N° d'ordre : 10/2019-D/INF

République Algérienne Démocratique et populaire
Ministère de l'Enseignement Supérieur de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE

THESE DE DOCTORAT EN SCIENCES

Présentée pour l'obtention du grade de **Docteur**

En : **INFORMATIQUE**

Spécialité : INTELLIGENCE ARTIFICIELLE ET BASES DE DONNEES AVANCEES

Par **MAHANI AOUATEF**

Thème

Une Approche Mémétique pour le Problème de Fouille de Données Contenant des Phénomènes rares (nuggets discovery)

Soutenue publiquement le 09/10/2019 devant le jury composé de:

Mme. H. F. KHELLAF	Professeur à l'USTHB	Présidente
M. A. R. BABA ALI	Professeur à l'USTHB	Directeur de thèse
Mme. F. BENBOUZID- SI TAYEB	Professeur à l'E.S.I	Examinatrice
Mme. M. BESSEDIK	Maître de conférences A à l'E.S.I	Examinatrice
M. O. KAZAR	Professeur à l'U. Biskra	Examinateur
M. S. KECHID	Professeur à l'USTHB	Examinateur

Chapitre 3 : La classification dans les bases de données déséquilibrées

Chapitre 4 : Le sous-échantillonnage par les algorithmes génétiques (USGA) et le sous-échantillonnage par les algorithmes mémétiques (USGA_ILS)

Chapitre 5 : Etudes expérimentales

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

INTRODUCTION GENERALE

La quantité énorme et croissante des données collectées et stockées dans de vastes et nombreux dépôts de données a largement dépassé la capacité de compréhension humaine, faute d'outils puissants.

Ces données contiennent des connaissances riches qui sont utiles pour l'opération de prise de décision. Cependant, l'extraction de connaissances à partir de ces données est souvent basée sur l'intuition d'un décideur qui ne possède pas d'outils puissants. En outre, la technologie des systèmes experts repose généralement sur des experts du domaine qui saisissent manuellement des connaissances dans des bases de connaissances. Malheureusement, cette procédure est sujette à des biais et des erreurs et elle est extrêmement longue et coûteuse.

Ce besoin a conduit à l'apparition d'un domaine appelé le Data Mining (ou exploration de données) ou découverte de connaissances. Il s'agit d'un domaine interdisciplinaire qui sert à extraire des connaissances de haut niveau à partir des bases de données du monde réel. Le Data Mining est l'étape centrale d'un processus plus large, appelé découverte de connaissances (Knowledge Discovery from Data KDD) dans les bases de données **[1]**.

Le KDD comprend l'application de plusieurs méthodes de prétraitement visant à faciliter l'application de l'algorithme d'exploration de données et des méthodes de post-traitement visant à affiner et à améliorer les connaissances découvertes. Aussi, il permet de bien présenter les connaissances extraites aux utilisateurs par l'utilisation des techniques de visualisation et de représentation des connaissances **[1]**.

Donc, le Data Mining est considéré comme une étape centrale dans le processus KDD. Il est défini comme un processus de découvertes des tendances cachées dans des grandes masses de données **[1].** Il est appliqué dans plusieurs domaines tels que ***la grande distribution, la détection des fraudes, le diagnostic médical, le web Mining et la bioinformatique***.

Contexte de travail :

Le Data Mining traite plusieurs tâches telles que ***le clustering, la régression, la découverte des règles d'association et la classification.***

Dans le Data Mining, les données sont généralement organisées sous forme de ***table*** appelée base de données (dataset) contenant plusieurs lignes et plusieurs colonnes. Chaque ligne représente une seule instance qui correspond à un cas expérimental à analyser et chaque colonne représente toutes les valeurs possibles associées à un ***attribut***. Le dernier attribut représente l'***attribut classe.***

Le degré de distribution de ces instances varie d'une classe à une autre. Cependant, la différence entre les nombres d'instances dans chaque classe peut être grande. Par conséquent, les bases de données existantes sont divisées en deux catégories : les bases de données équilibrées et les bases de données ***déséquilibrées*** (***imbalanced dataset***).Ces dernières sont composées de deux types d'instances : ***majoritaires*** (les plus fréquentes) et ***minoritaires*** (les moins fréquentes).

Notre travail est focalisé sur la classification dans les bases de données déséquilibrées. Ces dernières apparaissent dans plusieurs domaines où l'ignorance des évènements ou des cas rares causent plusieurs problèmes. Par exemple, dans les systèmes de détection d'intrusions où les attaques contre les systèmes informatiques et les réseaux sont en croissance. Différents types d'attaques de réseau existent certaines sont nombreuses et les autres sont rares. Par exemple, les données du concours KDD-CUP'99 contiennent quatre catégories d'attaques réseau : déni de service (DoS), surveillance (la sonde, Probe), distance à local (Remote-to-local R2L) et utilisateur à la racine (User-to-Root U2R). Parmi ces 4 types d'attaques, l'U2R et R2L sont intrinsèquement rares **[2]**.

Problématique:

Les algorithmes classiques de la classification donnent beaucoup d'importances aux classes majoritaires **[3]**. Ils tendent à extraire des règles qui ont des valeurs élevées de l'exactitude et de la couverture. Ces règles sont usuellement spécifiques aux instances majoritaires et les règles qui prédisent les instances minoritaires sont ignorées ou considérées comme bruitées. Par conséquent, les instances minoritaires sont souvent mal classées.

Généralement, comme la majorité des algorithmes de classification sont conçus à minimiser le taux d'erreur global **[4]**, alors quelques problèmes apparaissent. Premièrement, ils ont des mauvaises performances pour les bases de données déséquilibrées et ils produisent soient des règles générales, soient des règles spécifiques. Dans le premier cas, ils s'intéressent aux instances majoritaires et ils ignorent les instances minoritaires. Dans le deuxième cas, ils ont tendance au sur-ajustement des données d'apprentissage qui produisent de mauvaises performances sur les données de test. Deuxièmement, le coût de la mauvaise classification d'une instance minoritaire est généralement plus cher que celui de la mauvaise classification d'une instance majoritaire **[5] [6]**.Finalement, dans de nombreuses applications, un événement rare mal classé peut entraîner des problèmes plus graves qu'un événement habituel **[7]**. Par exemple, dans le cas de détection des cellules cancéreuses, une mauvaise classification des cellules non-cancéreuses peut conduire à des tests cliniques supplémentaires, mais une mauvaise classification des cellules cancéreuses conduit à des risques de santé très graves.

Le problème de la classification dans les bases de données déséquilibrées est un problème fondamental dans l'apprentissage automatique qui a reçu beaucoup d'attention **[8] [9] [10][11][12][13][14]**. Par conséquent, une autre catégorie de classification a été apparue, connue comme étant la classification partielle **[15]** ou nugget discovery **[16]** ou problème de classification dans les bases de données déséquilibrées **[17]** ou les bases de données avec des classes rares **[18]**.

Les solutions proposées :

Les différentes techniques et approches proposées pour traiter le problème de la classification dans les bases de données déséquilibrées sont divisées en trois niveaux **[19]** : le niveau données, le niveau algorithme et le niveau hybride. Les approches du premier niveau altèrent la distribution des données dont le but de diminuer le degré du déséquilibre, elles sont divisées en 3 groupes : le sous-échantillonnage supprime quelques instances majoritaires, le sur-échantillonnage duplique quelques instances minoritaires et hybride combine les deux groupes précédents. Les approches du deuxième niveau sont divisées en deux catégories : la proposition des algorithmes spécifiques et la modification des algorithmes de la classification complète afin de les adapter à la classification dans les bases de données déséquilibrées. Dans le niveau hybride, quelques méthodes de la classification complète doivent être combinées avec d'autres techniques pour qu'elles puissent traiter ce problème.

Nos contributions :

Nous rappelons que l'objectif de notre thèse est l'extraction d'un modèle à partir des bases de données déséquilibrées, qui doit contenir des règles de classification représentant les instances majoritaires et minoritaires. Pour atteindre cet objectif, nous proposons une approche hybride à trois phases. La première phase consiste à équilibrer la base de données déséquilibrée en utilisant une méthode intelligente de sous-échantillonnage. Elle consiste à supprimer les instances majoritaires et les remplacer par des règles de classification pour conserver les concepts contenus dans ces instances. Dans la deuxième phase, nous utilisons la base de données équilibrée pour extraire le deuxième modèle contenant des règles de classification représentant les instances majoritaires et minoritaire. Enfin, dans la troisième phase, nous fusionnons les deux modèles précédents pour construire le troisième modèles qui va subir une phase de post-traitement pour supprimer les règles redondantes.

Nous utilisons les algorithmes génétiques dans la première phase pour construire le premier modèle, d'où le nom que nous avons attribué à la première approche : le sous-

échantillonnage par les algorithmes génétiques USGA (Under-Sampling by Genetic Algorithm).

Dans la deuxième phase, nous utilisons deux approches évolutionnaires différentes : l'approche génétique pure USGA et l'approche mémétique USGA_ILS qui résulte de l'intégration de la méthode de recherche locale itérée ILS dans le cycle génétique où nous apportons des modifications sur elle, afin de l'adapter à notre problème.

Organisation de la thèse :

Afin de présenter les travaux de cette thèse, nous avons retenu une organisation qui s'articule autour de cinq chapitres et d'une conclusion générale.

Nous abordons dans le premier chapitre ***« les méthodes d'optimisation combinatoires »***, nous présentons quelques méthodes de résolution qui sont classées en deux catégories : les méthodes à base de solution unique telles que les méthodes de descente et les méthodes de recherches locales itérées ILS, et les méthodes à base de population telles que les algorithmes génétiques et les algorithmes mémétiques.

Le deuxième chapitre ***« Le Data Mining : Concepts et définitions »*** présente l'historique du Data Mining ainsi que ses domaines d'application et ses tâches. Aussi, nous nous focalisons sur la classification, plus précisément sur la classification supervisée en définissant ses concepts de base, en présentant ses mesures de performances utilisées et en finissant par décrire ses différentes méthodes et approches utilisées.

Le troisième chapitre ***« La classification dans les bases de données déséquilibrées »*** s'intéresse à ce type de classification. Nous commençons d'abord par présenter ce problème ainsi que ses domaines d'application. Puis, nous montrons les insuffisances des mesures de performances utilisées par la classification complète afin de présenter les mesures les plus appropriées pour la classification partielle. Nous finissons par présenter les différentes méthodes et approches spécifiques à notre problème.

Le quatrième chapitre ***« Nos contributions »*** est divisé en deux sections : la première section présente un état de l'art sur les approches qui utilisent les algorithmes évolutionnaires pour traiter le problème de la classification dans les bases de données déséquilibrées. La deuxième section présente nos contributions. Elle est divisée en deux parties pour présenter toutes les étapes nécessaires de l'approche génétique USGA et de l'approche mémétique USGA_ILS.

Dans le cinquième chapitre ***« Etudes expérimentales »***, nous procédons à une série de tests sur 38 bases de données de KEEL afin de prouver l'efficacité et la robustesse de nos approches proposées. Pour cela, nous présentons tous les outils utilisés et les résultats obtenus

en utilisant diverses mesures de performances. Ainsi, nous présentons les études comparatives suivantes en utilisant les tests statistiques: d'abord entre l'approche USGA et quelques approches existantes dans la littérature. Ensuite, entre USGA_ILS et quelques approches existantes dans la littérature. Enfin, entre USGA et USGA_ILS afin de voir l'utilité de l'approche génétique hybride par rapport à l'approche génétique pure.

Enfin, la dernière partie ***« Conclusion et perspectives »*** récapitule nos travaux de recherche et propose quelques perspectives de recherches futures.

Chapitre 1 :

LES METHODES D'OPTIMISATION COMBINATOIRES

1. Introduction :

Un problème d'optimisation combinatoire consiste à trouver la (ou les) *meilleure* solution (s) dans un ensemble discret dit espace de recherche. En général, cet ensemble est fini mais compte un très grand nombre d'éléments. Il est décrit de manière implicite, c'est-à-dire par une liste, relativement courte, de contraintes que doivent satisfaire les solutions réalisables.

Trouver une solution optimale dans un ensemble discret et fini est un problème facile en théorie : il suffit d'essayer toutes les solutions et de comparer leurs qualités pour voir la meilleure. Cependant, en pratique, l'énumération de toutes les solutions peut prendre trop de temps; or, le temps de recherche de la solution optimale est un facteur très important. Par conséquent, les problèmes d'optimisation combinatoire sont réputés difficiles.

2. Les méthodes de résolution des problèmes d'optimisation combinatoire :

La résolution d'un problème d'optimisation combinatoire signifie la détermination de la solution optimale qui nécessite un parcours ou une exploration de l'espace de recherche.

L'espace de recherche est constitué de toutes les solutions qui satisfont le problème. L'exploration de l'espace de recherche permettra la sélection des solutions qui sont potentiellement de bonne qualité dans le but d'atteindre à la fin de ce processus la solution optimale. Ces méthodes sont divisées en deux catégories :

- Les méthodes à base de solution unique.
- Les méthodes à base de population.

2.1. Les méthodes à base de solution unique :

Dans les problèmes d'optimisation où nous cherchons à optimiser une fonction objective sur un espace de décision donné, une petit changement sur un point de cet espace induit souvent une petite variation des valeurs de la fonction objective en ce point.

Nous déduisons que les bonnes solutions ont tendance à trouver à proximité d'autres bonnes solutions, les mauvaises étant proches d'autres mauvaises solutions. D'où l'idée qu'une bonne stratégie consisterait à se déplacer à travers l'espace de recherche en effectuant de petits pas (petits changements sur le point courant) dans des directions qui améliorent la fonction objective. Cette idée est à la base d'une grande famille d'algorithmes appelée ***méthodes de recherche locale.***

La recherche locale est une technique très utilisée en pratique pour aborder des problèmes d'optimisation combinatoire difficiles. L'idée naturelle, est d'améliorer de façon itérative une solution existante en explorant un voisinage de celle-ci. Les solutions dites voisines sont généralement obtenues en appliquant des transformations (aussi appelées mouvements) plus ou moins importantes (c'est-à-dire plus ou moins locales) à la solution courante **[20]**.

Parmi les méthodes de recherche locale, nous présentons: les méthodes de descente et les méthodes itérées.

2.1.1. Les méthodes de descente :

Les méthodes de descente sont classiques et très rapides **[21]**. Ses étapes de fonctionnement pour un problème de minimisation sont décrites dans l'***algorithme 1.1***.

Algorithme 1.1. Les méthodes de descente

Entrée : l'espace de recherche Ω.
la fonction objective f.
la solution initiale S.
Sortie : l'optimum local S.
Début
Répéter
Choisir le voisin S' de S avec $f(S) \leq f(S')$;
S :=S' ;
Jusqu'à ce que (S soit le meilleur de tous ses voisins) ;
Fin.

2.1.2. Les méthodes itérées (Iterated Local Search ILS) :

Les méthodes de descente sont caractérisées par la rapidité et la simplicité de la mise en œuvre. Mais, elles ne permettent pas d'obtenir les meilleurs optima car elles s'arrêtent dés qu'elles trouvent un optimum local. Pour ne pas rester bloqué sur cet optimum local, il existe différentes stratégies qui permettent de poursuivre la recherche après avoir trouvé un optimum. D'où l'apparition des méthodes itérées.

Les méthodes itérées combinent plusieurs structures de voisinage pour la recherche dans les problèmes disposant de nombreux optimums locaux.

a. Description des méthodes itérées :

Elles reposent sur les combinaisons d'une méthode de recherche locale et des perturbations pour permettre le poursuivre de recherche après avoir trouvé l'optimum. ILS est une marche dans l'espace des optimums locaux. Le modèle de base de la méthode ILS est donné dans l'***algorithme 1.2*** **[22]**

Algorithme 1.2. Modèle de la méthode ILS

Début
s_0 :=Génération de la solution initiale ();
s^* :=Recherche Locale (s_0);
Répéter
s' := Perturbation(s^*, historique);
s'^* := Recherche Locale (s');
s^* := Critère d'acceptation (s^*, s'^*, historique);
Jusqu'à ce que le critère d'arrêt est atteint ();
Fin.

b. Les modules de la méthode ILS :

Les modules importants de cette méthode sont la génération de la solution initiale, la méthode de recherche locale, la perturbation et enfin le critère d'acceptation. La performance de la méthode ILS repose sur tous ces critères. Dans les sous sections suivantes, nous détaillons chaque module.

b.1. Génération de la solution initiale :

La solution initiale s_0 représente le point de départ de la marche dans l'ensemble des solutions S^*. Commencer par un bon point peut être important si des solutions de haute qualité doivent être atteintes le plus rapidement possible.

Les choix standards pour le point de départ sont soit une solution initiale aléatoire, soit une solution retournée par une heuristique de construction gourmande (greedy construction heuristic)

b.2. Recherche locale :

Elle utilise une structure de voisinage assez simple qui ne permet pas de s'échapper des optimums locaux.

b.3. Perturbation :

Elle est basée sur l'utilisation d'un voisinage plus large permettant de modifier une grande partie de la solution et ainsi de tenter de s'échapper des optimums locaux. Elle sert à la diversification.

Selon Lourenço et al. : "Une bonne perturbation transforme une bonne solution en un excellent point de départ pour une méthode de recherche locale" **[22][23]**.

La perturbation est caractérisée par la portée (appelé aussi degré) de la perturbation qui représente le nombre de composants qui vont être modifiés. La portée a une influence capitale sur le fonctionnement de la méthode :

Si la portée est faible, la perturbation ne permet pas de s'échapper des optimums locaux et de diversifier la recherche.

Si la portée est élevée, la perturbation effectue une relance aléatoire de la méthode et ne permet pas la convergence.

Il existe différentes stratégies proposées dans le passé pour gérer la force de la perturbation pendant la recherche:

- **Statique:** la portée est fixée à l'avance et elle ne sera plus modifiée pendant la recherche.
- **Dynamique :** la portée est modifiée dynamiquement pendant la recherche sans que l'historique de recherche ne soit pris en compte.

- **Adaptatif :** la longueur de perturbation est modifiée de manière dynamique pendant la recherche en fonction de l'historique de recherche.

b.4. Critère d'acceptation :

Cette procédure sert à déterminer si la solution $s^{*'}$ est acceptée ou non comme une nouvelle solution courante. Ce critère a une forte influence sur la nature et l'efficacité de la marche dans l'espace des solutions S^*. Ce critère contrôle la balance entre l'intensification et la diversification dans l'espace de recherche. Parmi les stratégies du critère d'acceptation existantes : le meilleur critère et la marche aléatoire.

b.4.1. Le Meilleur critère d'acceptation *:*

Dans cette stratégie, la meilleure solution est acceptée **[23]**. Par conséquent, une forte intensification est achevée. Pour un problème de minimisation, nous écrivons

$$\boldsymbol{Meilleur}(\boldsymbol{s}^*, \boldsymbol{s}^{*'}, \boldsymbol{historique}) = \begin{cases} \boldsymbol{s}^{*'} & \boldsymbol{si} \quad \boldsymbol{C}(\boldsymbol{s}^{*'}) < C(\boldsymbol{s}^*) \\ \boldsymbol{s}^* & \boldsymbol{sinon} \end{cases} \qquad \textbf{\textit{(1.4)}}$$

b.4.2. La marche aléatoire:

La marche aléatoire (RandomWalk) RW choisit toujours l'optimum local récemment visité, quel que soit son coût **[23]**. Ce critère favorise fortement la diversification, car toute solution en S^* est acceptée pour la prochaine étape.

$$\boldsymbol{RW}(\boldsymbol{s}^*, \boldsymbol{s}^{*'}, \boldsymbol{historique}) = \boldsymbol{s}^{*'} \qquad \textbf{\textit{(1.5)}}$$

b.5. Le critère d'arrêt de méthode ILS :

Les critères d'arrêt courant sont le temps d'exécution et le nombre d'itérations.

c. La mémoire de la méthode ILS :

En pratique, une grande partie de la complexité potentielle de la méthode ILS est cachée dans la dépendance de l'historique. S'il n'y a pas de telle dépendance, la marche dans l'espace de recherche n'a pas de mémoire: le critère d'acceptation ne dépend d'aucune des solutions visitées précédemment. Cela conduit à des marches aléatoires dynamiques qui sont "Markoviennes". La plupart des travaux utilisant ILS sans mémoire, bien que des études récentes montrent sans ambiguïté que l'intégration de la mémoire améliore les performances **[24]**.

2.2. Les méthodes à base de population :

Cette classe de métaheuristiques travaille explicitement avec une population de solutions. Dans cette classe, nous pouvons distinguer les algorithmes génétiques, les algorithmes mémétiques et la recherche dispersée.

2.2.1. Les algorithmes génétiques :

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle : sélection, croisement, mutation, etc **[25] [26]**.

Un algorithme génétique cherche le ou les extrema d'une fonction définie sur un espace de recherche. Pour l'utiliser, nous devons disposer les cinq éléments suivants :

1. Un principe de codage de la population.
2. Un mécanisme de génération de la population initiale.
3. Une fonction à optimiser.
4. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace de recherche.
5. Des paramètres de dimensionnement : taille de la population, nombre total de générations, critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

Le principe général du fonctionnement d'un algorithme génétique est représenté sur la ***figure 1.1.***

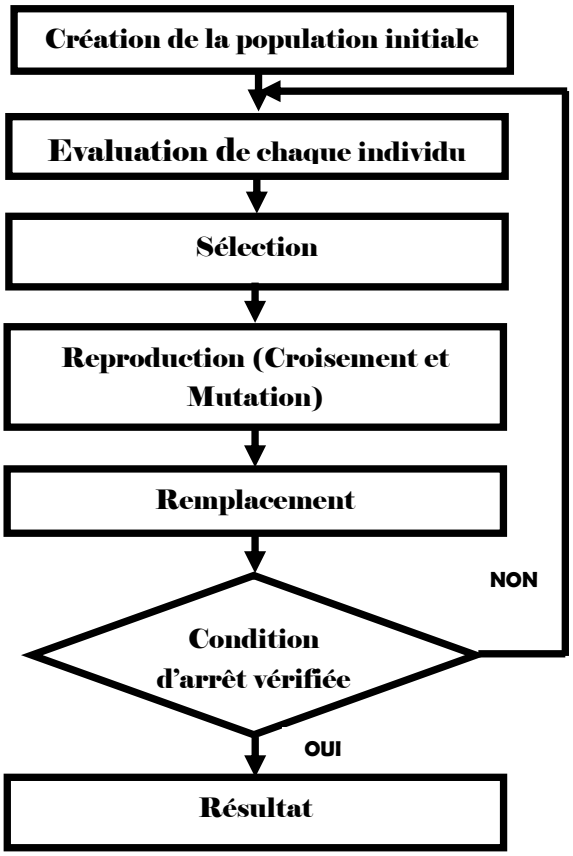

Figure 1.1. ***Processus de l'algorithme génétique***

a. Codage :

Le codage est le premier pas dans l'implémentation d'un algorithme génétique **[27]**. Le choix du codage dépend du problème traité et conditionne son efficacité (vitesse de convergence, précision,…).

b. Sélection :

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer partiellement les mauvais **[26]**. Parmi les méthodes de sélection, nous avons la loterie biaisée et le tournoi.

c. Condition d'arrêt:

Une condition d'arrêt peut être fixée de diverses manières, les plus utilisées sont :

- La convergence de la population : c'est-à-dire la population devient stable ce qui engendrera les générations futures « stériles » dans le sens où il ne se produit pas d'amélioration dans le fitness des individus **[28].**
- Un certain nombre de générations **[28]**.

d. Les opérateurs génétiques de reproduction :

d.1. Croisement :

Cet opérateur mélange les gènes de deux individus, sans toucher à leurs contenus. Il réalise la reproduction entre les individus de la population **[29]**. Il s'applique avec une certaine probabilité P_{cro}. Il peut être effectué avec plusieurs manières selon le nombre de points de coupure, d'où l'existence de plusieurs méthodes telles que **le croisement en un point** (voir l'exemple donné dans ***la figure 1.2***) et **le croisement en k points** (voir l'exemple donné dans ***la figure 1.3***).

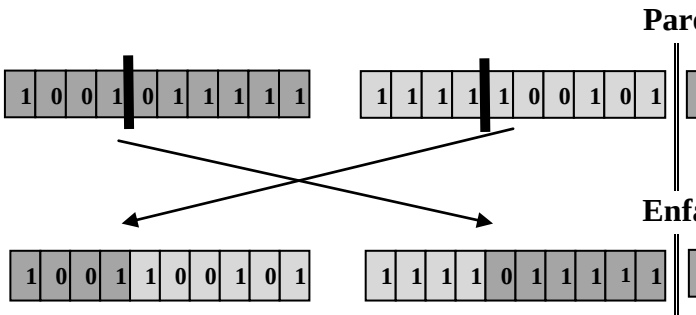

Figure 1.2. ***Exemple de croisement en un point***

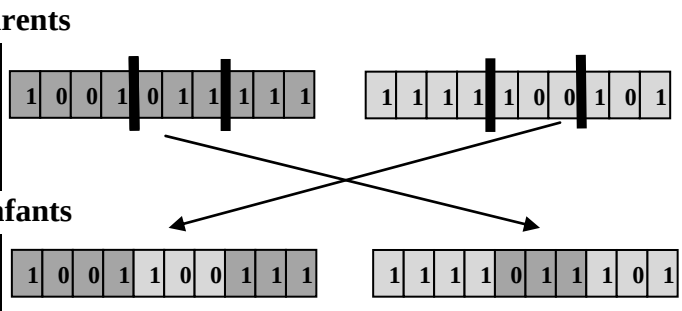

Figure 1.3. ***Exemple de croisement en k points***

d.2. Mutation :

La mutation d'un individu consiste à remplacer un de ses gènes par une autre valeur choisie aléatoirement. Elle s'applique généralement avec une faible probabilité P_{mut}. Le but

est de permettre une meilleure recherche locale et de maintenir la diversité génétique utile pour une bonne exploration de l'espace de recherche. La ***figure 1.4*** illustre un exemple de mutation.

