dans la décision finale du groupe. On peut observer sur la Figure 15 l'erreur relative à chaque classifieur (qui augmente au cours de l'apprentissage car il devient plus difficile de séparer les individus) et le poids attribué à chacun (qui diminue pour ne pas influencer la décision finale de façon néfaste).



(a) Erreur relative à chaque classifieur de base



(b) Poids relatif à chaque classifieur de base

Figure 15

Remarque : Il est aussi possible d'utiliser AdaBoost pour sélectionner les variables les plus déterminantes pour la classification.

6 Conclusion

Dans cette étude nous avons utilisé plusieurs bases de données plus ou moins grandes et de compexités variables afin de comparer différentes méthodes d'apprentissage ensembliste. En particulier, nous avons utilisé différents algorithmes en utilisant comme classifieur de base les arbres de décision.

Une première étude nous a permis de comparer les arbres de décision simples et les forêts aléatoires, le classifieur ensembliste le plus répandu. Cette étude a notamment montré que l'utilisation d'une forêt aléatoire donne généralement de meilleurs résultats que les arbres de décision mais que ces derniers restent très efficaces pour résoudre les problèmes les plus simples.

Nous avons ensuite mené une étude plus poussée pour comparer les méthodes de bagging et de boosting sur une étude de cas de données médicales. L'algorithme AdaBoost ayant donné les résultats, nous avons poursuivi le travail avec ce classifieur et avons tenté d'adapter les hyper paramètres (classifieur de base, nombre de classifieurs de base et learning rate) pour obtenir les meilleurs résultats possibles.

Cette étude nous a permis de comprendre au travers de plusieurs études de cas les avantages et inconvénients des différents algorithmes d'apprentissage ensemblistes. Ces méthodes peuvent donner de très bons résultats mais elles présentent toutes des limites dont il faut être conscient.

L'ensemble de cette étude nous a une fois de plus montré l'importance de la méthode de développement dans un projet d'apprentissage machine. Une bonne connaissance des algorithmes, une analyse détaillée des données et la recherche de compromis dans l'ajustement des hyper paramètres sont des étapes clefs à respecter et nécessitent un travail d'analyse rigoureux.