Figure 1.4. ***Exemple de mutation***

e. Evaluation des individus de la population :

Chaque individu de la population doit être évalué en calculant la valeur de sa fonction objective qui représente son degré d'adaptation à son environnement.

2.2.2. Les algorithmes mémétiques :

Les algorithmes mémétiques ont été proposés par Moscato 1989, ils s'inspirent de certains modèles d'adaptation dans la nature, qui combinent l'évolution adaptative de populations d'individus avec l'apprentissage des individus au cours de leur vie. Le terme "algorithme mémétique" trouve son origine dans le concept de "meme" introduit par Richard Dawkins **[30]** qui représente une unité d'information évoluant avec les échanges d'idées entre individus. Une différence fondamentale entre les notions de gènes (manipulés par les algorithmes génétiques) et de "memes" est que ces derniers sont adaptés par la personne qui les transmet (en introduisant ses réflexions et ses déductions personnelles), alors que les gènes sont transmis tels quels.

Ces algorithmes se retrouvent dans la littérature sous plusieurs autres noms : algorithmes génétiques hybrides, recherche locale génétique, algorithmes évolutionnaires Baldwiniens, algorithmes évolutionnaires Lamarkiens et autres.

Les algorithmes mémétiques sont des algorithmes évolutionnaires hybridés avec des méthodes de recherche locale. Cette hybridation est destinée à accélérer la découverte de bonnes solutions pour laquelle la seule évolution serait trop longue à découvrir. Elle est également destinée à trouver des solutions qui, autrement, seraient inaccessibles par une évolution ou une seule recherche locale. La recherche évolutionnaire fait une large exploration de l'espace de solution tandis que la recherche locale permet en quelque sorte de "zoomer" des solutions prometteuses.

L'intégration d'une méthode de recherche locale dans un algorithme génétique peut se faire avec les stratégies suivantes :

- Après la sélection.
- Après le croisement.

- Après la mutation
- Par combinaisons des stratégies précédentes. Par exemple, après le croisement et après la mutation.

3. Conclusion:

Nous avons vu dans ce chapitre plusieurs méthodes de résolution des problèmes d'optimisation combinatoire où chaque méthode a ses propres caractéristiques. Cependant, il faut choisir la bonne méthode pour résoudre un problème donné. Par exemple, dans le cas où nous voulons résoudre un problème qui a un grand espace de recherche, les méthodes à base de population seront utilisées.

Chapitre 2 :
LE DATA MINING : CONCEPTS ET DEFINITIONS

1. Introduction :

Le Data Mining peut être considérée comme le résultat de l'évolution naturelle de la technologie de l'information.

Depuis les années 1960, la base de données et la technologie de l'information évoluaient systématiquement les systèmes de traitement de fichiers primitifs vers les systèmes de bases de données sophistiqués et puissants.

La recherche et le développement dans les systèmes de base de données depuis les années 1970 évoluaient les systèmes de base de données hiérarchiques et réseau vers le développement des systèmes de bases de données relationnels, des outils de modélisation des données et des méthodes d'indexation et d'accès aux données.

La technologie de base de données depuis le milieu des années 1980 a été caractérisée par l'adoption populaire de la technologie relationnelle et la recrudescence des activités de recherche et de développement des systèmes de bases de données nouveaux et puissants. Ceux-ci favorisent le développement des modèles de données avancés tels que les modèles relationnels étendus, les modèles orientés-objets et les modèles objets-relationnels.

Les progrès constants et étonnants de la technologie du matériel informatique au cours des trois dernières décennies ont mené à des fournitures importantes d'ordinateurs puissants, d'équipements de collecte des données et de supports de stockage. Par conséquent, cette technologie a fournit un formidable coup de pouce à l'industrie de la base de données et de l'information et rend un grand nombre de bases de données et de référentiels d'information disponibles pour la gestion des transactions, la recherche d'information et l'analyse des données.

Les données peuvent être maintenant stockées dans différents types de bases de données et de référentiels d'informations. Une architecture de référentiel de données qui a émergé est l'entrepôt de données. En outre, d'énormes volumes de données peuvent être accumulés au-delà des bases de données et des entrepôts de données.

Ces données contiennent des connaissances riches qui sont utiles pour l'opération de prise de décision. Cependant, l'extraction de connaissances à partir de ces données était souvent basée sur l'intuition d'un décideur qui ne possédait pas des outils puissants. En outre, la technologie des systèmes experts reposait généralement sur des experts du domaine qui saisissaient manuellement des connaissances dans des bases de connaissances. Malheureusement, cette procédure est sujette à des biais et des erreurs et elle est extrêmement longue et coûteuse. Ce besoin a conduit à l'apparition d'un domaine appelé le Data Mining.

Dans ce chapitre, nous allons d'abord définir le Data Mining ainsi que ses domaines d'application et ses tâches en se focalisant sur la classification.

2. Définition du Data Mining :

Le Data Mining est connu sous plusieurs noms tels que « exploration de connaissances à partir des données» ou « exploration de connaissances » ou « extraction de connaissances ». De nombreuses personnes considèrent le Data Mining comme un synonyme d'un autre terme couramment utilisé : Extraction de connaissances à partir des données KDD (Knowledge Discovery from Data). Cependant, KDD consiste en une séquence itérative des étapes suivantes **[1]**:

1. Nettoyage des données : est appliqué pour supprimer les données bruyantes et corriger les données incohérentes.

2. Intégration des données : sert à collecter des données provenant de différentes sources et de les fusionner dans une structure de données cohérente.

3. Sélection des données : sélectionne les données pertinentes à partir d'une base de données pour l'analyse.

4. Transformation des données : consiste à transformer les données dans des formes appropriées pour l'exploration en effectuant des opérations de synthèse ou d'agrégation.

5. Le Data Mining : est un processus de découverte des tendances cachées dans de grandes masses de données.

6. Evaluation des modèles : identifie les modèles les plus intéressants par l'utilisation de mesures d'intérêt.

7. Représentation des connaissances : les techniques de visualisation et de représentation de connaissances extraites sont utilisées pour les présenter à l'utilisateur.

3. Les domaines d'application du Data Mining:

Le Data Mining est appliqué dans plusieurs domaines, tels que:

3.1. Détection des fraudes :

Les techniques du Data Mining sont également appliquées à la détection des fraudes, notamment dans les domaines où plusieurs transactions s'effectuent simultanément ce qui rend le système vulnérable aux fraudes (ex : cartes de crédit, télécommunications, systèmes informatiques…etc.) **[31] [32].**

3.2. Diagnostic médical :

Certains outils, par l'application de méthodes de clusterisation ou d'extraction de règles d'association, à partir des symptômes d'un patient ou de ses données pathologiques (ex :

radiographies, électrocardiogrammes), sont capables d'effectuer des diagnostics plus fiables et plus objectifs que ceux des médecins **[33].**

3.3. Web Mining :

L'application du Data Mining sur le Web, vise à rendre la navigation plus personnalisée et plus adaptée aux besoins des utilisateurs **[34]**. Parmi ces applications, nous citons:

- La déduction des relations d'associations entre les passages sur différents sites.
- La prévision d'achats sur un site donné en fonction d'autres observations : impact sur la stratégie de communication.
- L'étude de l'efficacité de bannières (publicitaires) et autres techniques de référencement.
- La segmentation des profils des visiteurs en fonction des comportements de navigation sur Internet.

3.4. Bioinformatique :

Les approches du Data Mining semblent parfaitement adaptées à la bioinformatique **[35] [36]**, car elles sont riches en données. L'extraction de données biologiques aide à extraire des connaissances utiles à partir des bases de données massives recueillies en biologie et dans d'autres domaines comme la médecine et les neurosciences. Les applications du Data Mining à la bioinformatique incluent la découverte de gènes, l'inférence de fonction de protéine, le diagnostic de maladie et l'optimisation des traitements.

4. Représentation des données dans le Data Mining :

Dans le Data Mining, les données sont généralement organisées sous forme de ***table*** appelée base de données (dataset) qui contient plusieurs lignes et plusieurs colonnes. Chaque ligne représente une seule instance qui correspond à un cas expérimental à analyser et chaque colonne représente toutes les valeurs possibles associées à un ***attribut***. Les premiers attributs représentent ***les attributs prédictifs*** et le dernier attribut représente l'***attribut classe*** ou ***l'attribut à prédire.*** L'ensemble des valeurs de l'attribut classe représente l'ensemble des catégories que nous attribuons à chaque instance **[37] [38]**. Le type d'un attribut peut être nominal ou numérique.

Exemple :

Soit la base de données client représentée dans ***le tableau 2.1***. Elle contient deux attributs nominaux : marié et catégorie client et deux attributs numériques : Age et Revenu. Selon les valeurs de chaque attribut, un client peut être crédible ou non crédible.

Tableau 2.1.*Exemple d'une base de données*

	Attributs prédictifs			*Attribut classe*
	Marié	Age	Revenu	Catégorie client
Instances	Oui	44	1200	Crédible
	Non	32	800	Non Crédible
	Non	26	1000	Non Crédible
	Oui	36	6000	Crédible
	Non	56	3000	Crédible

5. Les tâches du Data Mining :

Parmi les tâches du Data Mining, nous avons le clustering, la régression, la découverte des règles d'association et la classification. Dans les sous sections suivantes, nous décrivons chaque tâche.

5.1. Clustering:

Est le processus de regroupement des données en clusters où les instances de chaque cluster ont une grande similarité, mais sont différentes des instances d'autres clusters **[1]**. Les algorithmes de clustering sont classés en deux catégories selon la classification proposée par Farley et Raftery **[39]** : les algorithmes hiérarchiques et les algorithmes à partitionnement.

5.1.1. Les algorithmes hiérarchiques :

Ces algorithmes sont divisés en deux classes : les approches ascendantes et les approches descendantes. Le fonctionnement des approches **[40]** de la première classe consiste à créer d'abord un cluster pour chaque classe, puis de fusionner successivement les 2 clusters les plus proches. Par contre, le principe des approches **[41]** de la deuxième classe, consiste à mettre toutes les instances dans le même cluster, puis diviser un cluster en sous clusters jusqu'à ce qu'il y ait suffisamment de niveaux ou que les clusters ne contiennent plus qu'une seule instance.

5.1.2. Les algorithmes à partitionnement :

Dans ces algorithmes, le nombre de clusters est fixé à l'avance. Leur principe consiste à créer une partition initiale de K clusters, puis itérer un processus qui optimise le partitionnement en déplaçant les instances d'un cluster à un autre. Parmi ces algorithmes, nous citons K-Moyenne (K-Mean) **[42]** et K-Medoids **[41] [43].**

5.2. Régression :

La régression consiste à trouver une relation entre les attributs prédictifs (X_1, X_2, …, X_n) d'une base de données et l'attribut à prédire Y **[44]**. Cette relation est sous la forme suivante :

$$\boldsymbol{Y = \theta_1 * X_1 + \theta_2 * X_2 + \cdots + \theta_n * X_n + e} \qquad \textbf{(2.1)}$$

Le processus d'apprentissage consiste à trouver les meilleures valeurs des paramètres θ_1, θ_2, …, θ_n qui minimisent une mesure d'erreur.

Par exemple, la régression linéaire consiste à chercher une relation entre deux variables seulement si cette relation peut être approximée par une droite **[1]**.

Une relation linéaire est exprimée par la fonction suivante :

$$\mathbf{Y = \theta_2 * X + \theta_1} \qquad \textit{(2.2)}$$

Où :

- θ_1 : est l'ordonnée à l'origine, c'est-à-dire le point x=0.
- θ_2 : est la pente. Elle représente l'angle entre une donnée et la ligne de régression.

θ1 et θ2 sont appelés les coefficients de régression. Ils peuvent être calculés par la méthode de moindres carrés qui cherche à minimiser l'erreur entre une donnée et la ligne estimée **[44]**. Les formules de calcul des coefficients sont données dans les équations (2.3) et (2.4).

$$\boldsymbol{\theta_2} = \frac{\sum_{i=1}^{|D|}\left((x_i - \bar{x}) * (y_i - \bar{y})\right)}{\sum_{i=1}^{|D|}(x_i - \bar{x})} \qquad \textit{(2.3)}$$

$$\boldsymbol{\theta_1} = \bar{y} - \boldsymbol{\theta_2} * \bar{x} \qquad \textit{(2.4)}$$

Où :

- |D| : est la taille de la base d'apprentissage D où chaque instance de D est représentée par le couple (x_i,y_i).
- x : est la valeur de la variable prédictive.
- y : est la valeur de la variable à prédire.
- $\bar{x}$: est la moyenne des valeurs $x_1, x_2, \ldots, x_{|D|}$.
- $\bar{y}$: est la moyenne des valeurs $y_1, y_2, \ldots, y_{|D|}$.

5.3. Découverte des règles d'association :

Cette tâche a pour but la découverte des associations fréquentes entre les items dans les bases de données transactionnelles **[45]**. Elle est utilisée dans plusieurs domaines tels que le panier de la ménagère **[46]** pour extraire les produits qui se vendent ensemble à partir d'une base de données contenant des achats.

Une association est représentée par une règle d'association de la forme : C→P où C est la partie condition et P est la partie prédiction.

Il existe plusieurs algorithmes d'extraction des règles d'association tels que l'algorithme Apriori **[47]** et Apriori-TFP **[48]**. Par exemple, Apriori a été développé par AGRAWAL en 1994. Il permet d'extraire toutes les règles dont le support et la confiance sont supérieurs aux

seuils MINSUP et MINCONF fixés arbitrairement. La confiance et le support sont définis comme suit :

Confiance : représente la fréquence de P lorsque la condition C est vérifiée. Elle est définie dans l'équation (2.5).

$$\boldsymbol{Confiance = \frac{|C\ et\ P|}{|C|}} \qquad \textbf{(2.5)}$$

Support : représente la fréquence de toute la règle. Il est défini dans l'équation (2.6).

$$\boldsymbol{Support = |C\ et\ P|} \qquad \textbf{(2.6)}$$

Les étapes de l'algorithme Apriori sont les suivantes:

1. La recherche des itemsets fréquents, qui sont les groupes d'items possédant un support supérieur à MINSUP.
2. L'extraction des règles d'association : les itemsets fréquents qui possèdent une confiance supérieure à MINCONF sont utilisés pour extraire les règles d'association.

5.4. Classification :

Est la tâche la plus populaire du Data Mining. Elle consiste à assigner à chaque instance une classe choisie à partir d'un ensemble de classes prédéfinies, en fonction de la valeur de certains attributs prédictifs **[47].**

Le problème de la classification permet de classer correctement une instance de classe indéterminée. Ce classement peut se faire par plusieurs méthodes qui sont divisées en deux catégories : la première catégorie s'appuie sur l'utilisation d'un modèle comme les arbres de décision et les règles de classification. Cependant, la deuxième catégorie se base sur le fonctionnement interne de l'algorithme d'apprentissage tel que les réseaux de neurones **[49]** et les machines de vecteurs du support.

La classification est divisée en deux catégories : la classification supervisée et la classification non supervisée. Dans la première catégorie, les données d'apprentissage sont étiquetées, c'est-à-dire, les valeurs de l'attribut classe sont connues. Cependant, les données d'apprentissage ne sont pas étiquetées dans la deuxième catégorie. Dans ce chapitre, nous nous focalisons sur la classification supervisée.

5.4.1. Représentation des connaissances :

Dans le contexte de la classification, la connaissance extraite se présente généralement sous la forme d'un ensemble de règles de classification **[50]**. Généralement, une règle de classification a la forme suivante :

SI *condition* **ALORS** *conséquent*

La partie condition d'une règle peut être représentée sous différentes formes. Par exemples : la représentation de GABIL et la représentation de HIDER.

Le ***conséquent*** (C) est constitué de la classe prédite par la règle. En général, il est sous la forme : classe=valeur.

L'ensemble de règles forme un modèle de classification ou simplement un ***modèle*** **[51]**. Ilest appelé aussi ***modèle***.

a. GABIL :

Dans la représentation de GABIL, la condition est sous la forme normale conjonctive (**FNC**) **[52]**:

A1= (V_1^1 OU V_1^2…OU V_1^m) ET A2= (V_2^1 OU V_2^2…OU V_1^k) ET…ET An= (V_n^1 OU V_n^2…OU V_n^l)

Où A_i est le ième attribut du problème et V_i^j est la jième valeur du l'ième attribut.

Exemple :

Soient les attributs suivants :

- Ciel = {clair, partiellement nuageux, nuages sombres}
- Pression = {Basse, Moyenne, Haute}
- La classe = {pas de pluie, pluie}

Donc, une règle pourrait être:

Si (ciel est partiellement nuageux OU nuages sombres) ET (pression est faible) alors classe= pluie

b. HIDER :

Dans la représentation de HIDER, la condition est constituée de plusieurs conditions reliées par l'opérateur logique ET où la représentation de chaque condition dépend du type de l'attribut qui lui correspond. Pour un attribut continu, la condition est représentée par un intervalle qui est inclus dans l'intervalle global dont les bornes représentent les valeurs inférieure et supérieure de cet attribut. Pour un attribut discret, la condition contient quelques valeurs possibles de cet attribut **[53]**.

Exemple :

Soient les attributs :

- A_1 est un attribut continu, ses valeurs appartiennent à l'intervalle [1.4, 6.2].
- A2 est un attribut discret, ses valeurs appartiennent à l'ensemble {noir, vert, bleu, blanc, rouge}.
- Classe est un attribut discret, ses valeurs appartiennent à l'ensemble {0, 1}.

Une règle pourrait être :

Si $A_1 \in [3.9, 5.0]$ et $A_2 \in$ {vert, blanc, rouge} alors classe=0

5.4.2. Evaluation d'une règle de classification :

La qualité d'une règle, une fois établie, est à évaluer avec grand soin, car le plus important pour les utilisateurs de ces règles est qu'elles soient pertinentes, intéressantes et plus lisibles par rapport à son langage et plus précises par rapport à son domaine **[45]**. Pour cette raison, ils existent plusieurs mesures d'évaluation de qualité d'une règle de classification telles que :

a. Couverture:

La couverture d'une règle R est le pourcentage d'instances dans la base de données qui sont couvertes par R **[1]**.

$$\boldsymbol{couverture(R) = \frac{n_{couvertes}}{|D|}} \qquad \textit{(2.7)}$$

Où :

- $n_{couvertes}$: est le nombre d'instances couvertes par la règle R.
- |D| : est la taille de la base de données.

Une instance est couverte par R si elle vérifie la partie condition de R (peu importe la classe).

b. Exactitude :

L'exactitude d'une règle R représente le pourcentage d'instances de la base de données qui sont bien classées par R **[1]**. Cette mesure est appelée aussi la confiance **[54].**

$$\boldsymbol{Exactitude(R) = \frac{n_{correctes}}{n_{couvertes}}} \qquad \textit{(2.8)}$$

Où :

- $n_{correctes}$: est le nombre d'instances qui sont bien classées par la règle R.

Une instance est bien classée par R si elle vérifie la partie condition de R et elle possède la même classe que R.

c. Autres mesures:

Quelques mesures de qualité d'une règle de classification peuvent être résumées par une matrice appelée matrice de confusion **[54]**.

Une matrice de confusion de taille 2*2 illustrée dans le ***tableau 2. 2*** est utilisée pour les bases de données bi-classes en considérant l'une des deux classes comme positive C^+ et l'autre comme négative C^-. Nous notons que les instances possédant la classe positive (resp. négatives) sont appelées instances positives (resp. négatives).

Tableau 2.2. ***La matrice de confusion***

La classe réelle / La classe prédite	$\mathbf{C^+}$	$\mathbf{C^-}$
$\mathbf{C^+}$	TP	FN
$\mathbf{C^-}$	FP	TN

Où :

- **TP :** Le nombre d'instances positives bien classées.
- **FN :** Le nombre d'instances positives prédites comme négatives (le nombre d'instances positives mal classées).
- **FP :** Le nombre d'instances négatives prédites comme positives (le nombre d'instances négatives mal classées).
- **TN :** Le nombre d'instances négatives bien classées.

5.4.3. Les mesures de performances des modèles:

Les modèles construits à partir de la base de données d'apprentissage doivent être performants c'est-à-dire ils doivent bien classer les instances de la base de test. D'où la nécessité de définir des mesures de performances pour mesurer la puissance de ces modèles. Dans les sous sections suivantes, nous présentons quelques mesures qui sont dérivées à partir de la matrice de confusion (donnée dans le ***tableau 2.2)***.

a. Précision :

La précision d'un modèle représente le pourcentage d'instances bien classées **[51]**. Elle est définie dans l'équation (2.9).

$$\boldsymbol{Precision = \frac{TP + TN}{TP + TN + FN + FP}} \qquad \textbf{(2.9)}$$

b. Erreur de mal classification :

L'erreur de mal classification représente le pourcentage d'instances mal classées **[51]**, nous la notons err. Elle est définie dans l'équation (2.10).

$$\boldsymbol{Err = \frac{FP + FN}{TP + TN + FN + FP} = 1 - Precision} \qquad \textbf{(2.10)}$$

c. Le taux de vrais positifs TPrate:

TPrate est une mesure qui détermine le pourcentage d'instances positives qui sont bien classées **[51]**. Elle est définie dans l'équation (2.11).

$$\boldsymbol{TPrate = \frac{TP}{TP + FN}} \qquad \textbf{(2.11)}$$

d. Le taux de faux positifs FPrate:

FPrate est le pourcentage d'instances négatives mal classées **[51]**. Elle est définie dans l'équation (2.12)

$$FPrate = \frac{FP}{FP + TN} \qquad (2.12)$$

e. Courbe des caractéristiques du fonctionnement du récepteur ROC :

La courbe ROC (**Receiver Operating Characteristics Curve)** est une technique de visualisation, d'organisation et de sélection des modèles en se basant sur leurs performances **[56] [56]**. Ella a longtemps été utilisée dans la détection de signal pour représenter le compromis entre le taux de succès et le taux de fausses alarmes des modèles **[57] [58].**

La courbe ROC est un graphe bidimensionnel où TPrate est représentée sur l'axe des ordonnées et FPrate est représentée sur l'axe des abscisses.

Cependant, le comportement de la courbe ROC dépend du type du modèle : discret ou probabiliste (soft-type classifiers). Un modèle discret produit la paire (FPrate, TPrate) qui correspond à un seul point dans l'espace ROC. Cependant, un modèle probabiliste produit une valeur numérique continue pour présenter la confiance d'une instance appartenant à la classe prédite, donc un seuil peut être utilisé pour produire une série de points dans l'espace ROC pour générer une courbe au lieu d'un seul point.

Exemple :

La ***figure 2.1*** illustre un exemple typique de la courbe ROC avec les points A, B, C, D, E, F et les courbes L1 et L2.

Les performances des modèles sont déduites comme suit :

- Le point A (0, 1) représente une classification parfaite.
- Le modèle M1 est meilleur que le modèle M2 si le point qui lui correspond est proche du point A que le point correspondant au M2.
- Tout modèle dont le point ROC correspondant est situé sur la diagonale est un modèle aléatoire (le point E).
- Tout modèle qui apparaît dans le triangle inférieur droit de l'espace ROC se comporte pire que la conjecture aléatoire (le point F).
- La performance d'un modèle probabiliste se calcule à l'aide de la mesure l'aire sous la courbe AUC (décrite dans la section suivante).

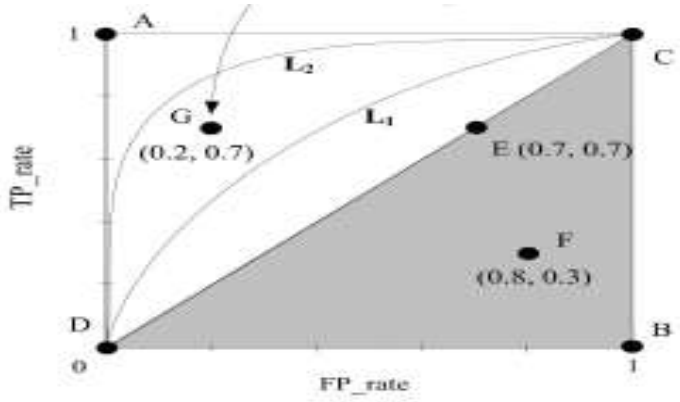

Figure 2.1. ***Exemple de la courbe ROC***

f. Aire sous la courbe AUC :

A partir de la courbe ROC, nous définissons une autre mesure appelée **Aire sous la courbe AUC** (définie dans l'équation (2.13)) afin de comparer les performances de deux modèles **[56] [57]**. Si la surface associée au modèle M1 est supérieure à celle associée au modèle M2 alors les performances de M1 sont meilleures que M2.

$$AUC = \frac{TPrate+TNrate}{2} = \frac{1+TPrate-FPrate}{2} \qquad (2.13)$$

6. Les méthodes de la classification supervisée :

Dans cette section, nous présentons quelques méthodes de la classification supervisée telles que les arbres de décision, les machines de vecteurs du support SVMs, les méthodes à plusieurs modèles et l'apprentissage sensible aux coûts.

6.1. Les arbres de décision :

Un arbre de décision est la forme la plus populaire des modèles à base de règles. Un arbre de décision est un outil d'aide à la décision et à l'exploration de données. Il permet de modéliser simplement, graphiquement et rapidement un phénomène plus ou moins complexe. Sa lisibilité, sa rapidité d'exécution et le peu d'hypothèses nécessaires à priori expliquent sa popularité actuelle **[51] [59] [60] [61].**

La construction d'un arbre de décision se fait selon l'algorithme d'apprentissage générique suivant :

Algorithme 2.1. Algorithme d'apprentissage générique d'un arbre de décision

Entrée : langage de description et l'échantillon S.
Sortie : un arbre de décision
Début
Initialiser l'arbre vide ; la racine est le nœud courant
Répéter
Si le nœud est terminal ***alors***
Affecter une classe à une feuille.
Sinon
Sélectionner un test à associer un nœud
Jusqu'à l'obtention d'un arbre de décision
Fin.

D'après l'algorithme générique, nous concluons que dans toutes les méthodes de construction d'un arbre de décision se trouvent les 03 opérateurs suivants :

1. Décider si un nœud est terminal : c'est-à-dire décider si un nœud doit être étiqueté comme une feuille.

2. Sélectionner un test à associer à un nœud.

3. Affecter une classe à une feuille : nous attribuons la classe majoritaire sauf dans le cas où nous utilisons des fonctions de coûts ou risques.

Les méthodes de construction des arbres de décision se diffèrent par les choix effectués pour ces différents opérateurs, c'est-à-dire sur le choix d'un test et le critère d'arrêt. CART et C4.5 sont les algorithmes les plus populaires de construction d'un arbre de décision.

6.1.1. L'algorithme CART :

Cet algorithme **[62]** permet d'inférer des arbres de décision binaires. Pour construire un bon arbre de décision, CART passe d'abord par la phase d'expansion, puis par la phase d'élagage. Pour cela, la base de données S utilisée est découpée en deux sous bases de données : la base d'apprentissage A et la base de test T.

Dans la phase d'expansion, la base d'apprentissage A est utilisée pour construire l'arbre de décision comme suit :

1. Décider si un nœud est terminal : en utilisant la fonction de Gini **[63]**. Un nœud p est terminal si $\mathbf{Gini(p) \leq i_o}$ ou $\mathbf{n(p) \leq n_0}$, où $\mathbf{i_0}$ et $\mathbf{n_0}$ sont des paramètres à fixer.

2. Sélectionner un test à associer à un nœud : Soit **p** une position et soit ***test*** un test. Si ce test devient l'étiquette du nœud à la position p, alors nous appelons $\mathbf{P_{gauche}}$ (respectivement $\mathbf{P_{droite}}$) la proportion d'éléments associés à **p** qui vont sur le nœud en position $\mathbf{p_1}$ (respectivement $\mathbf{p_2}$). L'étiquette ***test*** est celle qui maximise le gain définie dans l'équation (2.14).

$$\boldsymbol{Gain(p, test) = Gini(p) - \left(P_{gauche} * Gini(p_1) + P_{droite} * Gini(p_2)\right)} \quad \textbf{(2.14)}$$

3. Affecter une classe à une feuille : nous attribuons la classe majoritaire.

Dans la phase d'élagage, l'arbre t obtenu à la fin de la phase d'expansion est élagué en utilisant la base de test T. Les étapes de cette phase sont les suivantes **[64]** :

1. Construction de la suite d'arbres : nous construisons la suite d'arbres $t_1,\ldots,t_p$ où chaque arbre t_{i+1} est obtenu en élaguant l'arbre t_i comme suit :

- Pour toute position p de t_i, nous notons $\mathbf{u_p}$le sous-arbre de t_i en position p.
- Calculer $\boldsymbol{\Delta_{app}(p)}$ qui représente la variation de l''erreur apparente mesurée sur la base d'apprentissage *A* lorsque nous élaguons *t* à la position *p*. Elle est définie dans l'équation (2.15).

$$\boldsymbol{\Delta_{app}(p) = \frac{MC(p) - MC(u_p)}{N(p)}} \quad \textbf{(2.15)}$$

Où :

- ***N(p)*** est le cardinal de l'ensemble d'exemples de *A* associé à la position *p* de $\boldsymbol{t_i}$.
- ***MC(p)*** est le nombre d'éléments de *A* mal classés à la position *p* lorsque nous élaguons $\boldsymbol{t_i}$ en position *p*.
- ***MC(*** $\boldsymbol{u_p}$***)*** est le nombre d'éléments de *A* mal classés par $\boldsymbol{u_p}$ de $\boldsymbol{t_i}$.

- Remplacer le sous arbre à la position p par une feuille si cette branche montre la plus faible augmentation de l'erreur apparente.

2. Choix de l'arbre final : nous calculons pour chaque arbre t_i l'erreur apparente sur la base de test T et nous retournons celui qui minimise cette erreur.

6.1.2. L'algorithme C4.5 :

Cet algorithme permet d'inférer des arbres de décision n-aires **[63]**. Il passe aussi par la phase d'expansion et la phase d'élagage.

Dans la phase d'expansion, C4.5 passe par les étapes suivantes :

1. Décider si un nœud est terminal : Un nœud p est terminal si tous les éléments associés à ce nœud sont dans une même classe ou si aucun test n'a pas pu être sélectionné.

2. Sélectionner un test à associer à un nœud : pour une position *p*, nous choisissons le test ***test*** qui maximise le gain en utilisant la fonction entropie **[65]**. Cependant, cette fonction privilégie les attributs ayant un grand nombre de valeurs. Elle est donc pondérée par une fonction qui pénalise les tests et qui répartit les éléments en un trop grand nombre de sous-classes. Cette mesure de répartition est nommée ***SplitInfo*** et elle est définie dans l'équation (2.17).

$$SplitInfo(p, test) = \sum_{j=1}^{N} P'\left(\frac{j}{p}\right) * log\left(P'\left(\frac{j}{p}\right)\right) \quad (2.17)$$

Où :

- ***n*** : est l'arité de *test.*
- ***P'(j/p)*** : est la proportion d'éléments présents à la position *p* prenant la *j*-ème valeur de *test*.

3. **Affecter une classe à une feuille :** nous attribuons la classe majoritaire. S'il n'y a aucun exemple, nous attribuons la classe majoritaire du nœud père.

Dans la phase d'élagage**,** C4.5 utilise l'ensemble d'apprentissage pour élaguer l'arbre obtenu. Le critère d'élagage est basé sur une heuristique permettant d'estimer l'erreur réelle sur un sous-arbre donné.

6.2. Les machines de vecteurs du support SVMs:

Les méthodes d'apprentissage à base de noyaux (The Kernel-based learning) sont inspirées de la théorie statistique de l'apprentissage et les dimensions de Vapnik-Chervonenkis (VC) **[66]** telles que les machines de vecteurs du support SVMs. Ces dernières sont des méthodes d'apprentissage supervisé utilisées pour la classification dans les bases de données bi-classes dans le but de trouver un modèle qui sépare les données et qui maximise la distance entre ces deux classes. Ce modèle est linéaire et appelé **hyperplan**. Il est évident qu'il existe une multitude d'hyperplans valides mais la propriété remarquable des SVMs est que cet hyperplan doit être optimal. Donc, nous allons chercher parmi les hyperplans valides, celui qui passe au milieu des points de deux classes. Intuitivement, cela revient à chercher l'hyperplan le plus optimal. Ce dernier est celui qui maximise la marge dans le but de minimiser l'erreur de classification **[67]**. Sachant que la marge représente la distance entre l'hyperplan et les instances les plus proches. Ces dernières sont appelées les vecteurs du support.

Dans la ***figure 2.2***, l'hyperplan optimal est loin des instances, tant dés que les hyperplans valides sont proches aux instances.

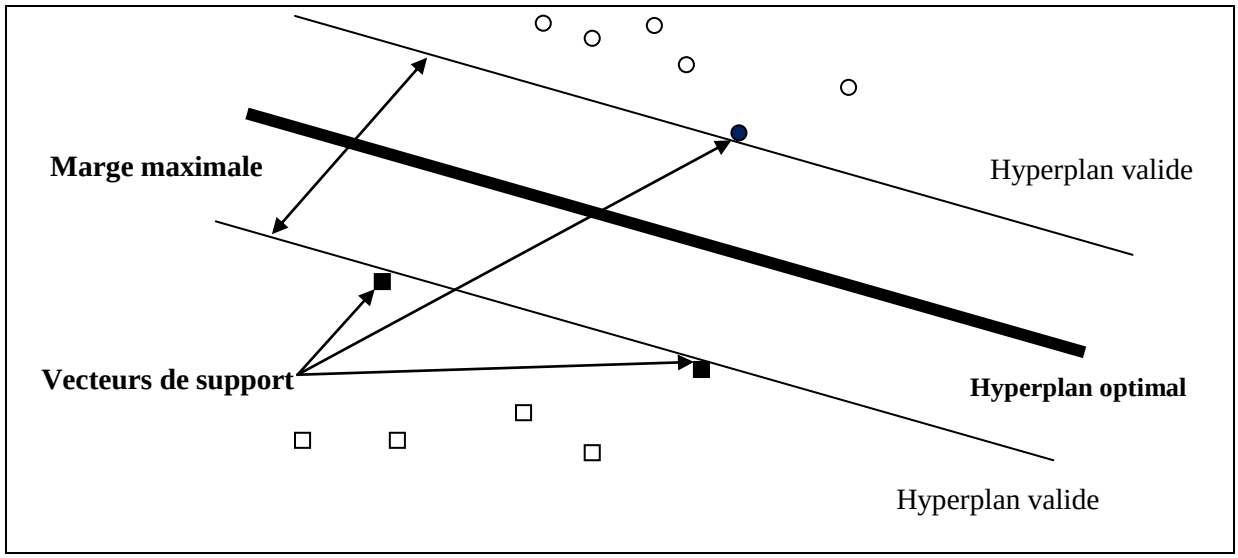

Figure 2.2. ***Représentation de la marge et les vecteurs de support***

6.3. Les méthodes à plusieurs modèles :

Le principe de ce type d'apprentissage est la combinaison d'une série de n modèles afin de créer le modèle final C*, qui est obtenu par la combinaison des modèles construits en faisant appel aux stratégies de vote. Les méthodes à plusieurs modèles ont pour but la production d'un modèle de haute précision. Elles sont divisées en deux classes : boosting et bagging. La ***figure 2.3*** illustre l'architecture générale de ces méthodes.

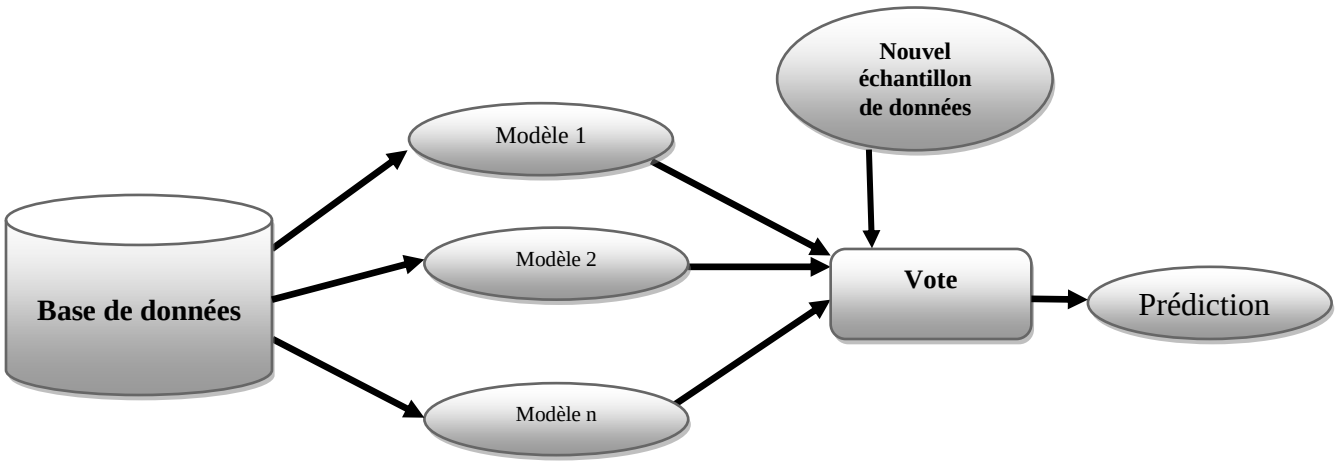

Figure 2.3. Architecture générale des méthodes à plusieurs modèles

6.3.1. Boosting :

Boosting utilise toute la base de données dans la phase d'apprentissage. A chaque itération, elle se focalise sur les instances difficiles afin de les bien classer à l'itération suivante **[67].** Les instances difficiles sont les instances mal classées durant l'itération courante. Pour cela, elle utilise la notion de poids. A l'itération 1, toutes les instances ont le même poids initial.

Après la construction du premier modèle M1, l'erreur de mal classification est calculée (qui est la somme des poids des instances mal classées). Cette erreur est utilisée pour mettre à jour le poids des instances de la base d'apprentissage qui va être utilisée dans l'itération suivante, de telle sorte que le poids d'instances mal classés augmente et le poids d'instances bien classées diminue dont le but d'avoir une erreur faible. Le dernier modèle M* est celui qui a le vote le plus élevé (sachant que le vote est en fonction de la précision). En conclusion, boosting se focalise sur les instances qui sont difficiles à classer.

L'algorithme AdaBoost **[68]** est l'algorithme le plus populaire de boosting. Ses étapes sont résumées dans l'***algorithme 2.2***.

Algorithme 2.2. AdaBoost

Entée : La base d'apprentissage S de taille N : S= $\{x_i, y_i\}$, i=1, 2… N et y_i= {+1,-1}.
Le nombre d'itérations T.

Sortie :Le modèle final H_{fin}

$H_{fin}(x) :=$ **signe**$(\sum_{t=1}^{T} \alpha t\ ht\ (x))$ ***(2.19)***

Début

T=1.

1. Toutes les instances ont le même poids : $W_i(1)=1/N$.
2. Phase d'apprentissage : appliquer l'algorithme d'apprentissage I sur la base d'apprentissage S pour obtenir le modèle h_t : $\boldsymbol{h_t := I(S, W_t)}$.
3. Evaluer la qualité du h_t en calculant le taux d'erreur (la somme des poids d'instances mal classées) :
   $\boldsymbol{\epsilon_t := \sum_i W_t(i) I(h_t(x_i) \neq y_i)}$ ***(2.20)***
   Si $\epsilon_t < 0.5$***alors*** continuer ***sinon*** T=T-1.
4. Choisir la mesure $\propto_t$ qui mesure l'importance du h_t (le poids de vote)
   $$\boldsymbol{\alpha_t := \frac{1}{2} \log\left[\frac{1-\epsilon_t}{\epsilon_t}\right]} \quad \textbf{(2.21)}$$
5. Mettre à jour la distribution W_t pour la prochaine itération. De telle sorte, que le poids des instances mal classées décroît et le poids des instances mal classées accroît.
   $\boldsymbol{W_{t+1}(i) := W_t(i) * e^{\alpha_t y_i h_t(x_i)}}$ **(2.22)**
6. Normaliser les poids :
   $$\boldsymbol{W_{t+1}(i) := W_t(i) / Z_t} \quad \textbf{(2.23)}$$
   où Z_t est un facteur de normalisation
7. T=T+1, aller à 2 si le nombre d'itérations n'est pas atteint.

Fin.

Pour prédire la classe d'une instance, chaque modèle h_t vote pour sa prédiction avec le poids α_t. La classe prédite est celle qui a le poids maximal.

AdaBoost est utilisé pour les bases de données bi-classes seulement et il n'est pas capable de traiter les hypothèses ayant un taux d'erreur supérieur à 0.5. Cependant, pour les bases de données multi-classes, il est difficile d'atteindre ce seuil. Par conséquent, Freund et Schapire proposaient deux versions de cet algorithme: AdaBoost.M1 **[68]** et AdaBoost.M2 **[69]** pour traiter les bases de données multi-classes, qui se diffèrent dans la manière du calcul de poids, du calcul de l'erreur de mal classification et de la stratégie de choix du modèle final.

6.3.2. Bagging :

Le nom Bagging (Ensachage) provient de l'abréviation de Bootstrap AGGregatING**,** comme son nom l'indique, les deux ingrédients clés de Bagging sont le bootstrap et l'agrégation **[70]**.

Un échantillon Bootstrap contenant n instances est généré à partir d'une base de données de taille m (en général n=m). Cette génération se fait à l'aide du tirage aléatoire avec remplacement. Les étapes de bagging sont résumées dans l'***algorithme 2.3***.

Algorithme 2.3. Ensachage

Entée **:** La base d'apprentissage S de taille m.
Le nombre d'itérations T.
n : la taille de l'échantillon Bootstrap.

Sortie : le modèle final H (x).

Début

Pour i :=1 à T ***faire***
Générer la base d'apprentissage D_i de taille m à l'aide du tirage aléatoire avec remise.
Phase d'apprentissage : appliquer l'algorithme d'apprentissage I sur la base d'apprentissage D_i pour obtenir le modèle h_i : $\boldsymbol{h_i := I(D_i)}$.
fait ;

$$\boldsymbol{H(x) := sign\left(\sum_{t=1}^{T} h_t(x)\right)}$$

Fin.

Le modèle final H est obtenu comme suit : étant donné une instance X, chaque modèle h_i fournit la classe prédite qui est considérée comme un vote. La classe attribuée est celle qui a un vote maximal.

6.4. L'apprentissage sensible aux coûts :

La plupart des algorithmes de classification ignorent différentes erreurs de mal classification et ils considèrent que toutes ces erreurs ont le même coût. Dans de nombreuses applications du monde réel, cette hypothèse n'est pas vraie car la différence entre les différentes erreurs de classification peut être assez importante. Par exemple, dans le

diagnostic médical, manquer un diagnostic de cancer est beaucoup plus grave que l'inverse; le patient pourrait perdre sa vie en raison du retard dans le traitement.

L'apprentissage sensible aux coûts a eu beaucoup d'attention ces dernières années pour régler ce problème où de nombreux travaux ont été effectués **[71][72][73][74][75][76]**. Ces travaux ont été divisés en deux catégories : l'apprentissage sensible aux coûts directs et le méta apprentissage sensible aux coûts (*cost sensitive Meta_Learning).*

6.4.1. L'apprentissage sensible aux coûts directs :

Dans cette catégorie, les coûts sont introduits directement dans l'algorithme d'apprentissage **[71] [77]**. Il existe différents travaux dans cette catégorie, tels que les arbres de décision sensibles aux coûts **[77]** et l'algorithme nommé classification non coûteuse avec des tests coûteux (Inexpensive Classification with Expensive Tests) ICET **[71].**

6.4.2. Le méta apprentissage sensible aux coûts :

Les méthodes de cette catégorie convertissent les algorithmes d'apprentissage non sensibles aux coûts aux algorithmes d'apprentissage sensibles aux coûts. Ces méthodes sont classées en deux catégories principales : échantillonnage et non échantillonnage. Ce classement dépend de la modification ou non de la distribution des données d'apprentissage en fonction des coûts de mal classification. Dans ce qui suit, nous détaillons ces deux catégories.

a. Echantillonnage :

Elle modifie la distribution des données d'apprentissage, puis elle appelle directement les modèles non sensibles aux coûts, comme : Costing **[78]** et CS Roulette **[79]**.

b. Non-échantillonnage :

Les méthodes de non-échantillonnage sont classées en 3 sous catégories : le ré-étiquetage, la pondération et le seuillage.

b.1. Le ré-étiquetage :

Ces méthodes changent la classe de chaque instance de la base d'apprentissage en faisant appel au critère du coût minimal. Ce critère est défini par la fixation des coûts de mal classification et les probabilités postérieures **[80]**. Le ré-étiquetage peut être divisé en deux branches:

1. Ré-étiquetage d'instances d'apprentissage (exemple : Méta cost **[81]**).
2. Ré-étiquetage d'instances de test (exemple : Cost Sensitive Classifier CSC **[51]**)

b.2. La pondération:

Cette méthode assigne un poids à chaque instance en fonction de sa classe proportionnellement aux coûts de mal classification **[76]**. Elle est utilisée à chaque fois que les modèles non sensibles aux coûts peuvent accepter la pondération des instances.

b.3. Le seuillage :

Le seuillage est utilisé pour les modèles qui peuvent produire des probabilités d'estimation sur les bases d'apprentissage et du test **[73]**. Il consiste à sélectionner la meilleure probabilité d'estimation qui minimise le coût total de mauvaise classification sur les instances d'apprentissage. Cette probabilité est utilisée comme seuil pour prédire la classe des instances de test: une instance de test avec une probabilité prédite supérieure ou égale à ce seuil est prédite comme positive; sinon comme négative.

6.5. Les approches basées sur les algorithmes génétiques :

Un AG évolue un espace de recherche composé de plusieurs individus qui représentent un ensemble de règles de classification. Il existe deux stratégies de représentation de chaque individu : l'approche de Michigan et l'approche de Pittsburgh.

Dans l'approche de Michigan **[82]**, chaque individu représente une seule règle seulement et la population représente l'ensemble de ces règles. Par contre, dans l'approche de Pittsburgh **[83]**, chaque individu représente un ensemble de règles. Dans ce qui suit, nous présentons quelques approches de classification basées sur les algorithmes génétiques.

6.5.1. GA Batch Incremental concept Learner GABIL:

GABIL utilise l'approche de Pittsburgh pour extraire les règles de classification **[51]**. Chaque règle est représentée par GABIL (***vu dans la section 5.4.1***) et elle est codée par le codage binaire. La fonction d'évaluation est en fonction du nombre d'instances bien classées au carré. Afin de donner l'algorithme de fonctionnement de cette approche, nous définissons d'abord le mode batch et le mode incrémental.

Les systèmes d'apprentissage traditionnels se diffèrent de la façon dont les instances de la base d'apprentissage sont présentées **[51]**. Dans le mode batch, toute la base d'apprentissage est présentée au système à la fois. Cependant, dans le mode incrémental, une ou quelques instances sont présentées au système.

GABIL utilise les deux modes dans la phase d'apprentissage. Ses étapes sont résumées dans l'***algorithme 2.4***.

Algorithme 2. 4. ***L'approche GABIL***

***Entrée* :** l'ensemble d'apprentissage S.
***Sortie* :** l'ensemble de règles E.
Début
Initialement, l'ensemble S' est constitué d'une seule instance.
S' est utilisé pour chercher l'ensemble de règles E aussi parfait que possible.
Pour chaque instance i de la base S ***faire***
Utiliser l'ensemble de règles E pour prédire la classe de l'instance i.
Si la prédiction est correcte ***alors***
Ajouter l'instance i à l'ensemble S' : $S' = S' \cup \{i\}$.
Ne pas changer E.
Sinon
Appeler l'AG en mode batch pour chercher le nouvel ensemble de règles E en utilisant l'ensemble $S' \cup \{i\}$.
fait ;
Fin.

6.5.2. Le système de modèles à base d'algorithmes génétiques GAssist:

GAssist (**Genetic Algorithms based claSSIfier system)** utilise l'approche de Pittsburgh pour extraire les règles de classification **[84]**. Il utilise la représentation de GABIL (***vue dans la section 5.4.1***) pour les attributs nominaux et la discrétisation adaptative des intervalles (Adaptive Discretization Intervals) ADI **[85]** pour les attributs réels. Le codage binaire est utilisé pour coder les règles. L'évaluation de chaque individu se fait à l'aide de l'utilisation d'une fonction d'adaptation spéciale basée sur le principe de **MDL** (**M**inimum **D**escription **L**ength) **[86]** qui mélange la précision et la complexité d'un individu. La formule de MDL est définie dans l'équation (2. 23).

$$\boldsymbol{MDL = W * TL + EL} \qquad \textbf{\textit{(2.23)}}$$

Où :

- **TL** : est la complexité de l'ensemble de règles qui prend en considération le nombre de règles et le nombre d'attributs importants dans chaque règle.
- **EL** : mesure l'erreur d'un individu par rapport à l'ensemble d'apprentissage.
- **W** : est une constante qui exprime une relation entre TL et EL.

GAssist utilise également un schéma de fenêtrage appelé apprentissage incrémental avec strates alternées (Incremental Learning with Alternating Strata) ILAS **[87].** Dans ce schéma, la base d'apprentissage est divisée en S sous bases d'apprentissage de même taille. Chaque itération de l'algorithme génétique utilise une sous base de données différente en utilisant round robin.

La longueur variable des individus pose quelques problèmes importants **[88]**. Donc, il faut contrôler la taille des individus qui sont en cours d'évolution. Ce contrôle est assuré par

l'utilisation de deux opérateurs **[89]**: suppression de règle et description minimale de la longueur basée sur la fonction d'adaptation (Minimum description length-based fitness function).

6.5.3. Règles de décision hiérarchiques HIDER :

L'algorithme Hider (**Hierarchical Decision Rules)** produit un ensemble de règles hiérarchiques en utilisant les algorithmes génétiques et l'approche de Michigan. Il utilise le codage hybride c'est-à-dire le codage binaire pour les attributs discrets et le codage réel pour les attributs continus. Chaque attribut discret (contenant k valeurs possibles) est représenté par k bits et chaque attribut continu est représenté par deux valeurs réelles qui représentent la borne inférieure et la borne supérieure de cet attribut **[87]**. La ***figure 2.4*** illustre cette représentation.

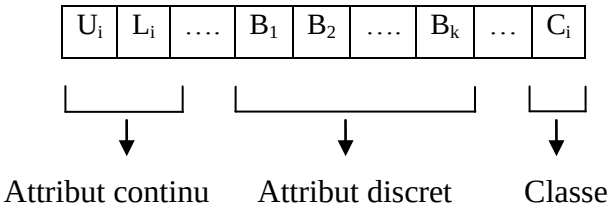

Figure 2.4. ***Représentation de codage hybride***

L'algorithme d'extraction est un algorithme de recouvrement séquentiel **[59]** avec les algorithmes génétiques. Il choisit le meilleur individu à chaque lancement de l'algorithme génétique et la transforme en une règle. Puis, il supprime les instances qui sont couvertes par cette règle. De cette façon, la base d'apprentissage se réduit à chaque itération. Le critère d'arrêt pourrait être atteint lorsque la base d'apprentissage devient vide.

L'évaluation de chaque individu de la population utilise la fonction d'évaluation donnée dans l'équation (2. 24).

$$\boldsymbol{f(I) = N - CE(I) + G(I) + Couverture(I)} \qquad \textbf{(2.24)}$$

Où :

- **I** : est un individu.
- **N** : est le nombre d'instances qui doivent être traitées.
- **CE** : est l'erreur de la classe qui est égal aux nombre d'instances mal classées par I.
- **G** : est le nombre d'instances bien classées par I.
- **Couverture**: est la proportion d'éléments de l'espace de recherche couverts par I.

7. Conclusion :

Dans ce chapitre, nous avons montré l'importance de Data Mining en présentant ses domaines d'applications ainsi que ses différentes tâches en focalisant sur la tâche de la classification où nous avons exposé les différentes méthodes d'extraction des modèles.

Chapitre 3 :

LE PROBLEME DE LA CLASSIFICATION DANS LES BASES DE DONNEES DESEQUILIBREES

Chapitre 3 Le problème de la classification dans les bases de données déséquilibrées

1. Introduction :

Les approches classiques d'extraction des règles de classification à partir des bases de données déséquilibrées ignorent les instances les moins fréquentes. Pour cette raison, la classification a été divisée en deux types; la classification complète et la classification partielle. Dans ce chapitre, nous allons présenter le problème de la classification partielle, les mesures de performances des modèles appropriées à ce type de classification et les différentes techniques et méthodes proposées pour traiter ce problème.

2. Présentation du problème de la classification partielle :

Dans les bases de données déséquilibrées bi-classes, le nombre d'instances d'une classe est trop élevé par rapport à l'autre classe. Par conséquent, la première classe est appelée ***la classe majoritaire*** et la deuxième classe est appelée ***la classe minoritaire***. Donc, une base de données déséquilibrée contient deux types d'instances :

- ***Les instances majoritaires :*** sont les instances les plus fréquentes et qui appartiennent à la classe majoritaire.
- ***Les instances minoritaires :*** sont les instances les moins fréquentes et qui appartiennent à la classe minoritaire.

La distribution des instances minoritaires et majoritaires dans les bases de données déséquilibrées bi-classes est mesurée par le degré du déséquilibre (Imbalanced Ratio) IR **[90]**, qui est égal à la fraction entre les instances majoritaires et minoritaires. Il est défini dans l'équation (3.1).

$$IR = \frac{Nombre\ d'instances\ majoritaires}{Nombre\ d'instances\ minoritaires} \qquad (3.1)$$

Selon IR, les bases de données déséquilibrées sont divisées en trois classes **[17]** :

1. Les bases de données faiblement déséquilibrées : si IR est entre 1.5 et 3.
2. Les bases de données moyennement déséquilibrées : si IR est entre 3 et 9.
3. Les bases de données fortement déséquilibrées : si IR est supérieur à 9.

3. Les domaines d'application :

Les bases de données déséquilibrées apparaissent dans plusieurs domaines tels que :

3.1. Détection des fraudes :

Par exemples, la fraude par carte de crédit et la fraude cellulaire sont des problèmes coûteux pour les organisations commerciales. Aux États-Unis, la fraude cellulaire coûte aux industries de télécommunication des centaines de millions de dollars par an **[91] [92]**. Une méthode pour détecter la fraude consiste à vérifier les changements suspects dans le

comportement des utilisateurs. Le comportement d'achat de quelqu'un qui vole une carte de crédit est probablement différent de la propriétaire d'origine. Les entreprises tentent de détecter les fraudes en analysant les différents modèles de consommation dans les bases de données de transaction. Cependant, dans ces dernières, il y a un plus grand nombre d'utilisateurs légitimes que frauduleux.

3.2. Diagnostic médical :

Les bases de données cliniques stockent de grandes quantités d'informations sur les patients. Les techniques de Data Mining appliquées sur ces bases de données tentent de découvrir les relations et les tendances entre les données cliniques et pathologiques afin de comprendre l'évolution et les caractéristiques de certaines maladies. Les connaissances découvertes peuvent être utilisées pour le diagnostic. Dans les bases de données cliniques, les cas de maladies sont plus rares par rapport à la population normale **[93]**.

3.3. Détection d'intrusions:

Les systèmes informatiques basés sur le réseau (Network-based computer systems) jouent de plus en plus un rôle vital dans les sociétés modernes. Les attaques contre les systèmes informatiques et les réseaux sont en croissance. Différents types d'attaques réseau existent ; certaines sont nombreuses et d'autres sont rares. Par exemple, les données du concours KDD-CUP'99 contiennent quatre catégories d'attaques réseau : déni de service (DoS), la surveillance (la sonde, Probe), Root-to-local (R2L) et User-to-Root(U2R). Parmi ces 4 types d'attaques, l'U2R et R2L sont intrinsèquement rares **[2]**.

3.4. Gestion des risques :

Chaque année, l'industrie des télécommunications subit plusieurs milliards de dollars de dettes irrécouvrables. Par conséquent, le contrôle irrécouvrable est un problème important dans l'industrie. Une solution consiste à utiliser de grandes quantités de données historiques pour construire des modèles qui servent à évaluer les risques par client ou par transaction afin de supporter la gestion des risques qui réduit le niveau de créances irrécouvrables. Cependant, dans un ensemble de données, le non-paiement des clients comprend quelques pourcent de la population **[94]**.

3.5. Détection des déversements de pétrole à partir des images radar de la surface de l'océan:

Seulement environ 10% des déversements de pétrole proviennent de sources naturelles, comme les fuites de fonds marins. Beaucoup plus répandue est la pollution causée intentionnellement par les navires qui veulent éliminer les résidus de l'huile dans leurs réservoirs. Bien que les satellites produisent des images continuellement, les images contenant

des déversements de pétrole sont beaucoup moins nombreuses que celles qui n'ont pas de déversements **[95]**.

4. Les mesures de performances des modèles:

Les mesures de performances de modèles classiques utilisées dans la classification complète ne sont pas appropriées pour la classification partielle. En effet, elles ont une forte tendance vers la classe majoritaire et elles sont aussi sensibles au déséquilibre entre les classes **[96][97][98][99]**. Par exemple, l'exactitude ***(vue dans le chapitre 2, section 5.4.2)*** est une mesure qui n'est pas appropriée à la classification partielle **[100]**.

Prenons un exemple d'une base de données déséquilibrée contenant 1% d'instances minoritaires et 99% d'instances majoritaires. La classification de toutes les instances comme majoritaires donne une exactitude de 99%, mais elle classe mal 1% d'instances minoritaires qui peut conduire à un coût énorme. Par conséquent, 99% d'exactitude pourrait être une catastrophe pour un diagnostic médical.

Par conséquent, de nouvelles mesures appropriées pour la classification partielle ont été proposées. Spécifiquement, quelques mesures ont été extraites directement à partir de la matrice de confusion ***(voir chapitre 2, tableau 2.2)*** afin de mesurer les performances de classes majoritaires et de classes minoritaires indépendamment. Cependant, la comparaison des performances des différents modèles est difficile, d'où la nécessité de définir des mesures combinées. Dans les deux sous sections suivantes, nous présentons quelques mesures directes et combinées. Nous notons que la classe minoritaire est considérée comme la classe positive et la classe majoritaire est considérée comme la classe négative. Nous nous rappelons que les mesures TP, FP, TN et FN sont définies dans ***la section 5.4.2.***

4.1. Les mesures directes :

4.1.1. La précision:

Est une mesure d'exactitude qui calcule le pourcentage d'instances minoritaires correctement classées par rapport à toutes les instances dont la classe prédite est minoritaire **[101]**.

$$\boldsymbol{Précision = \frac{TP}{TP + FP}} \qquad \textit{(3.2)}$$

4.1.2. Le taux de vrais positifs (TPrate):

Est une mesure de complétude qui détermine le pourcentage d'instances minoritaires qui sont bien classées **[101]**. Dans la littérature, cette métrique est connue sous plusieurs noms : la sensitivité, le rappel, l'exactitude positive (positive accuracy notée acc^{+}) **[102]**. Cette mesure est définie dans l'équation (3.3).

$$TPrate = \frac{TP}{TP + FN} \quad (3.3)$$

4.1.3. Le taux de vrais négatifs (TNrate):

Est le pourcentage d'instances majoritaires qui sont bien classées. Cette mesure est connue aussi sous les noms : spécificité ou exactitude négative (negative accuracy, notée acc^-). Cette mesure est définie dans l'équation (3.4).

$$TNrate = \frac{TN}{TN + FP} \quad (3.4)$$

4.1.4. Le taux de faux négatifs (FNrate):

Est le pourcentage d'instances minoritaires qui sont mal classées. Elle est définie dans l'équation (3.5).

$$FNrate = \frac{FN}{FN + TP} \quad (3.5)$$

4.1.5. Le taux des faux positifs (FPrate):

Est le pourcentage d'instances majoritaires qui sont mal classées. Elle est définie dans l'équation (3.6)

$$FPrate = \frac{FP}{FP + TN} \quad (3.6)$$

4.2. Les mesures combinées :

En examinant séparément les mesures précédentes, il est difficile de décider quel modèle est meilleur, car l'un d'entre eux gagnera habituellement sur l'une des mesures et l'autre sur une autre mesure. Par conséquent, il est nécessaire de définir des mesures qui combinent les mesures directes. Parmi ces mesures, nous avons G-Mean, F-Mesure, Caractéristiques de fonctionnement du récepteur ROC et l'aire sous la courbe AUC.

4.2.1. G-Mean:

G-Mean **[102]** indique un équilibre de performance entre les classes majoritaires et minoritaires. Une mauvaise performance dans la prédiction des instances minoritaires conduira à une faible valeur de G-Mean, même si les instances majoritaires sont correctement classées par le modèle **[103]**. Elle a été utilisée par plusieurs chercheurs pour évaluer les performances des modèles dans le cas des bases de données déséquilibrées **[103] [104]**. Cette métrique prend en compte TPrate et TNrate simultanément. Elle est définie dans l'équation (3.7).

$$\textbf{G-Mean} = \sqrt{TPrate * TNrate} \quad (3.7)$$

La propriété importante de G-Mean est qu'elle est indépendante de la distribution des instances entre les différentes classes **[105]**. Donc, elle est adéquate pour le problème de la classification partielle.

4.2.2. F-Mesure:

Cette métrique est la moyenne harmonique de la précision et du TPrate (équations (3.2) et (3.3)) pour mesurer l'efficacité d'un modèle **[56]**.

$$F - Mesure = \frac{2 * TPrate * Précision}{TPrate + Precision} \quad (3.8)$$

4.2.3. Autre mesures

Caractéristiques de fonctionnement du récepteur ROC et Aire sous la courbe AUC ***(vues dans le chapitre 2, section 5.4.3)*** sont aussi utilisées par la classification partielle.

5. Les méthodes de la classification partielle:

Les différentes solutions et techniques utilisées pour traiter le problème de la classification partielle peuvent être divisées en trois niveaux **[19]** : le niveau donnée, le niveau algorithme et le niveau hybride. Dans ce qui suit, nous allons expliquer le principe de chaque niveau ainsi que ses méthodes. Avant de présenter ces méthodes, nous donnons d'abord la définition formelle de la classification partielle:

La classification partielle **[15]**, également appelée ***découverte de pépites*** **[16]**, cherche à trouver des modèles qui représentent une description d'une classe particulière. Cette tâche de Data Mining est particulièrement pertinente lorsque certaines classes d'une base de données sont minoritaires, c'est-à-dire celles avec peu d'instances représentatives dans la base de données, car, dans ces cas, une globale exactitude de classification peut être obtenue même lorsque les instances minoritaires sont mal classées par le modèle induit.

5.1. Niveau données :

Les approches de ce niveau sont des approches externes. Elles altèrent la distribution des instances dans les bases de données déséquilibrées afin de diminuer le degré du déséquilibre **[3]** par l'application des méthodes d'échantillonnage.

Il est prouvé que ces méthodes fournissent une solution positive au problème de la classification partielle **[3].** Cependant, son principal inconvénient est qu'elles sont indépendantes du modèle utilisé. Ces méthodes sont divisées en trois groupes : les méthodes du sous-échantillonnage, les méthodes du sur-échantillonnage et les méthodes hybrides.

5.1.1. Les méthodes du sous-échantillonnage :

Ce type d'échantillonnage sert à équilibrer les bases de données déséquilibrées par la suppression de quelques instances majoritaires. Différentes méthodes du sous-

échantillonnage existent. Elles se diffèrent de la manière de sélection d'instances majoritaires à supprimer.

a. Le sous-échantillonnage aléatoire RUS :

RUS (Random UnderSampling) diminue la taille de la base d'apprentissage par la suppression de quelques instances majoritaires choisies aléatoirement **[4] [106]**. Cette méthode peut potentiellement entraver l'apprentissage **[107] [108] [109]**. Les instances majoritaires supprimées peuvent faire en sorte que le modèle ignore des concepts importants relatifs à la classe majoritaire.

b. Le sous-échantillonnage informé :

Ce type d'échantillonnage a été proposé pour éviter la perte d'information causée par le sous-échantillonnage aléatoire **[110]**. Parmi les algorithmes de ce type d'échantillonnage, nous avons les algorithmes : EasyEnsemble, BalanceCascade et sous-échantillonnage informé avec KNN (K-plus proches voisins).

b.1. EasyEnsemble:

EasyEnsemble a pour objectif la bonne exploitation d'instances majoritaires ignorées par le sous-échantillonnage aléatoire. Il divise la base d'apprentissage en deux ensembles : l'ensemble minoritaire (P) de taille p et l'ensemble majoritaire (N) de taille n, puis il applique l'échantillonnage aléatoire avec remplacement sur N, T fois pour produire les sous ensembles N_1, N_2, …, N_T de taille p. Ensuite, pour chaque ensemble N_i ($1 \leq i \leq T$) un modèle H_i est généré à partir de l'ensemble N_i et tout l'ensemble P. Enfin, tous les modèles générés sont combinés pour former le modèle final H **[110]**.

AdaBoost est utilisé pour générer tous les modèles H_i qui contiennent les concepts de toutes les instances minoritaires et majoritaires.

Les étapes de génération des modèles sont indépendantes ; l'échantillonnage utilise toujours l'ensemble initial d'instances majoritaires N et par conséquent, il augmente la redondance des informations d'instances majoritaires.

Cet algorithme est considéré comme un algorithme d'apprentissage non supervisé qui explore les instances majoritaires par l'échantillonnage aléatoire avec remplacement.

b.2. BalanceCascade:

BalanceCascade est un algorithme d'apprentissage supervisé qui génère un ensemble de modèles pour sélectionner les instances majoritaires qui doivent être supprimées.

Pour une base d'apprentissage composée de deux ensembles : l'ensemble minoritaire (P) de taille p et l'ensemble majoritaire (N) de taille n, BalanceCascade construit à chaque itération i le modèle H_i à partir de tout l'ensemble P et le sous ensemble E de N choisi

aléatoirement, avec |E|=p. Puis, il met à jour N par la suppression de toutes les instances majoritaires qui sont bien classées par H_i **[110].**

Cet algorithme explore les instances majoritaires d'une manière supervisée car l'ensemble majoritaire est mis à jour après chaque itération, c'est à dire après la construction de chaque modèle.

b.3. Le sous-échantillonnage informé avec KNN:

Cette technique est basée sur les caractéristiques de distribution de données **[106]** en utilisant l'algorithme KNN **[111]**. Les trois méthodes suivantes de cette technique ont été proposées :

- ***La méthode NearMiss-1*** sélectionne les instances majoritaires qui sont proches de quelques instances minoritaires. L'***algorithme 3.1*** résume les étapes de cette méthode :

Algorithme 3.1. La méthode NearMiss-1

Entrée : la base de données déséquilibrée composée de l'ensemble d'instances minoritaires S_{min} et l'ensemble d'instances majoritaires S_{maj}.

Sortie : la base d'apprentissage échantillonnée.

Début

Pour chaque instance majoritaire x_i de l'ensemble S_{maj} ***faire***

Pour chaque instance minoritaire x_j ***faire***

Calculer la distance d_{ij} entre x_i et x_j.

fait;

Déterminer les 3-plus proches voisins x_k représentant les instances minoritaires de x_i

Calculer la distance moyenne d_i des distances de ces 3-plus proches voisins.

fait;

Sélectionner les instances majoritaires qui ont les distances moyennes les plus petites.

Fin.

- ***La méthode NearMiss-2*** sélectionne les instances majoritaires qui sont proches de toutes les instances minoritaires. La sélection est basée sur la distance moyenne entre chaque instance majoritaire et ses 03 instances minoritaires les plus éloignées.
- ***La méthode NearMiss-3*** sélectionne un nombre donné de plus proches instances majoritaires pour chaque instance minoritaire. Cette méthode garantit que chaque instance minoritaire est entourée par quelques instances majoritaires.

c. Le sous-échantillonnage avec les techniques de nettoyage des données :

Les techniques de nettoyage de données ont été appliquées sur les bases de données pour éliminer le chevauchement telles que les liens de Tomek, la règle condensée la plus proche, la règle de nettoyage du voisinage et la méthode d'échantillonnage unilatéral. Dans ce qui suit, nous expliquons chacune de ces méthodes.

c.1. Les liens de Tomek:

La méthode liens de Tomek (Tomek links) peut être utilisée comme une méthode de sous-échantillonnage pour supprimer les instances majoritaires bruitées et les instances majoritaires qui sont proches de la frontière (sont celles qui séparent les instances minoritaires et majoritaires et qui causent le chevauchement entre les classes) **[112]**.

Un lien de Tomek est défini comme une paire (x_i, x_j) où x_i et x_j sont les plus proches voisins appartenant à deux classes opposées, c'est-à-dire si $d(x_i,x_j)$ est la distance entre x_i et x_j et il n'existe pas un exemple x_k tel que $d(x_i, x_k)<d(x_i, x_j)$ ou $d(x_j, x_k)<d(x_i, x_j)$.

La base d'apprentissage obtenue après la suppression de liens de Tomek est organisée en un ensemble de clusters. Les ***figures 3.1 et 3.2*** illustrent l'identification de liens de Tomek et l'état de la base de données après la suppression de ces liens. Sachant que les petits cercles représentent les instances majoritaires et les étoiles représentent les instances minoritaires.

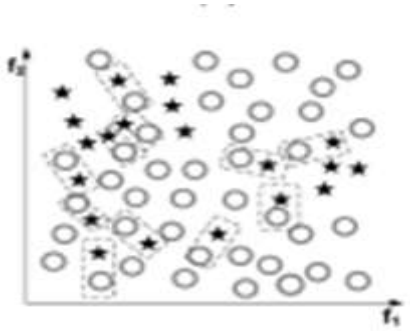

Figure 3.1. ***Identification des liens de Tomek***

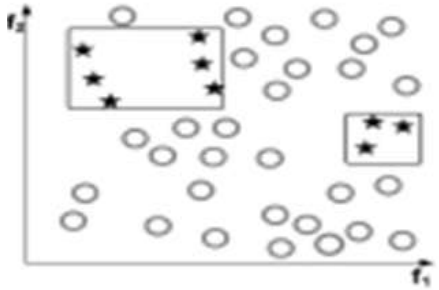

Figure 3.2. ***La base de données après la suppression des liens de Tomek***

Remarque : la méthode Liens de Tomek peut être utilisée aussi comme une méthode de nettoyage de données pour supprimer les instances minoritaires et majoritaires.

c.2. La règle condensée la plus proche CNN:

CNN (Condensed Nearest Neighbor rule) est un algorithme de réduction d'instances proposé par HART **[113].** Il a été utilisé comme une méthode de sous-échantillonnage pour supprimer les instances majoritaires redondantes (peuvent être déduites à partir d'autres instances). A partir de la base d'apprentissage initial E, nous construisons le sous ensemble consistant E', c'est-à-dire E' contient les instances qui classent correctement toutes les instances de E en utilisant l'algorithme 1-NN. Cette suppression se fait selon les étapes décrites dans l'***algorithme 3.2***.

Algorithme 3.2. La règle condensée la plus proche CNN

Entrée : la base d'apprentissage E.
Sortie : la base d'apprentissage échantillonnée E'.
Début
Copier la première instance majoritaire *x* et toutes les instances minoritaires de la base d'apprentissage E dans la sous base de données E'et 1-NN.
Tant qu'il y a des instances mal classées dans E ***faire***
Classer l'instance *x'* de E en utilisant les instances de E'.
Si x' est mal classée ***alors*** la rajouter dans E'.
Fait;
Fin.

CNN est sensible au bruit. Cependant, les instances bruitées sont susceptibles d'être mal classées **[114]**, de plus elles vont mal classer les instances de la base de test **[114] [115].**

c.3. La règle de nettoyage du voisinage NCL :

NCL (Neighbourhood Cleaning Rule) est une technique de sous-échantillonnage introduite par LAURIKKALA pour équilibrer les bases de données déséquilibrées en effectuant des réductions de données **[116]**. Son principal avantage est qu'elle prend en considération la qualité des données en mettant l'accent sur le nettoyage de données plus que la réduction.

NCL réduit la taille d'une base de données déséquilibrée par la suppression des instances majoritaires bruitées en utilisant l'algorithme Edited Nearest Neighbor ENN.

ENN est un algorithme de réduction d'instances développé par Wilson **[117].** Il sert à supprimer toutes les instances dont la classe se diffère au moins deux fois de la classe de ses 3-plus proches voisins.

c.4. La méthode d'échantillonnage unilatéral OSS:

OSS (One-Sided Sampling) est le résultat d'application de CNN suivi de l'application des liens de Tomek **[17]**. L'algorithme CNN est appliqué pour supprimer les instances majoritaires qui sont redondantes, tandis que les liens de Tomek servent à supprimer les instances majoritaires bruitées et les instances de la frontière.

d. Le sous-échantillonnage basé sur le clustering SBC :

Le principe de l'approche SBC (under-Sampling Based on Clustering) consiste à équilibrer la base d'apprentissage en utilisant le clustering **[118].** Ce dernier sert à déterminer le nombre d'instances que doit contenir la base de données équilibrée finale.

Afin de présenter les étapes de SBC, nous donnons d'abord les notations suivantes
Soient :

- **N** : la taille de la base d'apprentissage.
- **$Size_{MA}$** : le nombre d'instances majoritaires.

- **$Size_{MI}$** : le nombre d'instances minoritaires.
- **K** : le nombre de clusters (la valeur de K dépend de l'étude expérimentale).
- **$Size^{n}_{MA}$** : le nombre d'instances majoritaires dans le nième cluster.
- **$Size^{n}_{MI}$** : le nombre d'instances minoritaires dans le nième cluster.
- **M_n** : le rapport entre **$Size^{n}_{MA}$** et **$Size^{n}_{Mi}$** du nième cluster.
- **M** :Supposons que le rapport entre $Size_{MA}$ et $Size_{MI}$ est défini sur M: 1 (M≥1).
- $\boldsymbol{M * Size_{MI}}$ **:** est le nombre total d'instances majoritaires sélectionnées que nous supposons avoir dans la base d'apprentissage finale.
- $\boldsymbol{SSize^{n}_{MA}}$ **:** le nombre d'instances majoritaires sélectionnées dans le nième cluster.

$$\boldsymbol{SSize^{n}_{MA} = (M * Size_{MI}) * \frac{M_n}{\sum_{i=1}^{K} M_i}} \qquad \textit{(3.9)}$$

Les étapes de l'approche SBC sont résumées dans l'***algorithme 3.3***.

Algorithme 3.3. L'approche SBC

Entrée : la base d'apprentissage déséquilibrée D et le nombre de clusters K.
Sortie : la base d'apprentissage équilibrée E.
Début
Diviser la base de données D en K clusters.
Calculer le nombre total d'instances majoritaires que nous supposons avoir dans la base d'apprentissage finale : $\boldsymbol{M * Size_{MI}}$
Pour chaque cluster n ***faire***
Calculer M_n.
Calculer le nombre d'instances majoritaires $\boldsymbol{SSize^{n}_{MA}}$ à sélectionner.
Sélectionner aléatoirement $\boldsymbol{SSize^{n}_{MA}}$ instances majoritaires et les rajouter dans E.
fait ;
Fin.

e. Le sous-échantillonnage avec les algorithmes évolutionnaires :

Les algorithmes évolutionnaires ont été aussi utilisés comme méthodes de sous-échantillonnage pour équilibrer la base de données initiale. Parmi ces algorithmes, nous présentons l'utilisation des algorithmes génétiques et les colonies de fourmis.

e.1. Le sous échantillonnage évolutionnaire :

Le sous échantillonnage évolutionnaire (**Evolutionary Undersampling) EUS** est le résultat d'utilisation des algorithmes évolutionnaires avec la méthode de sélection d'instances **[122]** au problème de la classification dans les bases de données déséquilibrées **[19]**. Afin d'expliquer le principe de cette approche, nous donnons d'abord la représentation d'un chromosome et la fonction d'évaluation utilisée.

e.1.1. La représentation d'un chromosome:

Soient :

- TR : la base d'apprentissage contenant N instances.
- S : une sous base de données de TR (S⊆ TR).

L'espace de recherche est constitué par tous les sous ensembles S de TR. Donc, chaque chromosome contient N gènes et il représente le sous ensemble S. Le codage utilisé est binaire où chaque gène prend la valeur 1 ou 0. Si le gène i contient la valeur 1, alors l'instance i de TR est présente dans S **[119]**.

Exemple 4.1 :

Soit la base de données TR contenant 13 instances et deux classes A et B. les sept premières instances appartenant à la classe A et les six dernières représentent la classe B. Soit le sous ensemble S contenant les instances 2, 3, 5, 6, 8 et 12 alors le chromosome a la représentation donnée dans la ***figure 3.3***.

0	1	1	0	1	1	0	1	0	0	0	1	0

Figure 3.3. ***La représentation d'un chromosome dans l'approche EUS***

e.1.2. La fonction d'évaluation:

Cette fonction combine deux valeurs : le taux de classification associé à S et le pourcentage de réduction d'instances de S relativement à TR **[120]**. Cette fonction a pour objectif la minimisation du nombre d'instances et la maximisation du taux de la classification. Le meilleur chromosome est celui qui maximise la fonction d'évaluation donnée dans l'équation (3.10).

$$\boldsymbol{fitness(S) = \alpha * Taux\_Class + (1 - \alpha) * Taux\_Red} \qquad \textit{(3.10)}$$

Où :

- ***Taux_Class*** : est le pourcentage d'instances bien classées de la base de données TR en utilisant S seulement pour trouver les voisins les plus proches. La valeur de cette mesure est calculée par KNN avec K=1, c'est-à-dire pour chaque instance y appartenant à S, le voisin le plus proche est cherché parmi ceux de l'ensemble S\{y}.
- ***Taux_Red*** : est le pourcentage de réduction. Il est calculé en fonction du nombre d'instances présentes dans S.

$$\boldsymbol{Taux\_Red = 100 * \frac{|TR| - |S|}{|TR|}} \qquad \textit{(3.11)}$$

L'approche utilisée est l'approche de Michigan **[82]** où deux versions de l'algorithme génétique ont été utilisées : CHC (**C**ross generational elitist selection, **H**eterogeneous recombination and **C**ataclysmic mutation) **[121]** et l'algorithme génétique intelligent (Intelligent Genetic Algorithm) IGA **[122]**.

e.1.3. Les modèles de l'approche EUS :

Il existe plusieurs modèles de l'approche EUS. Ils dépendent des objectifs que l'approche EUS veut atteindre. Ces objectifs ont été divisés en deux catégories :

Dans la première catégorie, il y'a les deux buts suivants :

1. Equilibrer la base de données sans perte effective de l'exactitude de la classification. Dans ce cas, EUS est appelée ***sous-échantillonnage évolutionnaire d'équilibrage (Evolutionary balancing undersampling EBUS).***

2. Obtenir une puissance optimale de la classification sans prendre en considération l'équilibrage de la base de données. Les modèles de EUS qui suivent cette tendance sont appelés ***sous-échantillonnage évolutionnaire guidé par les mesures de classification (Evolutionary Undersampling guided by Classification Measures EUSCM).***

Dans la deuxième catégorie, nous distinguons deux cas, selon le type d'instances sélectionnées par EUS:

1. La sélection majoritaire MS (Majority Selection) : si la sélection concerne les instances majoritaires seulement.

2. La sélection globale GS (Global Selection): si les instances majoritaires et minoritaires sont concernées par la sélection.

Les deux catégories précédentes produisent quatre sous-groupes de EUS. Cependant, deux méthodes sont incluses dans chaque sous-groupe. Elles diffèrent par la mesure utilisée dans la fonction d'évaluation (G-Mean ou AUC). Par conséquent, EUS contient les huit modèles résumés dans la ***figure 3.4*** et qui vont être décrits dans les sous sections suivantes.

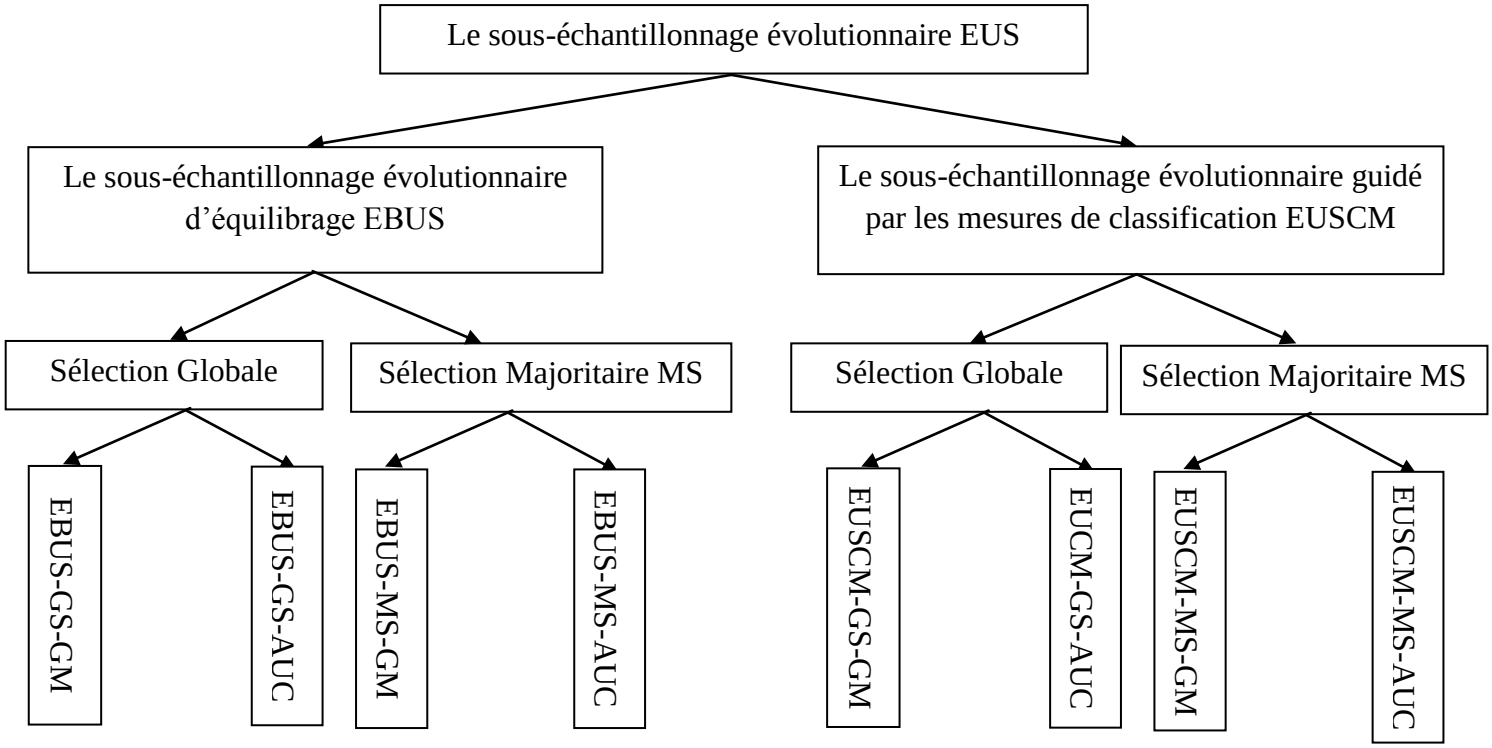

Figure 3.4. ***Taxonomie de l'approche EUS [19]***

A. Le sous-échantillonnage évolutionnaire d'équilibrage EBUS :

Ce sous-groupe contient quatre modèles: EBUS-GS-GM, EBUS-MS-GM, EBUS-GS-AUC et EBUS-MS-AUC.

A.1. EBUS-GS-GM:

Ce modèle a été utilisé dans **[123]**, ses objectifs sont les suivants:

- La suppression de toute sorte d'instances.
- L'identification des instances minoritaires qui influencent négativement sur la classification.
- L'obtention d'une réduction maximale sur les instances minoritaires.

Sa fonction d'évaluation utilisée est donnée dans l'équation (3.12). Elle essaye de trouver un compromis entre le nombre d'instances de chaque classe et l'exactitude de la classification.

$$\boldsymbol{Fitness_{Bal}(S)} = \begin{cases} \boldsymbol{GM - \left|1 - \frac{n^+}{n^-}\right| . Pifn^- > 0} \\ \boldsymbol{GM - Pifn^- = 0} \end{cases} \qquad \textbf{(3.12)}$$

Où :

- GM : est la mesure G-Mean ***(définie dans le chapitre 3, section 4.2.1)***.

- n^+ : est le nombre d'instances minoritaires.
- n^- : est le nombre d'instances majoritaires.
- P : est un facteur de pénalisation qui est utilisé pour garder le même nombre d'instances majoritaires et minoritaires dans la base de données réduite et de ne pas obtenir des sous bases de données réduites spécialisées pour les instances minoritaires seulement. Empiriquement, une faible valeur de P n'est pas suffisante pour l'équilibrage et une valeur élevée de P peut perdre le compromis entre l'équilibre et l'exactitude.

A.2. EBUS-MS-GM :

Il est similaire au modèle précédent, mais il sélectionne les instances majoritaires seulement. La réduction est aussi contrôlée et elle dépend d'instances minoritaires. La fonction d'évaluation utilisée est donnée dans l'équation (3.13)

$$\boldsymbol{Fitness_{Bal}(S)} = \begin{cases} \boldsymbol{GM - \left|1 - \frac{N^+}{n^-}\right| . P} & \boldsymbol{if\ n^- > 0} \\ \boldsymbol{GM - P} & \boldsymbol{if\ n^- = 0} \end{cases} \quad \textbf{(3.13)}$$

$\mathbf{N^+}$: est une constante qui représente le nombre initial d'instances minoritaires dans la base d'apprentissage.

A.3. EBUS-GS-AUC et EBUS-MS-AUC :

Ces deux modèles ont le même principe que les deux modèles précédents, mais ils utilisent la mesure de performance AUC au lieu de G-Mean.

B. Le sous-échantillonnage évolutionnaire guidé par les mesures de classification EUSCM :

Dans EUSCM, l'équilibre n'est pas contrôlé entre les classes. Il est composé de quatre modèles suivants: EUSCM-GS-GM, EUSCM-MS-GM, EUSCM-GS-AUC, EUSCM-MS-AUC.

e.2. Le sous-échantillonnage basé sur les colonies de fourmis :

ACOSampling est une méthode de sous-échantillonnage **[13]** basée sur les colonies de fourmis **[124]**. Elle a été proposée pour traiter les bases de données déséquilibrées microarray. Elle consiste à équilibrer la base de données d'abord par le sous-échantillonnage avec ACO en suivant les étapes résumées dans l'***algorithme 3.4.***

Algorithme 3.4. ACOSampling

Entrée : la base de données d'apprentissage originale S.
Le nombre d'itérations T.
Sortie : La base de données équilibrée optimale SOP.
Début
Phase 1 : Construire T sous bases de données équilibrées.
Pour i :=1 ***à*** T ***faire***
- Diviser S en deux sous bases de données: la base d'apprentissage S_i et la base de validation V_i.
- Si'= Sous-échantillonnage ACO (Si, Vi) ;

fait ;
Phase 2 : Construire la base de données équilibrée finale OPS.
- Etablir la liste de fréquence de toutes les instances majoritaires de toutes les bases de données Si'.
- Associer un rang à chaque instance majoritaire selon l'ordre décroissant de sa fréquence.
- Sélectionner quelques instances majoritaires possédant une fréquence élevée et les combiner avec toutes les instances minoritaires pour construire la base de données équilibrée optimale SOP.

Fin.

La procédure Sous-échantillonnage ACO est résumée dans l'algorithme suivant :

Algorithme 3.5. Sous-échantillonnage ACO

Entrée : la base d'apprentissage S_i et la base de validation V_i.
Sortie : la base de données optimale S_i'.
Début
Initialiser la solution optimale OPS à 0
Pour chaque itération j de l'algorithme de colonies de fourmis ***faire***
Pour chaque fourmi k ***faire***
La base de données S_{jk}=filtrage(S_i).
Le modèle C_{jk}=Apprentissage (S_{jk}).
Evaluer les performances de C_{jk} en utilisant V_i.
Fait ;
Chercher la solution optimale OPS_j : ***Si*** OPS_j > OPS ***alors*** OPS=OPS_j.
Fait ;
Retourner la base optimale S_i' qui correspond à la solution optimale OPS.
Fin.

Le filtrage de la base de données Si consiste à supprimer quelques instances majoritaires en utilisant la version modifiée ACO **[125]**. Donc, le filtrage est basé sur le déplacement d'une fourmi du nid vers la source de nourriture en passant par les différents sites et en suivant le chemin 0 ou 1, sachant que chaque site représente une instance majoritaire. Si le chemin suivi par la fourmi est 1 alors l'instance majoritaire suivante va être filtrée sinon elle va être retenue.

La fonction d'évaluation **[126]** donnée dans l'équation (3.14) est utilisée pour évaluer les performances de la base de validation V_i.

$$\boldsymbol{fitness(S') = \alpha * F - Mesure + \beta * GM + \delta * AUC} \quad \boldsymbol{avec\ \alpha + \beta + \delta = 1} \qquad \textit{(3.14)}$$

5.1.2. Les méthodes du sur-échantillonnage :

Le sur-échantillonnage sert à augmenter la taille d'une base de données déséquilibrée pour l'équilibrer par la duplication de quelques instances minoritaires. Cette duplication peut se faire avec plusieurs manières, d'où l'existence des méthodes suivantes :

a. Le sur-échantillonnage aléatoire:

Elle sert à dupliquer un certain nombre d'instances minoritaires choisies aléatoirement **[3]**.

Cependant, l'ajout de plusieurs copies d'instances minoritaires augmente le chevauchement entre ces instances **[109]**. Particulièrement, le chevauchement apparait lorsque le modèle produit contient des règles plus spécifiques pour plusieurs copies de la même instance. Par conséquent, la précision de l'apprentissage est élevée dans ce scénario et les performances du modèle pour les données du test sont généralement faibles **[108]**.

b. L'échantillonnage synthétique avec génération des données :

SMOTE (Synthetic Minority Oversampling Technique) est une méthode d'échantillonnage synthétique avec génération de données **[127]** qui a réalisé plusieurs succès dans divers domaines **[3]**. Elle crée un exemple synthétique $\boldsymbol{x_{new}}$ pour chaque instance minoritaire $\boldsymbol{x_i}$ comme suit :

- Déterminer les ***K- plus proches voisins*** de x_i (Voir ***figure 3.5***). Les K- plus proches voisins sont aussi des instances minoritaires dont la distance euclidienne entre elles et $\boldsymbol{x_i}$ est la plus petite).
- Sélectionner aléatoirement un de ses K- plus proches voisins $\boldsymbol{x_i'}$.
- Appliquer l'équation : $\boldsymbol{x_{new} = x_i + (x_i' - x_i) \times \delta}$ **(** $\boldsymbol{\delta estunnombrealéatoire \in [0, 1])}$**.** Nous comprenons donc que $\boldsymbol{x_{new}}$ est un point de segment joignant $\boldsymbol{x}_i$ et $\boldsymbol{x'_i}$ (Voir ***figure 3.6***).

SMOTE généralise des régions de décision pour les classes minoritaires pour ne pas être ignorées par les algorithmes d'apprentissage.

SMOTE a deux problèmes **[128] :** la sur généralisation et la variance (le manque de flexibilité). Le premier problème est dû à la généralisation aveugle de la zone minoritaire sans tenir en compte la classe majoritaire, ce qui augmente le nombre d'occurrence de chevauchement entre les classes. Cependant, le deuxième problème concerne le nombre

d'échantillons synthétiques générés qui est fixé à l'avance sans prendre en considération le taux de rééquilibrage.

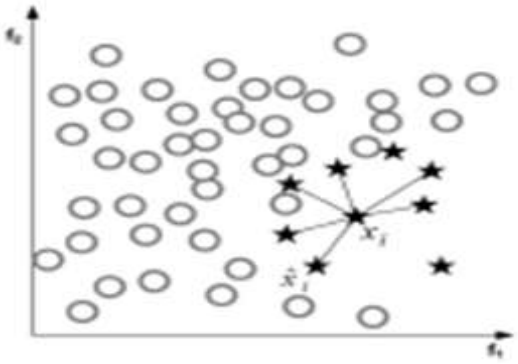

Figure 3.5. Exemple de K-plus proches voisins de x_i (K=6)

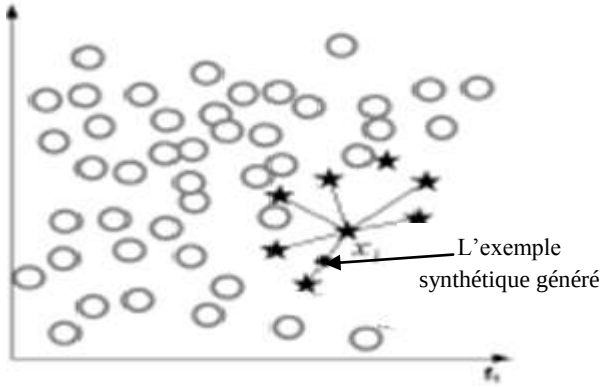

Figure 3.6. Création des exemples synthétiques

c. MSMOTE :

SMOTE génère des exemples synthétiques pour chaque instance minoritaire sans prendre en considération la distribution d'instances minoritaires et les instances minoritaires bruitées dans les bases de données. Pour cette raison, MSMOTE **[129]** divise les instances minoritaires en 03 groupes par la détermination des K-plus proches voisins de chaque instance:

- Une instance est sécuritaire (safe) : si le nombre de ses proches voisins appartenant à la classe minoritaire est supérieur à ceux appartenant à la classe majoritaire.
- Une instance est limite (borderline) : si le nombre de ses proches voisins appartenant à la classe minoritaire est inférieur à ceux appartenant à la classe majoritaire.
- Une instance est bruitée (noise) : si tous ses proches voisins possèdent la classe majoritaire.

MSMOTE génère des exemples synthétiques pour les instances sécuritaires de la même manière que SMOTE. Par contre, pour chaque exemple limite, il sélectionne le plus proche voisin pour générer l'exemple synthétique. Mais, il ne génère rien pour les exemples bruités (ils diminuent les performances d'un modèle).

d. Borderline-SMOTE :

Les instances de la frontière et celles qui sont à proximité (sont appelée aussi les instances de la frontière) sont plus susceptibles d'être mal classées que celles qui sont loin de la frontière, donc elles sont les plus importantes pour la classification.

Basant sur cette analyse, les instances de la frontière contribuent peu dans la classification. Donc, la méthode Borderline-SMOTE a été proposée pour appliquer le sur-échantillonnage

sur les instances minoritaires de la frontière au lieu de l'appliquer sur toutes les instances minoritaires **[130]**. Les étapes de cette méthode sont données dans l'***algorithme 3.6***.

Algorithme 3.6. Borderline-SMOTE

Entrée : la base de données déséquilibrée composée de l'ensemble d'instances minoritaires S_{min} et l'ensemble d'instances majoritaires S_{maj}.
K : le nombre de voisins les plus proches.
Sortie : la base de données équilibrée.
Début
L'ensemble DANGER :=Φ.
Pour chaque instance x_i appartenant à l'ensemble S_{min} ***faire***
Déterminer l'ensemble des plus proches voisins.
Identifier le nombre des proches voisins appartenant à l'ensemble S_{maj}.
Si le nombre de plus proches voisins majoritaires est supérieur à ceux qui sont minoritaire ***alors***
DANGER :=DANGER ∪x_i
fin ;
fait ;
Appliquer SMOTE sur l'ensemble DANGER pour générer les exemples synthétiques.
Fin.

e. L'échantillonnage synthétique adaptatif ADASYN:

L'idée de l'algorithme ***ADASYN*** est d'utiliser une fonction appelée ***densité*** comme un critère automatique pour prendre une décision sur le nombre d'exemples synthétiques qui doivent être générés pour chaque instance minoritaire **[131]**. Ce nombre est déterminé par adaptation en changeant à chaque fois le poids des instances minoritaires pour compenser la distribution des données. Les étapes de cet algorithme sont les suivantes:

Algorithme 3.7. L'algorithme ADASYN

Entrée :la base de données déséquilibrée composée de l'ensemble d'instances minoritaires S_{min} et l'ensemble d'instances majoritaires S_{maj}.
K : le nombre de voisins les plus proches.

Sortie : la base de données équilibrée.

Début

- Calculer le nombre d'exemples synthétiques nécessaires ***G*** qui doivent être générés :

$$\boldsymbol{G} := \left(|\boldsymbol{S_{maj}}| - |\boldsymbol{S_{min}}|\right) \times \boldsymbol{\beta} \qquad \textbf{(3.15)}$$

Où : $\boldsymbol{\beta}$ est un paramètre ϵ [0,1] utilisé pour spécifier le degré d'équilibre désiré après la génération d'exemples synthétiques.

- Pour chaque exemple $\boldsymbol{x_i}$ de $\boldsymbol{S_{min}}$, chercher les K-plus proches voisins en utilisant la distance euclidienne.
- Calculer le ratio $\boldsymbol{\Gamma_i}$:

$$\Gamma_i := \frac{\Delta_i / K}{Z} \qquad \textbf{(3.16)}$$

- $\boldsymbol{\Delta_i}$ **:** représente le nombre d'exemples de l'ensemble des K-plus proches voisins qui appartiennent à $\boldsymbol{S_{maj}}$.
- ***Z*** est un facteur de normalisation (Γ_i est une fonction de distribution et $\sum \Gamma_i = 1$).

- Calculer le nombre d'exemples synthétiques $\boldsymbol{g_i}$ qui doivent être générés pour chaque exemple $\boldsymbol{x_i}$ dans $\boldsymbol{S_{min}}$: $\mathbf{g_i} := \mathbf{\Gamma_i} \times \mathbf{G}$ **(3.17)**
- Finalement, générer g_i exemples synthétiques pour chaque exemple x_i dans S_{min} par l'algorithme ***SMOTE***.

Fin.

f. Le sur-échantillonnage à base de cluster CBO :

L'algorithme CBO **(Cluster-Based Oversampling)** divise d'abord la base d'apprentissage en K clusters par la méthode K-Moyennes **[42]** pour chaque classe pour obtenir M_{maj} clusters majoritaires et M_{min} clusters minoritaires **[132]**.

CBO gonfle les clusters majoritaires par le sur-échantillonnage pour qu'ils aient tous la même taille. Donc, le nombre total d'instances majoritaires devient N_{CBO}. Ensuite, nous appliquons le sur-échantillonnage sur les clusters minoritaires pour que le nombre d'instances minoritaires dans chaque cluster devienne N_{CBO} / M_{min} et que le nombre d'instances minoritaires soit égal au nombre d'instances majoritaires.

Exemple :

La ***figure 3.7*** montre la distribution initiale des clusters. La classe majoritaire a 03 clusters (M_{maj}=3) A, B et C qui contiennent 20, 10 et 8 instances respectivement. La classe minoritaire a 02 clusters (M_{min}=2) D et E, chaque cluster contient 8 et 5 instances respectivement.

La ***figure 3.8*** montre l'état de la base de données après l'application de l'algorithme CBO.

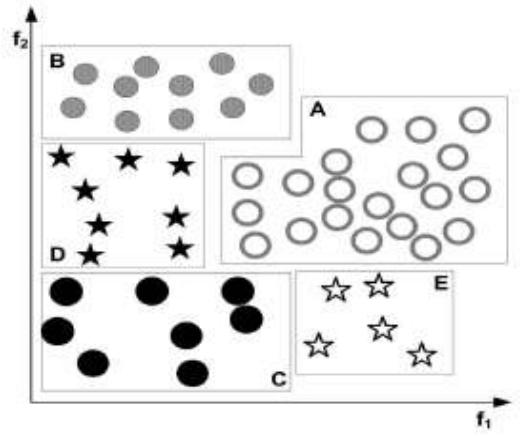

Figure 3.7. ***Distribution initiale d'instances avant l'application de CBO***

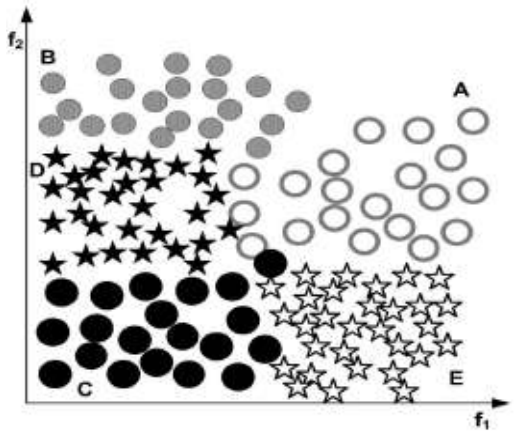

Figure 3.8. ***Distribution finale après l'application de CBO***

5.1.3. Les méthodes hybrides :

Ces méthodes combinent le sur-échantillonnage et le sous-échantillonnage. Elles éliminent quelques instances minoritaires produites par le sur-échantillonnage pour éviter le chevauchement entre les classes. Parmi les méthodes hybrides, nous avons:

a. SMOTE+Liens de Tomek :

Dans cette méthode, les liens de Tomek sont appliqués après la génération d'instances minoritaires synthétiques par SMOTE afin d'éviter le problème de chevauchement entre les classes **[17]**.

b. SMOTE+ENN:

Cette méthode utilise ENN pour supprimer les instances majoritaires et minoritaires. Pour cela, chaque instance de la base d'apprentissage mal classée par ses 03 plus proches voisins est supprimée **[17]**. ENN supprime plus d'instances que les liens de Tomek.

5.2. Niveau algorithme :

Les approches de ce niveau sont des approches internes. Elles se basent sur la modification des algorithmes de la classification complète ou par la proposition des algorithmes spécifiques à la classification partielle.

5.2.1. Modification des algorithmes existants :

Quelques algorithmes utilisés dans la classification complète sont modifiés afin de les adapter à la classification partielle. Dans ce chapitre, nous allons montrer les modifications apportées aux arbres de décision et aux machines de vecteurs de support SVMs.

a. Les arbres de décision :

Dans la phase de construction d'un arbre, C4.5 utilise le gain d'information **[63]** pour sélectionner l'attribut nœud. L'attribut choisi est celui qui maximise le gain d'information c'est-à-dire une valeur élevée de la confiance. Cependant, cette mesure n'est pas convenable pour les bases de données déséquilibrées parce que les règles les plus confidentes n'impliquent pas qu'elles sont les plus signifiantes et quelques règles les plus signifiantes peuvent ne pas être les plus confidentes (peuvent ne pas avoir une confiance élevée). Le même problème se pose pour CART qui utilise la fonction de GINI **[63]**. Ces algorithmes focalisent sur l'antécédent pour trouver la classe.

Solution: proposition des mesures non sensibles **[133]** aux bases de données déséquilibrées. D'où l'apparition de l'approche connue sous le nom **Class Confidence Proportion Decision Tree CCPDT.**

CCPDT est une approche robuste et insensible à la distribution des classes. Il génère des règles qui sont statistiquement significatives **[134].** Elle se focalise sur chaque classe pour trouver l'antécédent le plus signifiant. De cette façon, toutes les instances sont partitionnées selon leurs classes. Par conséquent, les instances qui appartiennent aux différentes classes ne vont pas avoir un impact sur les autres. Pour cela, la nouvelle mesure Class Confidence CC a été proposée pour trouver les antécédents les plus intéressants de chaque classe. Elle est définie dans l'équation (3.18)

$$\boldsymbol{CC(X \rightarrow y) = \frac{Support(X \cup y)}{Support(y)} = \frac{TP}{TP + FN}} \qquad \textbf{(3.18)}$$

Où :

- Support $(\boldsymbol{X \cup y})$: est défini dans le chapitre 2, section 5.3.
- Support (y) : est la fréquence de y.

Remarques:

1) La principale différence enctre la confiance et CC est le dénominateur : Supp(y) est utilisé au lieu de Supp(X) afin de se focaliser sur chaque classe seulement.
2) Dans les bases de données équilibrées, TP+FN<<FP+TN n'a pas d'influence sur CC (non sensible).

3) Pour la confiance, combien d'instances prédites positivement (respectivement négativement) sont actuellement positives (respectivement négatives).
4) Pour CC, elle se focalise sur combien d'instances qui sont actuellement positives (respectivement négatives) sont prédites positivement (respectivement négativement).

Cependant, l'obtention des règles qui ont une valeur élevée de CC est toujours insuffisante pour résoudre le problème de la classification. Donc, il est nécessaire de veiller à ce que les classes impliquées par ces règles n'ont pas seulement une grande confiance, mais plus intéressantes que leurs classes alternatives. Par conséquent, la nouvelle mesure Class Confidence Proportion CCP a été proposée. Elle est définie dans l'équation (3.19).

$$\boldsymbol{CCP(X \rightarrow y) = \frac{CC(X \rightarrow y)}{CC(X \rightarrow y) + CC(X \rightarrow \overline{y})}} \qquad \textbf{(3.19)}$$

Donc, l'approche CCPDT modifie l'algorithme C4.5 en remplaçant l'entropie (le critère de partition des attributs) par CCP.

b. Les machines de vecteurs du support SVMs:

Nous montrons d'abord l'insuffisance des SVMs dans le cas de la classification partielle puis nous présentons les différentes solutions proposées.

Dans la classification partielle, l'hyperplan idéal est au voisin d'instances majoritaires et la frontière de décision est très proche aux instances minoritaires, comme il est illustré dans la ***figure 3.9***. Dans ce cas, les vecteurs du support représentant les instances minoritaires sont loin de l'hyperplan idéal. Donc, leurs contribution à l'hypothèse finale est peu **[135] [136]**.

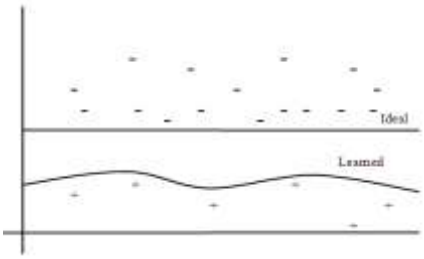

Figure 3.9. ***L'hyperplan dans les bases de données déséquilibrées***

Pour résoudre ce problème, plusieurs méthodes ont été proposées qui se diffèrent du mécanisme utilisé. Par exemple, les méthodes qui sont basées sur l'ajustement de la marge transforment la fonction du noyau de manière conforme pour élargir la marge autour de la région de la frontière. Par exemple, dans **[136]** les trois approches suivantes ont été présentées :

b.1. Mouvement frontalier :

Mouvement frontalier ***(Boundary Movement BM)*** modifie le coefficient b dans la fonction noyau comme il est indiqué dans l'équation (3.20) où Δb représente combien la frontière est déplacée. Cette méthode est considérée comme une méthode de post-traitement.

$$\boldsymbol{f(x) = sgn\left(\sum_{i=1}^{n} y_i * \alpha_i * K(x, x_i) + b + \Delta b\right)} \qquad \textbf{(3.20)}$$

b.2. Pénalités biaisées:

Pénalités biaisées ***(Biased Penalities BP)*** introduit différents facteurs de pénalité C^+ et C^- pour la classe minoritaire et majoritaire dans la fonction objective dans la formulation Lagrangien. Ces facteurs reflètent l'importance des classes durant la phase d'apprentissage.

b.3. Alignement de la classe frontalière:

Alignement de la classe frontalière ***(Class Boundary Alignement*** **CBA*)*** élargit beaucoup plus la frontière autour de la classe minoritaire par rapport à la frontière autour de la classe majoritaire.

5.2.2. Proposition des algorithmes spécifiques :

Des algorithmes spécifiques ont été proposés pour traiter le problème de la classification partielle. Parmi ces algorithmes, nous présentons dans les deux sous sections suivantes les algorithmes RLSD et LUPC.

a. L'algorithme Rule Learning for Skewed Datasets RLSD:

RLSD est un algorithme efficace pour le traitement des bases de données déséquilibrées. Son processus de découverte conduit d'une recherche spécifique à une recherche générale, en cherchant d'abord à découvrir des règles pour les instances d'apprentissage minoritaires puis les comparer avec les instances majoritaires **[137]**. Il contient les 03 phases de recherche suivantes :

1. La phase de discrétisation : elle consiste à diviser les valeurs d'un attribut numérique en un petit nombre d'intervalles où chaque intervalle est mappé par un symbole discret. Par exemple, la discrétisation de l'attribut âge est : [0..11]= Enfant, [12..17]=adolescent, [18..44]=adulte, [45..69]= âge moyen, [70…..]=âgé.

2. La phase de génération des règles : la génération des règles dans RLSD est un processus de découverte de données fréquentes pour les instances minoritaires. L'***algorithme 3.8*** résume cette phase :

Algorithme 3.8. RLSD : algorithme de génération des règles

Entrée : l'ensemble d'instances minoritaires P.
le nombre maximal des règles permises M.
Sortie : l'ensemble Règles.
Début
Règles :=Ø ;
Pour chaque instance p ϵ P ***faire***
Règle R :=p ;
La règle R est considérée comme la règle initiale.
Si R n'appartient pas à l'ensemble Règles ***alors***
Fusion_Règles(R, Règles, M) ;
Ajout_Règles(R, Règles, M) ;
fin ;
Fait ;
Pour chaque règle RE ϵ Règles ***faire***
Si TPrate(RE)<Min_TPrate ***alors*** supprimer RE ;
Fait ;
Fin.

Les algorithmes 3.9 et 3.10 résument les étapes des procédures Fusion_Règles et Ajout_Règles.

***Algorithme 3.9. RLSD : Fusion Règles*(R, Règles, M)**

Pour chaque règle RE dans règles ***faire***
Si RE et R partagent les mêmes conditions ***alors***
Générer une nouvelle NR avec les conditions communes ;
Ajout_Règles(NR, Règles, M) ;
fin ;
Fait ;

Algorithme 3.10. RLSD : Ajout_Règles(NR, Règles, M)

Si NR n'appartient pas à Règles ***alors***
Règles :=Règles U NR ;
Si |Règles| > M ***alors***
Supprimer une règle choisie aléatoirement.
fin ;

3. La phase d'évaluation et sélection des règles : dans cette phase, RLSD compare chaque règle générée avec les instances majoritaires. Cette phase contient les deux étapes suivantes :

Etape d'évaluation des règles : RLSD calcule la précision de chaque règle générée par la correspondance avec chaque instance majoritaire. Une règle est supprimée si sa précision est inférieure à la précision minimale.

Etape de sélection des règles : RLSD sélectionne la règle qui a la valeur de F-mesure ***(définie dans la section 4.2.2)*** la plus élevée, puis il supprime toutes les instances minoritaires couvertes par cette règle. Ensuite, F-mesure est recalculée pour les règles restantes en utilisant les instances minoritaires restantes. Ce processus est répété jusqu'à ce qu'il ne reste plus d'instances minoritaires ou il n'y a pas une règle qui couvre les instances minoritaires restantes.

b. Learning Unbalanced Positive Class LUPC:

La principale caractéristique de LUPC **[138]** est la combinaison de la méthode d'induction des règles par séparation et conquête (Separate and Conquer rule induction) **[139]** et les règles d'association **[45]**. La force de LUPC réside dans l'utilisation des seuils adaptatifs sur la précision et la couverture des règles dans les différentes heuristiques de recherche. Dans ce qui suit, nous formulons d'abord ce problème, puis nous donnons les mesures utilisées et enfin nous détaillons les différentes étapes de cet algorithme.

Formulation du problème : une base d'apprentissage déséquilibrée est l'union d'instances positives Pos (ou minoritaires) et d'instances négatives Nég (ou majoritaires) avec Pos <<Nég.

La règle R à prédire pour la classe positive C^+ est notée: $\bigwedge_{j=1}^{m}(A_{ij} = v_{ij}) \longrightarrow C^+$, avec $Cov(R) = Cov^+(R) \cap Cov^-(R)$, où m est le nombre d'attributs et v_{ij} est la $j^{ième}$ valeur du l'ième attribut.

LUPC utilise trois mesures de performances ; l'exactitude acc, le taux d'erreur err et le taux de couverture positive PCR (Positive cover ratio). Elles sont définies dans les équations (3.21), (3.22) et (3.23).

$$\boldsymbol{acc(R) = \frac{|Cov^+(R)|}{|Cov(R)|}} \qquad \textbf{(3.21)}$$

$$\boldsymbol{err(R)=1-acc(R)} \qquad \textbf{(3.22)}$$

$$\boldsymbol{PCR(R) = \frac{|Cov^+(R)|}{|D|}} \qquad \textbf{(3.23)}$$

Une règle est ***αβ- forte (αβ-strong)*** si les inéquations **(3.24) et (3.25)** sont vérifiées:

$$\mathbf{acc(R) \geq \alpha} \qquad \textbf{(3.24)}$$

$$\boldsymbol{PCR(R) \geq \beta} \qquad \textbf{(3.25)}$$

Où les paramètres α et β sont les seuils, avec $0 \leq \alpha, \beta \leq 1$.

Une règle est ***non αβ- forte*** si l'inéquation (3.26) est vérifiée:

$$\boldsymbol{Cov^-(R) \geq \frac{1-\alpha}{\alpha} * Cov^+(R)} \qquad \textbf{(3.26)}$$

L'algorithme LUPC : les étapes de LUPC sont résumées dans l'***algorithme 3.11***.

Algorithme 3.11. LUPC

Entrée : Pos, Nég,
le seuil minimal de l'exactitude : min_acc
le seuil minimal du taux de couverture positive : min_cov
Sortie : l'ensemble de règles Ens_Règles.
Début
Ens_Règles :=Ø ;
α, β :=**Initialiser(Pos, min_acc, min_cov)** ;
Tant que (Pos ≠ Ø et (α, β)≠(min_acc, min_cov)) ***faire***
Règle R := ***Meilleure_Règle(Pos, Nég, α ,β)*** ;
Si (R ≠Ø) ***alors***
Pos := Pos – {les instances couvertes par R}
Ens_Règles := Ens_Règles U R ;
Sinon
Réduire(α,β) ;
fin ;
fait ;
Ens_Règles :=Post traitement(Ens_Règles) ; *// Est une étape optionnelle*
Fin.

Algorithme 3.12. LUPC : Meilleure_Règle (Pos,Neg,α,β)

Début
Ens_Règles_Candidates :=Ø ;
Paires_Attribut_Valeur(Pos,Nég,α,β) ;
Tant que Condition(Pos, Nég, α, β) ***faire***
Règles_Candidates(Pos, Nég, α, β) ;
Meilleure règle← Première règle candidate ϵ Ens_Règles_Candidates ;
Fin.

Explication des procédures utilisées dans l'algorithme LUPC :

- **Initialiser (Pos, min_acc, min_cov) :** en fonction du biais spécifié par l'utilisateur sur le taux de couverture positive ou sur l'exactitude, cette procédure initialise α et β comme min_acc et min_cov respectivement. Autrement, α est initialisé à 0.95 ou à min_acc si min_acc est supérieur à 0.95. β est initialisé à la valeur maximale du taux de couverture positive des paires_Attributs_Valeurs disponibles sur les instances minoritaires Pos.
- **Paires_Attribut_Valeur (Pos,Nég,α,β)** : le but de cette procédure est de construire ***l'ensemble des Paires_Attribut_Valeur*** comme suit :

1. Construire l'ensemble E1 de toutes les paires : chaque paire (Attribut, Valeur) disponible pour les instances de classe positive est considérée comme une partie de condition dont la classe est C^+ (cette règle contient une seule prémisse).

2. Choisir l'ensemble E2 des paires candidates (E2 est un sous ensemble de E1) : chaque paire appartenant à E1 est considérée comme une paire candidate si elle couvre plus de α*β*|D| instances minoritaires (c'est une contrainte nécessaire pour devenir αβ-forte).

3. Identifier les paires αβ-fortes : chaque paire candidate sera vérifiée sur les instances de classe Nég pour voir si elle est αβ-forte.

4. Ordonner les paires αβ-fortes soit par leur précision soit par leur taux de couverture positive.

5. Choisir η candidates Paires_Attribut_Valeur qui sont αβ-fortes et les rajouter dans ***l'ensemble des Paires_Attribut_Valeur***. Le cas suivant peut apparaitre :

Si le nombre de paires αβ-forte<η alors rajouter les paires qui ne sont pas αβ-fortes qui ont soit une précision élevée ou un taux de couverture positive élevé.

- **Règles_Candidates(Pos,Nég,α,β) :** la sortie de cette procédure est la génération d'un ensemble donné de règles candidates contenant γ règles appartenant à l'ensemble des Paires_Attribut_Valeur en passant par les étapes suivantes :

1. Ordonner l'ensemble des Paires_Attribut_Valeur selon la précision ou/et selon le taux de couverture positive.

2. Construire l'ensemble des règles candidates comme suit :

Si le nombre des paires αβ-fortes est supérieur ou égal à γ alors ajouter γ paires à l'ensemble des règles candidates sinon :

2.1. Mettre les paires αβ-fortes et les paires non αβ-fortes dans l'ensemble des règles candidates, c'est-à-dire cet ensemble est composé de deux parties changeables.

2.2. Supprimer les règles non αβ-fortes ayant un taux de couverture positive inférieur à β.

2.3. Améliorer l'ensemble des règles candidates par l'exécution itérative de la procédure suivante tant qu'il y a un changement dans les règles non αβ-fortes ou le nombre de règles dans l'ensemble de règles candidates est inférieur à γ (c'est-à-dire **Condition (Pos,Nég,α,β)**) :

Générer de nouvelles règles en combinant chaque règle non αβ-forte de l'ensemble des règles candidates avec les paires Attribut_Valeur qui sont dans l'ensemble des Paires_Attribut_Valeur : si les règles générées deviennent αβ-fortes alors elles seront insérées dans la première partie de l'ensemble des règles candidates.

2.4. Rejeter les règles qui vérifient la condition suivante: $\boldsymbol{Cov^{-}(R) \geq \frac{1-\alpha}{\alpha} \times Cov^{+}(R)}$**.**

Remarque : les paramètres γ et η peuvent influencer sur l'algorithme LUPC. Généralement, s'ils ont des valeurs élevées alors il y aura une grande chance pour obtenir des meilleures règles.

- **Réduire (α, β) :** cette procédure réduit les valeurs α et β graduellement par le taux Δa et Δc respectivement. Les quantités par défaut utilisées sont Δa=2% et Δc=1%.

5.3. Niveau hybride :

Quelques méthodes de la classification complète ne peuvent pas traiter le problème de la classification dans les bases de données déséquilibrées sans être combinées avec d'autres techniques. Parmi ces méthodes, nous citons ***les méthodes à plusieurs modèles*** et ***l'apprentissage sensible aux coûts*** que nous allons les détailler dans les sous sections suivantes :

5.3.1. Les méthodes à plusieurs modèles :

a. Boosting :

Les algorithmes de boosting **(*vus dans le chapitre 2, section 6.3.1*)** se focalisent sur les instances difficiles à classer sans différencier leurs classes. Selon le Prof. Zhou Zhihua **[140]**, les algorithmes de boosting sont très efficaces et capables de traiter le problème de la classification dans les bases de données déséquilibrées, car les instances minoritaires sont susceptibles d'être mal classées et par conséquent, elles vont avoir des poids plus élevés dans les itérations suivantes.

Cependant, Mikel G. et al. **[141]** ont estimé que l'intégration des méthodes d'échantillonnage des données peut réduire les coûts supplémentaires pour détecter automatiquement la distribution optimale des classes et des échantillons représentatifs et aussi réduit le biais d'un algorithme d'apprentissage spécifique.

D'où l'existence de plusieurs algorithmes tels que SMOTEBoost, RUSBoost et DataBoost-IM.

a.1 SMOTEBoost :

SMOTEBoost altère la distribution de la base d'apprentissage par l'ajout d'instances minoritaires générées par SMOTE afin de la fournir à l'algorithme AdaBoost **[142]**. M2. Les résultats expérimentaux indiquent que SMOTEBoost est plus performant que boosting standard et SMOTE avec un seul modèle.

a.2. RUSBoost :

RUSBoost fonctionne de manière similaire à SMOTEBoost, mais il applique le sous-échantillonnage aléatoire sur la base d'apprentissage à chaque itération **[143]**.

a.3. DataBoost-IM :

Cet algorithme combine l'algorithme AdaBoost. M1 avec une stratégie de génération de données **[103]**. Il se diffère des algorithmes précédents car il effectue le processus d'équilibre

pour les classes majoritaires et les instances minoritaires après l'identification d'instances difficiles. L'***algorithme 3.13*** résume les étapes de DataBoost-IM :

Algorithme 3.13. DataBoost-IM

Entrée : la base de données déséquilibrée composée de Nmin instances minoritaires et Nmaj instances majoritaires.

Sortie: la nouvelle base de données équilibrée.

Début

1. Identifier les instances difficiles :
   1.1. Construire le modèle C.
   1.2. Identifier les instances mal classées par C : E_{smin} instances minoritaires et E_{smaj} instances majoritaires. Ces instances sont appelées les graines.
   1.3. Ordonner les graines selon l'ordre croissant de leurs poids.
   1.4. Construire les ensembles MS et ML contenant M_S instances minoritaires et M_L instances majoritaires ayant les poids les plus élevés.
2. Générer les exemples synthétiques :
   2.1. Chaque graine majoritaire doit générer N_{maj} exemples synthétiques.
   2.2. Chaque graine minoritaire doit générer N_{min} exemples synthétiques.
3. Equilibrer le poids des instances de la nouvelle base d'apprentissage :
   3.1. Assigner à chaque exemple synthétique un poids initial.
   3.2. Calculer la somme des poids d'instances majoritaires W_{maj}.
   3.3. Calculer la somme des poids d'instances minoritaires W_{min}.
   3.4. Equilibrer les poids : deux cas peuvent exister :
   Si $W_{maj}>W_{min}$ alors le poids de chaque instance minoritaire doit être multiplié par W_{maj}/W_{min}.
   Si $W_{min}>W_{maj}$ alors le poids de chaque instance majoritaire doit être multiplié par W_{min}/W_{maj}.

Fin.

b. Bagging :

Dans les bases de données déséquilibrées, les instances majoritaires dominent aussi dans les échantillons. Le principal facteur à appliquer sur l'ensachage, pour l'adapter au problème de la classification dans les bases de données déséquilibrées, est la manière de collecter les échantillons. Nous distinguons 03 algorithmes principaux dans cette famille ; OverBagging, UnderBagging et UnderOverBagging.

b.1. OverBagging:

Dans cet algorithme, la distribution des instances doit être prise en considération lors de la construction d'un sac pour que le nombre d'instances minoritaires soit égal au nombre d'instances majoritaires N_{maj} **[144]**. Au lieu de construire les sacs aléatoirement, nous procédons à l'application du sur-échantillonnage selon les deux possibilités suivantes:

La première possibilité :

Les instances minoritaires sont dupliquées par le sur-échantillonnage aléatoire et les instances majoritaires sont rajoutées directement ou elles sont sélectionnées par un tirage aléatoire avec remplacement pour augmenter la diversité.

La deuxième possibilité : **SMOTE-Bagging [144]**

Les instances minoritaires sont obtenues comme suit: a%*N_{maj} sont choisies par le tirage aléatoire avec remplacement à partir de la base de données originale et le reste est généré par SMOTE. Le facteur a est appelé le taux d'échantillonnage qui s'augmente à chaque itération allant de 10% dans la première itération à 100% dans la dernière (toujours en multiple de 10).

b.2. UnderBagging:

Dans UnderBagging, le nombre d'instances majoritaires est réduit au nombre d'instances minoritaires dans chaque sac. Toutes les instances minoritaires peuvent être existées dans le même sac. Mais pour la diversité, elles peuvent être choisies par un tirage aléatoire avec remise **[145]**.

b.3. UnderOverBagging :

UnderOverBagging suit les méthodologies de UnderBagging et OverBagging mais il est identique à SMOTEBagging. Le nombre d'instances de chaque classe est $a\%*N_{maj}$. Le premier modèle est construit avec un nombre d'instances inférieur au dernier modèle **[145]**.

5.3.2. L'apprentissage sensible aux coûts avec boosting:

Dans chaque itération de boosting, les poids d'instances mal classées augmentent par le même ratio quel que soit leurs classes. Cependant, dans les bases de données déséquilibrées, le nombre d'instances minoritaires mal classées est plus élevé. D'où la nécessité de distinction entre les différents types d'instances dans la phase d'attribution des poids. Donc, les poids élevés sont attribués aux instances minoritaires pour qu'elles aient plus de chance d'être bien classées.

Pour atteindre cet objectif, les coûts de mal classifications ont été introduits dans l'équation de mise à jour de poids. Différents algorithmes ont été proposés qui diffèrent dans la façon dont ils mettent à jour les poids, parmi ces algorithmes, nous citons AdaC1, AdaC2 et AdaC3.

a. AdaC1:

Cet algorithme modifie AdaBoost par l'introduction des coûts de mal classification dans la formule de mise à jour de poids à l'intérieur de la partie exponentielle (voir l'équation (3.27)) et dans l'équation du calcul de la performance du modèle (Equation (3.28)) **[146]**.

$$\boldsymbol{D_{t+1}(i) = D_t(i) * e^{-\alpha_t * C_i * h_t(x_i) * y_i}} \qquad \textbf{(3.27)}$$

$$\boldsymbol{\alpha_t = \frac{1}{2} * \ln \frac{1 + \sum_{i,y_i=h_t(x_i)} C_i * D_t(i) - \sum_{i,y_i \neq h_t(x_i)} C_i * D_t(i)}{1 - \sum_{i,y_i=h_t(x_i)} C_i * D_t(i) + \sum_{i,y_i \neq h_t(x_i)} C_i * D_t(i)}} \quad \textit{(3.28)}$$

b. AdaC2:

AdaC2 intègre les coûts dans l'équation de mise à jour de poids mais à l'extérieur de la partie exponentielle **[146]** :

$$\boldsymbol{D_{t+1}(i) = C_i * D_t(i) * e^{-\alpha_t * h_t(x_i) * y_i}} \quad \textit{(3.29)}$$

Par conséquent, l'équation de α_t devient :

$$\boldsymbol{\alpha_t = \frac{1}{2} * ln \frac{\sum_{i,y_i=h_t(x_i)} C_i * D_t(i)}{\sum_{i,y_i \neq h_t(x_i)} C_i * D_t(i)}} \quad \textit{(3.30)}$$

c. AdaC3:

Dans AdaC3, les coûts sont introduits à l'intérieur et à l'extérieur de la partie exponentielle (voir équation (3.31)) **[146].**

$$\boldsymbol{D_{t+1}(i) = C_i * D_t(i) * e^{-\alpha_t * C_i * h_t(x_i) * y_i}} \quad \textit{(3.31)}$$

L'équation de mise à jour de poids devient :

$$\boldsymbol{\alpha_t = \frac{1}{2} * ln \frac{\sum_i D_t(i) + \sum_{i,y_i=h_t(x_i)} C_i^2 * D_t(i) - \sum_{i,y_i \neq h_t(x_i)} C_i^2 * D_t(i)}{\sum_i D_t(i) - \sum_{i,y_i=h_t(x_i)} C_i^2 * D_t(i) + \sum_{i,y_i \neq h_t(x_i)} C_i^2 * D_t(i)}} \quad \textit{(3.32)}$$

5.3.3. L'apprentissage sensible aux coûts avec machines de vecteurs du support :

Les SVMs sont intégrées avec les méthodes d'échantillonnage pour traiter le problème des données déséquilibrées. Parmi ces méthodes, nous citons SMOTE avec différents coûts (SMOTE with Different Costs SDC) et les méthodes à plusieurs modèles de sous/sur-échantillonnage SVM.

a. SMOTE avec différents coûts SDC:

SDC est le résultat d'application de SMOTE avec les différentes erreurs de coûts (Different Error Costs) DEC **[147]**. Cette méthode sert d'abord à pousser la frontière de décision loin d'instances minoritaires et aussi à augmenter leurs nombre. Pour atteindre le premier objectif, Veropoulos et al **[148]** propose l'utilisation de différents coûts pour les classes minoritaires et majoritaires. Les instances minoritaires sont aussi dupliquées par SMOTE pour les rendre densément distribuées dans le but de garantir la frontière la plus bien définie (voir ***les figures 3.10 et 3.11)***).

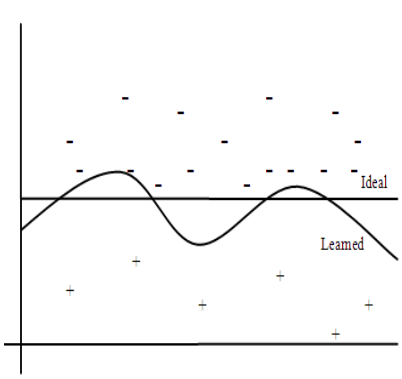

Figure 3.10. ***Frontière de décision après l'application de DEC***

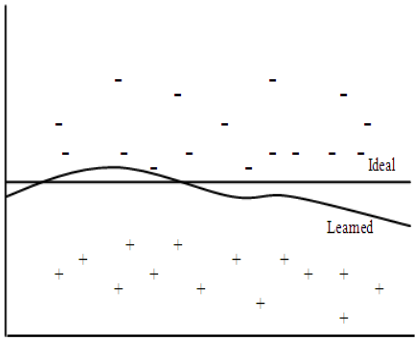

Figure 3.11. ***Frontière de décision après l'application de SDC***

b. Les méthodes à plusieurs modèles sous/sur échantillonnage SVM :

Dans cette catégorie, la base d'apprentissage doit subir un prétraitement afin de l'équilibrer puis elle sera fournie au SVM pour construire le modèle optimal **[149] [150] [151]**. Par exemple, ensemble de sous-échantillonnage SVM (Ensemble of Under-Sampling SVMs) EUS-SVMs **[151]** appelle SVM N fois sur N bases d'apprentissage. Chaque base d'apprentissage contient toutes les instances minoritaires et quelques instances majoritaires sélectionnées à l'aide de l'échantillonnage aléatoire. Le nombre d'instances majoritaires sélectionnées est égal au nombre d'instances minoritaires. Le modèle final est construit par la combinaison de N modèles produits. Les étapes sont illustrées dans l'***algorithme 3.14***.

Algorithme 3.14. Ensemble de sous-échantillonnage SVM

Entrée: la base d'apprentissage et le nombre d'itérations N.
Sortie: le modèle final.
Début
Pour i :=1 a N ***faire***
Construire la base d'apprentissage numéro i (BDA_i):
Copier toutes les instances minoritaires dont le nombre est égal à N^+.
Sélectionner N^+ instances majoritaire par l'échantillonnage aléatoire.
Appliquer SVM sur BDA_i
fait ;
Combiner les modèles finaux.
Fin.

6. Conclusion :

Dans ce chapitre, nous avons présenté les approches et les méthodes utilisées pour extraire les règles de classification à partir des bases de données déséquilibrées. Ces méthodes ont été divisées en trois grands nivaux ; le niveau données, le niveau algorithme et le niveau hybride.

En résumé, l'extraction des règles de classification se fait de manière approchée car il n'existe pas une méthode précise qui peut produire des règles pertinentes, utiles et intéressantes. Donc, c'est un problème combinatoire auquel nous devons trouver une solution satisfaisante en un temps raisonnable.

Chapitre 4 :

LE SOUS-ECHANTILLONNAGE PAR LES ALGORITHMES GENETIQUES (USGA)

Et

LE SOUS-ECHANTILLONNAGE PAR LES ALGORITHMES MEMETIQUES (USGA_ILS)

Chapitre 4 Le sous-échantillonnage par les algorithmes génétiques (USGA) et le sous-échantillonnage par les algorithmes mémétiques (USGA_ILS)

1. Introduction :

Dans ce chapitre, nous présentons nos contributions pour l'extraction des règles de classification à partir des bases de données déséquilibrées bi-classes. Afin d'atteindre cet objectif, nous présentons dans la section 2 nos deux approches contributives en détaillant toutes les étapes nécessaires de construction d'un modèle qui représente les instances majoritaires et minoritaires. Pour cela, cette section est divisée en deux parties : la première partie décrit l'approche génétique pure USGA et la deuxième partie concerne l'approche mémétique USGA_ILS. Ceci est effectué, dans le but de répondre à la question suivante : que nous apporte l'approche hybride par rapport à l'approche génétique pure sur la qualité du modèle construit ? Quelle est l'utilité d'intégrer les méthodes de recherche locale dans les algorithmes génétiques ?

2. Nos contributions : le sous-échantillonnage par les algorithmes génétiques (USGA) et le sous-échantillonnage par les algorithmes mémétiques (USGA_ILS) :

Nos approches proposées consistent à extraire des règles de classification à partir des bases de données déséquilibrées bi-classes en utilisant les algorithmes évolutionnaires. Avant d'expliquer le principe de nos approches proposées, nous formulons d'abord le problème de la classification.

2.1. Formulation du problème de la classification:

Le problème de la classification est un problème combinatoire NP-difficile **[152]**; dont la complexité est expliquée comme suit :

Généralement, $B_n >$ exponentielle$^{(n)}$. Par conséquent, nous ne pouvons pas résoudre ce problème par une méthode exacte lorsque la taille du problème est grande. Nous le résolvons par l'utilisation d'une méthode heuristique ou métaheuristique.

Un modèle à base de règles peut être extrait par différentes façon telles que les approches approximatives (rough set approches) **[153]**, les approches floues (fuzzy set approches) **[159]** et les algorithmes de recouvrement séquentiels **[35] [59] [155]**. Nous nous focalisons sur ces derniers, puisque notre approche est basée sur eux.

Dans les deux sous sections suivantes, nous présentons la forme d'une règle de classification et nous décrivons les algorithmes de recouvrement séquentiels.

2.2. Représentation d'une règle de classification :

Dans les approches à base de règles, un modèle est construit à partir d'une base de données d'apprentissage en utilisant un algorithme d'apprentissage. Le modèle construit est représenté par un ensemble de règles de classification.

Une règle de classification a la forme suivante :

Si (Condition 1) ⋀ (Condition 2) ⋀ … (Condition n) alors Classe

Où:

- n est inférieur ou égal au nombre d'attributs prédictifs dans une base de données.
- La ième condition est de la forme : **Attribut$_i$ opérateur valeur.** Où :
  - Le champ opérateur dépend du type de l'attribut$_i$:
    - Si le type de l'attribut est nominal alors l'opérateur prend "=".
    - Si le type de l'attribut est numérique alors l'opérateur prend "≥" ou "≤".
  - Le champ valeur prend une valeur parmi les valeurs possibles de l'attribut$_i$.
- Classe est de la forme : Attribut_classe= valeur

Exemple 4.1:

Soit la base de données météo (weather) de l'UCI **[156]**. Elle contient 03 attributs nominaux et deux attributs numériques : Temps = {ensoleillé, couvert, pluvieux} ; Vent = {vrai, faux} ; Températureϵ [64, 85], humidité ϵ [65, 96]. Le dernier attribut classe : Jouer = {oui, non}

La règle de la classification pourrait être, par exemple :

Si temps=ensoleillé et température ≤ 70 et vent= faux alors jouer=oui.

2.3. Les algorithmes de recouvrement séquentiels :

Les algorithmes de recouvrement séquentiels ont été d'abord utilisés par les algorithmes de la famille AQ **[157] [158]** à la fin des années 1960. Mais, au fil des années, ils ont été appliqués de manière exhaustive comme l'algorithme de base dans les systèmes d'induction des règles.

Le principe de fonctionnement d'un algorithme de recouvrement séquentiel est le suivant : il construit une règle à partir de la base de données d'apprentissage, ensuite il supprime toutes les instances couvertes par cette règle et récursivement, il construit une autre règle et il supprime les instances couvertes par cette dernière. Le processus d'apprentissage se poursuit jusqu'à ce qu'un critère prédéfini soit satisfait. Ce critère exige généralement que toutes ou presque toutes les instances de la base d'apprentissage soient couvertes. Les étapes de ces algorithmes sont résumées dans l'***algorithme 4.1***.

Algorithme 4.1. L'algorithme de recouvrement séquentiel

Entrée : la base de données d'apprentissage S contenant m instances.
Sortie : le modèle C.
Début
C :={} ;
Tant que (m>0) ***faire***
Règle R := ***Algorithme d'extraction*** (S, m) ;
Supprimer toutes les instances couvertes par R ;
m:=m- le nombre d'instances supprimées ;
C :=C ∪ R ;
fait ;
Fin.

2.4. Notre Algorithme d'extraction d'une règle de classification:

Comme le problème de la classification est un problème NP-difficile, nous faisons appel aux algorithmes génétiques. Ces derniers ont montré leur efficacité et leur robustesse pour résoudre des problèmes combinatoires difficiles.

En général, la motivation principale d'utilisation des algorithmes génétiques est la découverte de règles à haute prédiction. Ceci est dû à une recherche globale dans l'espace de recherche et font mieux face à l'interaction d'attributs que les algorithmes d'induction de règles gourmandes souvent utilisés dans l'exploration de données **[159]**.

Le fonctionnement de notre algorithme génétique est résumé dans l'***algorithme 4.2***.

Algorithme 4.2. Fonctionnement de l'algorithme génétique

Début
Initialiser les paramètres de l'algorithme génétique :
- Probabilité du croisement Pc.
- Probabilité de la mutation Pm.
- Taille de la population T.
- Nombre de génération Gen_Max.

Initialiser la population initiale.
Evaluer chaque individu de la population.
Pour i:=1 ***à*** Gen_Max ***faire***
Appliquer le cycle génétique.
Fait ;
Retourner le meilleur individu de la population.
Fin.

Les points importants de l'algorithme génétique sont : le codage, l'initialisation de la population initiale, l'évaluation de la population et le cycle génétique. Dans ce qui suit, nous détaillons chaque point.

2.4.1. Codage des règles de classification:

Nous utilisons l'approche de Michigan où chaque chromosome de la population représente une seule règle de classification. Nous appliquons le codage numérique pour coder chaque règle **[160]**. Donc, un chromosome est composé de deux parties. La première partie représente la partie condition et la deuxième partie représente Le conséquent. La ***figure 4.1*** illustre cette représentation.

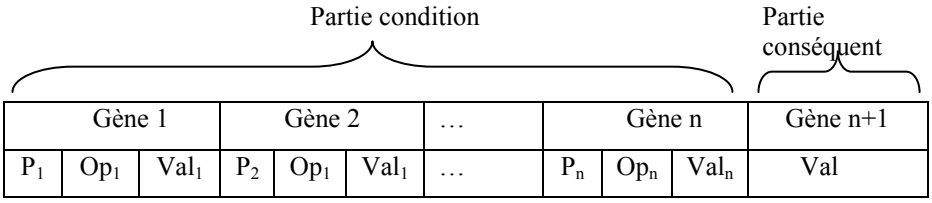

Figure 4.1. ***Représentation d'un chromosome***

La première partie est composée de n gènes, où chaque gène représente une seule condition. L'ordre de gènes suit l'ordre des attributs dans la base de données. Chaque gène est composé de 03 allèles (***voir la figure 4.1***) :

- L'allèle poids (P) prend ses valeurs dans l'ensemble {0, 1}. Cet allèle indique si la condition correspondante est présente ou non dans la règle. Plus précisément, lorsque P est égal à 0, la condition correspondante est effectivement supprimée de la règle.
- L'allèle opérateur (Op) : le codage de l'ensemble d'opérateurs {=, ≤, ≥} est {0, 1, 2} respectivement.
- L'allèle valeur (Val) : sa valeur dépend du type de l'attribut :
  - Si l'attribut est numérique alors Val prend une valeur parmi les valeurs possibles de l'attribut.
  - Si l'attribut est nominal, alors le codage numérique est appliqué comme suit : étant donné un attribut qui prend trois valeurs $\{V_1, V_2, V_3\}$. Ces valeurs sont converties en nombres entiers par énumération, donc l'ensemble convertit devient : {0, 1, 2}.

La deuxième partie est représentée par un seul gène composé d'un seul allèle qui représente l'allèle valeur (Val) de l'attribut classe.

Donc, la taille d'un chromosome L est calculée en fonction du nombre d'allèles (définie dans l'équation (4.1)) :

$$\mathbf{L = (Nombre\ d'attributs\ prédictifs * 3) + 1} \qquad \mathbf{(4.1)}$$

Exemple 4.2:

Les attributs nominaux de la base de données météo utilisée dans l'***exemple 4.1*** sont codés comme suit :

Le codage des attributs nominaux temps, vent et jouer est respectivement : {0, 1, 2}, {0, 1} et {0, 1}.

Donc, le codage de la règle donnée dans l'***exemple 4.1*** est illustré dans la ***figure 4.2***.

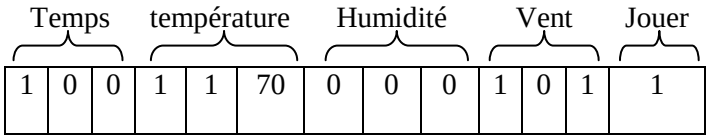

Figure 4.2. ***Exemple de codage d'une règle de classification***

2.4.2. Initialisation de la population initiale :

La population initiale est générée de façon aléatoire mais avec un contrôle. Cela veut dire que nous devons vérifier à chaque fois que le chromosome généré représente une règle valide (contient au moins une condition).

2.4.3. Evaluation d'un individu :

Les fonctions d'évaluation utilisées vont être données dans la section ***2.5.2.***

2.4.4. Le cycle génétique à l'état stationnaire ssGA:

Nous utilisons l'algorithme génétique à l'état stationnaire ssGA (**Steady State Genetic algorithm) [161]** décrit dans l'***algorithme 4.3***. Il fonctionne de la manière suivante : à chaque génération, deux individus sont sélectionnés d'abord. Puis ils se croisent pour produire deux enfants. Ensuite, le meilleur enfant est muté. Enfin, ce dernier remplace le mauvais individu de la population.

Algorithme 4. 3. Le cycle génétique à l'état stationnaire ssGA

Début

1. Sélection :
Sélectionner deux parents I1 et I2;

2. Croisement en un point:
Générer une valeur aléatoire R ;
Si R≤ Pc ***alors***
Choisir le point de coupure aléatoirement ;
Appliquer le croisement sur I1 et I2 pour obtenir deux enfants mais garder un seul enfant Off qui est le meilleur ;
Sinon
Offs := sélectionner aléatoirement I1 ou I2 ;

3. Mutation :
Pour chaque gène i de l'individu Off ***faire***
Générer une valeur aléatoire R ;
Si R≤Pm ***alors***
Muter le gène i ;
Fait ;
Evaluer Off;

4. Remplacement :
Remplacer le mauvais individu de la population par Off ;

Fin.

Dans ce qui suit, nous détaillons chaque étape du cycle génétique ssGA et nous montrons à chaque fois les modifications apportées.

a. Sélection:

Un individu est sélectionné par la méthode du Tournoi ***(Tournament Selection)*** qui fonctionne comme suit : elle sélectionne deux individus aléatoirement, puis elle les évalue pour prendre le meilleur. Dans ssGA, deux individus sont sélectionnés pour se reproduire. A noter, que ces deux individus peuvent être identiques. Dans notre cas, nous avons imposé que ces deux individus doivent être différents.

b. Croisement :

Le croisement est utilisé pour diversifier la recherche. SsGA utilise le croisement en un point. Le point de coupure est choisi aléatoirement. Si la probabilité du croisement est vérifiée, alors les gènes de deux individus sélectionnés sont échangés pour produire un seul enfant seulement, qui contient une partie du premier individu et une partie du deuxième individu. Sinon, le meilleur individu est retourné.

Dans notre cas, deux individus sont produits, puis, nous choisissons le meilleur. Donc, deux règles se croisent pour produire deux autres règles en échangeant les conditions seulement. Pour cette raison que le point de coupure doit être multiple de 3 pour avoir des règles cohérentes.

Exemple 4.3:

Soient les deux règles suivantes :

- R1 : si temps=ensoleillé et température ≤ 70 et vent= faux alors jouer=non
- R2 : si humidité ≥ 65 et vent=faux alors jouer=non

Le croisement appliqué sur les règles R1 et R2 résulte les règles R3 et R4 (tel qu'il est illustré dans la ***figure 4.3*** où le point de coupure est égal à 6).

- R3 : si temps=ensoleillé et température ≤ 70 et humidité ≥ 65 et vent=faux alors jouer=non
- R4 : si vent=faux alors jouer=non.

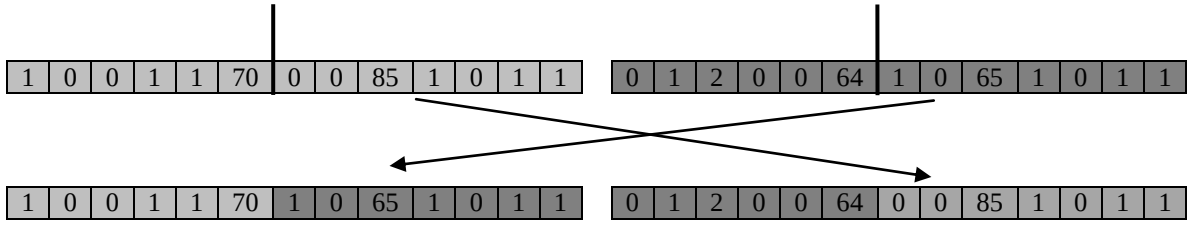

Figure 4.3. ***Exemple du croisement***

c. Mutation :

Consiste à changer chaque allèle d'un individu si la probabilité de mutation est vérifiée. A noter que le gène qui représente la classe n'est pas concerné par la mutation. L'***algorithme 4.4*** montre les étapes de ce processus.

Algorithme 4.4. Mutation

Entrée : la probabilité de mutation pm.
l'individu I de taille L
Sortie : l'individu muté I
Début
Pour j :=0 ***à*** L-2 ***faire*** *//pour chaque allèle de l'individu, excepté l'allèle classe*
Générer une valeur aléatoire R ;
Si R ≤ pm ***alors***
Si j modulo 3=0 ***alors****//Allèle poids*
Si I[j]=0 ***alors***
I[j] :=1 ; *//Activer la condition*
s***inon***
I[j] :=0 ; *//Désactiver la condition*
j :=j+3 ;
fin ;
fin ;
Si j modulo 3=1 ***alors****//Allèle opérateur*
Si l'attribut correspondant est numérique alors
Si I[j]=0 ***alors****// l'opérateur est* ≥
I[j] :=1 ; *//l'opérateur devient* ≤
Sinon*//l'opérateur* ≤
I[j] :=0 ; *//l'opérateur devient* ≥
fin ;
Sinon
Ne pas toucher cet allèle
fin ;
fin ;
Si j modulo 3=2 ***alors*** *//Allèle valeur*
Choisir la valeur V parmi l'ensemble de valeurs possibles de l'attribut correspondant, à condition que la valeur V soit différente de I[j] ;
I[j] :=V ;
fin ;
fin ;
fait ;
Fin.

Exemple 4.4 :

Soit la règle suivante :

- R : Si temps=ensoleillé et température ≤ 70 et vent=faux alors jouer=oui.

La règle R mutée est la suivante (***voir la figure 4.4***):

Si température ≥ 70 et humidité ≥ 65 et vent=faux alors jouer=oui.

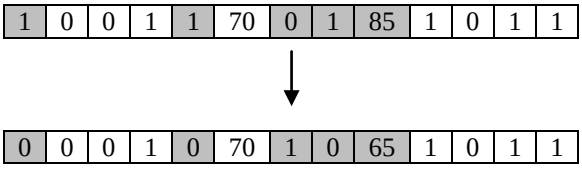

Figure 4.4. ***Exemple de mutation***

d. Remplacement :

Dans ssGA, le mauvais individu de la population est remplacé par l'individu produit par la mutation. Mais, ce dernier peut être moins performant que le mauvais. Donc, dans notre cas, nous appliquons le remplacement si le mauvais individu de la population est moins performant que l'individu produit.

2.5. Partie 1 : L'approche génétique proposée USGA

2.5.1. Principe de l'approche USGA:

Notre approche applique le sous-échantillonnage guidé par les règles de classification en utilisant les algorithmes génétiques, d'où le nom que nous lui avons donnée : USGA (UnderSampling by Genetic Algorithm) **[162]**.

USGA consiste à appliquer une méthode intelligente pour l'extraction de règles de classification pour les bases de données déséquilibrées bi-classes à trois phases illustrées dans la ***figure 4.5***.

- Premièrement, cette approche vise à équilibrer la base de données déséquilibrée par la construction d'un premier modèle contenant des règles de classification représentant les instances majoritaires seulement. Ensuite, elle supprime toutes les instances majoritaires qui sont bien classées par ces règles. Par conséquent, notre méthode proposée évite la perte des concepts contenus dans les instances majoritaires supprimées parce qu'elles sont remplacées par des règles de classification.
- Deuxièmement, USGA utilise la base de données équilibrée afin d'extraire le deuxième modèle contenant des règles de classification représentant les instances majoritaires et minoritaires.
- Enfin, les deux modèles extraits sont fusionnés pour obtenir le troisième modèle qui va subir un post-traitement pour supprimer les règles de classification redondantes.

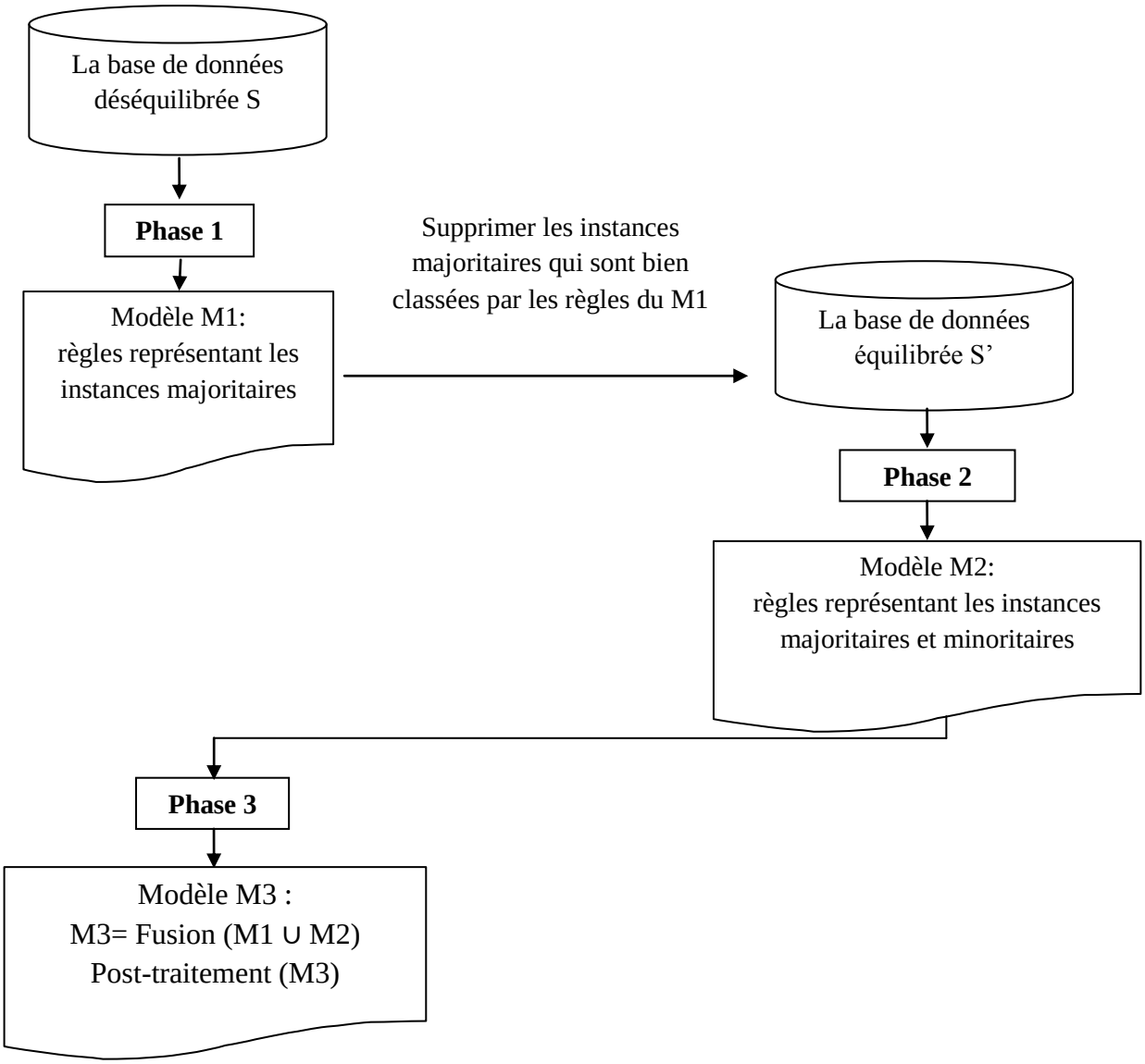

Figure 4.5. ***Les étapes de l'approche proposée USGA***

2.5.2. Les phases de l'approche USGA:

Nous détaillons chaque phase de l'approche USGA dans les sous sections suivantes :

a. Phase 1 : Altération de la distribution des données

L'objectif de notre approche est l'extraction de règles de classification à partir des bases de données déséquilibrées bi-classes. Comme nous avons déjà expliqué dans la problématique, les instances minoritaires sont ignorées parce que leur nombre est plus petit par rapport au nombre d'instances majoritaires. Pour cette raison, nous avons fait appel à la première phase pour diminuer le degré du déséquilibre. Les détails de cette phase sont donnés dans les sections suivantes :

a.1. L'algorithme d'extraction du premier modèle:

Dans cette phase, nous développons un algorithme d'apprentissage basé sur les algorithmes génétiques dont le but d'extraire le premier modèle M1. Les règles du M1 couvrent les instances majoritaires seulement. Par conséquent, les instances majoritaires qui sont bien classées par les règles du M1 sont supprimées. Donc, notre approche consiste à équilibrer la base de données déséquilibrée tout en évitant la perte des concepts contenus dans les instances majoritaires supprimées parce qu'elles sont représentées par des règles de classification du modèle M1. Le nombre d'instances majoritaires supprimées dépend de la valeur du ratio du déséquilibre IR. Ce processus est répété plusieurs fois jusqu'à ce que la valeur du ratio du déséquilibre IR soit égale à 1.

Comme la base de données d'apprentissage contient des instances minoritaires et majoritaires, nous pouvons trouver des règles qui ne peuvent pas couvrir les instances majoritaires seulement. Donc, l'algorithme de recouvrement séquentiel va boucler infiniment. Pour résoudre ce problème, nous avons rajouté une autre condition pour éviter la boucle infinie. Cette condition concerne le nombre d'itérations, c'est-à-dire après 10 itérations, si nous ne pouvons pas trouver une règle qui couvre les instances majoritaires seulement alors nous la rajouterons.

Nous avons choisit la valeur 10 aléatoirement car nous n'avons pas pu mettre ce nombre comme un paramètre à déduire expérimentalement pour la raison suivante : lorsque nous avons fait des tests sur quelques bases de données, nous n'avons pas trouvé une règle qui couvre les instances majoritaires et minoritaires à la fois. Donc, afin d'éviter la boucle infinie (dans le cas où nous testerons notre approche sur d'autres bases de données), nous avons rajouté cette condition par mesure de sécurité.

Les détails de cette phase sont résumés dans l'***algorithme 4.5***.

Algorithme 4.5. Algorithme d'apprentissage et de sous-échantillonnage

Entrée: la base de données déséquilibrée S.
Sortie: modèle M1 et la base de données équilibrée S.
Début
M1:={};
Min:= nombre d'instances minoritaires;
Maj:= nombre d'instances majoritaires;
IR:=Maj/Min ;
Tant que (IR>1) ***faire***
It:=1 ;
Répéter
Règle R:= Algorithme génétique (S);
It++ ;
Jusqu'à (R couvre les instances majoritaires seulement ou It=10) ;
M1:=M1 ∪ R;
Tant que (IR>1) ***faire***
S:=S – {une instance majoritaire bien classée par R} ;
Maj--;
IR:=Maj/Min;
fait ;
fait;
Fin.

L'algorithme génétique est utilisé pour trouver la meilleure règle.

En conclusion, le modèle M1 contient les règles qui ont la classe majoritaire. Ces règles servent à supprimer les instances majoritaires seulement qui sont bien classées par ces règles. Par conséquent, nous avons utilisé un nouveau type d'échantillonnage guidé par les règles de classification pour obtenir une base de données équilibrée.

a.2. La fonction d'évaluation utilisée dans la phase 1:

L'objectif de cette phase est l'extraction de règles de classification représentant les instances majoritaires seulement. Donc, la meilleure règle est celle qui classe un nombre important d'instances majoritaires. Par conséquent, la mesure utilisée pour évaluer la qualité d'une règle R est TNrate (définie dans ***le chapitre 3, section 4.1.3***). Nous rappelons que TNrate représente le pourcentage d'instances majoritaires bien classées par une règle (voir l'équation (4. 2))

$$\boldsymbol{Fitness(R) = TNrate = \frac{TN}{TN + FP}} \qquad \textbf{(4. 2)}$$

b. Phase 2: Extraction du deuxième modèle

Dans cette phase, la même procédure est appliquée sur la base de données équilibrée ***D*** en utilisant la validation croisée à K partitions (**K-Fold cross-validation) [163]** pour construire le modèle M2. Dans la validation croisée à K partitions, la base de données initiale est

partitionnée de façon aléatoire en K sous bases de données indépendantes D_1, D_2, ..., D_K, chacune de taille approximativement égale. Chaque sous base de données est connue sous le nom « partition» (fold). Pour chaque partition D_i, une phase d'apprentissage est réalisée sur la base d'apprentissage B_i contenant les (K-1) partitions restantes et une phase de test est réalisée sur la partition D_i.

Pour notre problème, nous n'avons pas partitionné la base de données équilibrée aléatoirement pour ne pas avoir des partitions déséquilibrées. Donc, une partition contient des instances majoritaires et minoritaires.

b.1. L'algorithme d'extraction du deuxième modèle :

Le modèle M2 est aussi construit par l'algorithme de recouvrement séquentiel en utilisant les algorithmes génétiques. L'algorithme d'apprentissage sur la base d'apprentissage B_i est donné dans l'***algorithme 4.6***.

Algorithme 4.6. Algorithme d'apprentissage

Entrée: la base d'apprentissage B_i.
Sortie: le modèle M2.
Début
M2:= {};
T := nombre d'instance de la partition B_i;
Tant que (T>0) ***faire***
Règle R: = Algorithme génétique(B_i);
Supprimer les instances couvertes par R :
B_i:=B_i-{les instances couvertes par R};
T :=T- nombre d'instances supprimées;
M2:=M2∪{R};
Fait;
Fin.

b.2. La fonction d'évaluation utilisée dans la phase 2:

L'objectif de cette phase est l'extraction de règles de classification qui représentent les instances majoritaires et minoritaires. Donc, la règle la plus performante est celle qui classe un nombre important d'instances. Par conséquent, la mesure la plus appropriée est l'exactitude (définie dans ***le chapitre 2, section 5.4.2***). Nous rappelons que l'exactitude d'une règle R est le pourcentage d'instances de la base de données qui sont bien classées par cette règle (voir l'équation 4.3).

$$\boldsymbol{Fitness(R) = Exactitude = \frac{Nombre\ d'instances\ bien\ classées\ par\ R}{Nombre\ d'instances\ couvertes\ par\ R}} \qquad \textbf{(4.3)}$$

c. Phase 3 : Fusion et post-traitement

Dans cette phase, nous fusionnons d'abord M1 et M2 pour obtenir le nouveau modèle M3, puis nous procédons à un post-traitement du M3 pour supprimer les règles redondantes si elles existent. Une règle est considérée comme redondante si elle est spécifique ou contradictoire.

c.1. Suppression des règles spécifiques:

Soient R1 et R2 deux règles qui ont la même classe. Si la partie condition de R2 est incluse dans la partie condition de R1, alors R2 est considérée comme une règle générale et R1 est considérée comme une règle spécifique. Donc, la règle spécifique R1 va être supprimée.

***Exemple 4.5*:**

Soient R1 et R2 :

- R1: Si temps=ensoleillé et température ≤ 85 et vent=vrai ALORS jouer= non.
- R2: Si temps=ensoleillé ALORS jouer=non.

Nous supprimons R1 qui est la plus spécifique et nous gardons R2 qui est la plus générale.

c.2. Traitement des règles contradictoires :

Deux règles sont considérées comme contradictoires si elles ne possèdent pas la même classe mais elles ont la même partie condition ou la partie condition de l'une est incluse dans la partie condition de l'autre.

***Exemple 4.6*:**

Les deux règles suivantes R1 et R2 sont contradictoires:

- R1: Si temps=ensoleillé ET température ≤ 70 et vent=faux ALORS jouer=Oui
- R2: Si temps=ensoleillé ET vent=faux ALORS jouer=Non.

Dans la littérature, il n'existe pas une méthode qui traite les règles contradictoires. Dans notre approche, nous avons gardé la règle la plus performante, c'est-à-dire la règle qui optimise les mesures de performances utilisées pour l'évaluer.

2.6. Partie 2 : L'approche mémétique proposée USGA_ILS

2.6.1. Principe de l'approche USGA_ILS :

La deuxième approche proposée USGA_ILS a le même principe que l'approche USGA sauf que nous faisons appel aux algorithmes mémétiques dans la deuxième phase **[164]**.

USGA_ILS résulte de l'intégration de la méthode de recherche locale itérée ILS ***(vue dans le chapitre 1, section 2.1.2)*** dans le cycle génétique ssGA pour obtenir le cycle mémétique ssGA_ILS. Donc, nous appelons notre approche USGA_ILS.

Nous rappelons que la méthode de recherche locale itérée ILS contient les modules suivants :

1) Génération de la solution initiale.

2) Recherche locale.
3) Perturbation
4) Critère d'acceptation d'une solution.
5) Le critère d'arrêt.

Dans ce qui suit, nous allons voir les modifications apportées sur chaque module de la méthode ILS afin de l'adapter au problème de la classification.

2.6.2. Adaptation de la méthode ILS au problème de la classification :

a. Génération de la solution initiale :

Dans notre approche, nous ne faisons pas appel à une méthode de génération de la solution initiale, mais elle est obtenue par l'une des deux possibilités suivantes :

1. Le meilleur individu trouvé par le croisement.
2. L'individu trouvé par la mutation.

b. La méthode de recherche locale utilisée:

Nous avons utilisé une méthode de recherche locale simple qui ne permet pas de s'échapper des optimums locaux.

Notre méthode consiste à créer un seul voisin ***x'*** d'un individu ***x*** qui se diffère par un seul gène choisi aléatoirement et plus précisément par un seul allèle. Donc, les deux individus ***x*** et ***x'*** se diffèrent par une seule condition seulement. Donc, le voisin v' de la solution v est généré selon l'***algorithme 4.7***.

Algorithme 4.7. La méthode de recherche locale utilisée

Entrée : l'individu ***x***;
Sortie : l'individu voisin ***x'***.
Début
Choisir aléatoirement le numéro de gène i à muter.
Si le poids correspondant à ce gène est 0 ***alors****//la ième condition est manquante*
le muter à 1 *//la ième condition devient présente*
sinon
Choisir aléatoirement l'allèle j à muter : *//l'allèle j correspondante au gène i*
Si j correspond à l'allèle poids ***alors***
Le muter à 0 *//la ième condition devient manquante*
Si j correspond à l'allèle valeur ***alors***
Le muter par le choix aléatoire d'une autre valeur appartenant à l'ensemble des valeurs possibles de l'attribut i
Si j correspond à l'allèle opérateur ***alors***
Si l'attribut correspondant au gène i est numérique ***alors***
Choisir un opérateur aléatoirement *//≤ ou ≥*
Si l'attribut correspond au gène i est nominal ***alors***
Choisir aléatoirement l'allèle poids ou l'allèle valeur.
Fin.

Exemple 4.7 :

Prenons le chromosome donné dans l'exemple 4. 2 (***figure 4.2***). Les voisins qui peuvent être générés à partir du gène 2 sont donnés dans la ***figure 4.6***.

Modification de Gène 2

1	0	0	1	1	70	0	0	0	1	0	1	1
1	0	0	0	0	0	0	0	0	1	0	1	1
1	0	0	1	0	70	0	0	0	1	0	1	1
1	0	0	1	1	85	0	0	0	1	0	1	1
1	0	0	1	1	64	0	0	0	1	0	1	1
1	0	0	1	1	75	0	0	0	1	0	1	1
. . .												

Figure 4.6. ***Exemple d'une structure de voisinage générée par la recherche locale***

c. La perturbation :

Comme la perturbation est basée sur l'utilisation d'un voisinage plus large que la recherche locale, alors le nombre de gènes mutés est supérieur ou égal à 1 et il dépend du degré de perturbation. Par conséquent, la taille de l'espace de recherche dans ce cas est plus grande que la taille de l'espace de recherche dans le cas de la recherche locale.

Le degré de perturbation est en fonction du nombre d'attributs dans une base de données. Dans l'étude expérimentale, nous allons essayer les degrés suivants:

- Degré= nombre d'attributs/4 (25%).
- Degré=nombre d'attributs/2 (50%).
- Degré=3*nombre d'attributs/4 (75%).

Exemple 4.8 :

Prenons le chromosome donné dans l'exemple 4.3 (***figure 4.2***). Les voisins qui peuvent être générés par la mutation de deux premiers gènes sont donnés dans la ***figure 4.7***.

Modification des gènes 1 et 2

1	0	0	1	1	70	0	0	0	1	0	1	1
0	0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	1	0	70	0	0	0	1	0	1	1
1	0	1	1	1	85	0	0	0	1	0	1	1
1	0	1	1	1	64	0	0	0	1	0	1	1
1	0	1	1	1	75	0	0	0	1	0	1	1
. . .												

Figure 4.7. ***Exemple d'une structure de voisinage générée par la perturbation***

d. Le critère d'acceptation d'une solution:

Nous allons tester les 02 critères existants : le meilleur critère d'acceptation et la marche aléatoire sachant que le premier critère favorise l'intensification et le deuxième critère favorise la diversification. Afin d'équilibrer l'intensification et la diversification, nous proposons un autre critère que nous avons appelé le critère alternatif.

Ce critère est obtenu en alternant entre le meilleur critère d'acceptation et la marche aléatoire par la génération de la variable aléatoire X tel que :

$$X = \begin{cases} \mathbf{0}\ \boldsymbol{le\ meilleur\ crit\grave{e}re\ est\ utilis\acute{e}} \\ \mathbf{1}\ \boldsymbol{la\ marche\ al\acute{e}atoire\ est\ utilis\acute{e}} \end{cases} \qquad \textit{(4.4)}$$

e. Le critère d'arrêt de la méthode ILS:

Le critère d'arrêt utilisé est le nombre d'itérations. Ce nombre va être déduit par l'étude expérimentale où nous allons tester différentes valeurs pour voir ses influences sur les performances du modèle produit.

2.6.3. Les stratégies d'intégration de la méthode ILS :

La méthode ILS va être intégrée dans le cycle génétique ssGA. Nous allons utiliser les deux stratégies suivantes:

- **USGA_ILS (C) :** ILS est intégrée après le croisement.
- **USGA_ILS (M) :** ILS est intégrée après la mutation.

3. Conclusion :

Nous avons vu dans ce chapitre les travaux de recherche que nous avons proposés pour l'extraction des règles de classification spécifiquement adaptées aux bases de données déséquilibrées bi-classes. Nous avons présenté la conception de deux approches, l'une est basée sur les algorithmes génétiques et l'autre est basée sur les algorithmes mémétiques.

Chapitre 5 :

ETUDES EXPERIMENTALES

1. Introduction:

Il convient à présent d'effectuer des tests afin d'évaluer l'efficacité et la robustesse de nos approches proposées. Pour cela, ces approchées ont été implémentées à l'aide du langage de programmation JAVA pour tester leurs performances, et de faire des études comparatives entre elles et quelques approches existantes dans la littérature.

2. Les outils utilisés:

2.1. Knowledge Extraction Evolutionary Learning KEEL:

KEEL **[165] [166]** est un outil logiciel (GPLv3) open source écrit en JAVA utilisé pour un grand nombre de tâches d'extraction de connaissances. Il fournit une interface graphique simple afin d'évaluer le comportement des algorithmes (Une attention particulière a été accordée aux algorithmes évolutionnaires). Il contient une grande variété d'algorithmes classiques d'extraction de connaissances, de techniques de prétraitement (sélection des bases de données pour l'apprentissage, sélection d'attributs, discrétisation, méthodes de traitement des valeurs manquantes, …), d'algorithmes d'apprentissage, des modèles hybrides et des tests statistiques. Nous avons utilisé KEEL pour obtenir les performances de quelques approches récentes utilisées dans les études comparatives.

2.2. Waikato Environment for Knowledge Analysis Weka:

Weka **[51]** est un logiciel écrit en JAVA développé à l'Université de Waikato en nouvelle Zélande. Weka est un ensemble d'outils permettant de manipuler et d'analyser des fichiers de données, implémentant la plupart des algorithmes d'intelligence artificielle, entre autres, les arbres de décision et les modèles à base de règles. Il se compose principalement :

- ✓ De classes JAVA permettant de charger et de manipuler les données.
- ✓ De classes qui implémentent quelques algorithmes de classification supervisée tels que J48, Id3, PART et OneR ou non supervisée, ce qui facilite l'étude comparative entre nos approches et quelques approches classiques.
- ✓ D'outils de sélection d'attributs et de statistiques sur ces attributs.
- ✓ De classes permettant de visualiser les résultats.

2.3. Les bases de données utilisées:

Nous avons utilisé 38 bases de données bi-classes de KEEL qui ont différentes valeurs du ratio du déséquilibre IR. Le ***tableau 5.1*** résume ces bases de données et montre pour chacune, le nombre d'instances (#Insts.), le nombre d'attributs (#Atts), le pourcentage de distribution d'instances sur la l'attribut classe (%Classe (min, maj)) et le ratio du déséquilibre (IR).

Tableau 5. 1. ***Caractéristiques des bases de données déséquilibrées utilisées***

Bases de données	#Insts.	#Atts.	%Classe(min, maj)	IR
Bases de données faiblement déséquilibrées (1.5≤IR≤3)				
Glass1	214	9	(35.51, 64.49)	1.82
Ecoli0vs1	220	7	(35.00, 65.00)	1.86
Wisconsin	683	9	(35.00, 65.00)	1.86
Pima	768	8	(34.84, 66.16)	1.90
Iris0	150	4	(33.33, 66.67)	2.00
Glass0	214	9	(32.71, 67.29)	2.06
Yeast1	1484	8	(28.91, 71.09)	2.46
Vehicle2	846	18	(28.37, 71.63)	2.52
Vehicle3	846	18	(28.37, 71.63)	2.52
Haberman	306	3	(27.42, 73.58)	2.68
Vehicle1	846	18	(25.65, 74.34)	2.89
Bases de données moyennement déséquilibrées (3≤IR≤9)				
Glass0123vs456	214	9	(23.83, 76.17)	3.19
Vehicle0	846	18	(23.64, 76.36)	3.23
Ecoli1	336	7	(22.92, 77.08)	3.36
New-thyroid2	215	5	(16.89, 83.11)	4.92
New-thyroid1	215	5	(16.28, 83.72)	5.14
Ecoli2	336	7	(15.48, 84.52)	5.46
Glass6	214	9	(13.55, 86.45)	6.38
Yeast3	1484	8	(10.98, 89.02)	8.11
Ecoli3	336	7	(10.88, 89.12)	8.19
Page-blocks0	5472	10	(10.23, 89.77)	8.77
Yeast2vs4	514	8	(09.92, 90.08)	9.08
Yeast05679vs4	528	8	(09.66, 90.34)	9.35
Bases de données fortement déséquilibrées (IR>9)				
Vowel0	988	13	(09.01, 90.99)	10.10
Glass016vs2	192	9	(08.89, 91.11)	10.29
Glass2	214	9	(08.78, 91.22)	10.39
Ecoli4	336	7	(06.74, 93.26)	13.84
Yeast1vs7	459	8	(06.72, 93.28)	13.87
Shuttle0vs4	1829	9	(06.72, 93.28)	13.87
Glass4	214	9	(06.07, 93.93)	15.47
Abalone9vs18	731	8	(05.65, 49.25)	16.68
Yeast1458vs7	693	8	(04.33, 95.67)	22.10
Yeast2vs8	482	8	(04.15, 95.85)	23.10
Yeast4	1484	8	(03.43, 96.57)	28.41
Yeast1289vs7	947	8	(03.17, 96.83)	30.56
Yeast5	1484	8	(02.96, 97.04)	32.78
Yeast6	1484	8	(02.49, 97.51)	39.15
Abalone19	4174	8	(00.77, 99.23)	128.87

2.4. Les approches utilisées dans l'étude comparative:

Nous avons comparé nos résultats avec des approches classiques et des approches récentes.

2.4.1. Les approches classiques:

Les approches classiques utilisées sont les suivantes :

2.4.1.1. C4.5 : décrit dans le ***chapitre 2, section 6.1.2***. Nous l'avons choisi car il figure parmi les 10 meilleurs algorithmes dans le Data Mining **[167]** et il va être utilisé comme un algorithme de base pour les approches récentes utilisées dans l'étude comparative.

2.4.1.2. RIPPER:

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) **[168]** est une approche d'apprentissage supervisée. Elle construit un ensemble de règles pour chaque classe puis elle les concatène.

2.4.1.3. Les arbres de décision partiels PART:

PART est aussi une approche d'apprentissage **[169]** supervisée. Elle adopte aussi la stratégie diviser et conquérir (divide-and-conquer) de RIPPER et la combine avec C4.5.

2.4.2. Les approches récentes:

Nous avons utilisé quelques approches récentes décrites dans les ***chapitres 3 et 4***. Le ***tableau 5.2*** contient les paramètres utilisés par ces approches ainsi que les abréviations utilisées dans ce chapitre. Comme ces approches utilisent C4.5 comme un algorithme de base donc nous l'avons rajouté dans le ***tableau 5.2***.

Tableau 5. 2. ***Paramètres des approches récentes et les abréviations associées***

Algorithme	Abréviation	Paramètres
C4.5	C4.5	Niveau de confiance=0.25 Le nombre minimal des items par nœud:2 Elagage après construction: vrai
SMOTEBoost	SMB	Nombre de voisins k=5 Distance utilisée : métrique de différences de valeurs hétérogènes(HVDM) **[170] [171]** α=0.5
UnderBagging1	UB1	Nombre de modèles=10
AdaC2	AC2	Coût de la classe majoritaire=0.25 Coût de la classe minoritaire=1 Nombre de modèles=10
Sous échantillonnage évolutionnaire d'équilibrage (EBUS-MS) avec CHC	CHC	Taille de la population=50 Nombre d'évaluations=10 000

2.5. Les tests statistiques:

Pour comparer les performances de nos approches proposées avec d'autres approches, nous avons utilisé les tests non paramétriques parce que leur utilisation est fortement recommandée **[172]**. Il existe différentes méthodes de tests statistiques. Dans ce chapitre, nous avons utilisé la méthode de test de rang signé apparié de Wilcoxon(Wilcoxon paired signed rank test) et la

méthode de test de rangs alignés de Freidman (Freidman Aligned-ranks test).Ces deux méthodes sont décrites dans les deux sous sections suivantes :

2.5.1. Test de rang signé apparié de Wilcoxon :

Cette méthode **[173]** est utilisée pour détecter la différence entre deux approches seulement. Ses étapes sont les suivantes :

- Calculer la différence de performance d_i entre deux approches de la $i^{ème}$ base de données parmi n bases de données.
- Associer un rang à la valeur absolue de cette différence.
- Calculer R^+ et R^- : R^+ est la somme des rangs pour les bases de données dont la différence est positive (dans lesquelles la seconde approche est moins performante) et R^- la somme des rangs dans le cas contraire. R^+ et R^- sont définis dans les équations (5.1) et (5. 2) respectivement.

$$\boldsymbol{R^+ = \sum_{d_i>0} rang(d_i) + \frac{1}{2}\sum_{d_i=0} rang(d_i)} \quad \textbf{(5. 1)}$$

$$\boldsymbol{R^- = \sum_{d_i<0} rang(d_i) + \frac{1}{2}\sum_{d_i=0} rang(d_i)} \quad \textbf{(5. 2)}$$

- Déterminer la valeur du test statistique T : $T=\min(R^+, R^-)$.
- Mettre l'hypothèse de test H_0: il n y'a pas de différence entre deux échantillons, dans notre cas entre deux approches.
- Comparer les performances des deux approches en prenant en considération la valeur de T et la valeur critique :

➢ Si T est inférieure à la valeur critique, alors l'hypothèse de test H_0 est rejetée et nous concluons que les deux approches sont significativement différentes, c'est-à-dire les performances de la première approche sont meilleures que celles de la deuxième approche.

➢ Si T est supérieure à la valeur critique alors les approches ne sont pas significativement différentes, c'est-à-dire les deux approches ont les mêmes performances.

Remarque :

Les valeurs critiques sont obtenues à partir du tableau B.12 dans **[174]**.

2.5.2. Test de rangs alignés de Freidman:

Cette méthode **[175] [176]** est utilisée pour comparer plusieurs approches. Dans nos études comparatives, nous avons utilisé la moyenne des rangs de chaque approche en tant qu'un outil de visualisation, afin de comparer à première vue le comportement de chaque approche par rapport aux autres.

Pour visualiser la différence entre k approches sur n bases de données, nous considérons la moyenne des rangs de chaque approche, qui est obtenue comme suit :

- Associer des rangs aux résultats : soit r_i^j le rang associé à l'approche j pour la ième base de données. Le rang 1 est associé à la meilleure approche et le rang k est associé à la mauvaise approche.
- Calculer la moyenne des rangs pour chaque approche pour voir sa performance globale.
- Déterminer la meilleure approche: une approche est la meilleure si la moyenne de ses rangs est la plus faible.

3. La méthodologie d'obtention des résultats :

Les résultats de nos approches ont été obtenus comme suit: appeler d'abord la phase 1 de l'approche USGA (ou USGA_ILS) sur la base de données déséquilibrées S pour obtenir la base de données équilibrée D et le premier modèle M1. Ensuite, faire appel à la validation croisée à 5 partitions sur D ***(expliquée dans le chapitre 4, section 2.5.2)*** dans la phase 2. Donc, 4 partitions sont utilisées pour l'apprentissage (c'est-à-dire 80% de la base de données D) et une partition pour le test (c'est-à-dire 20% de la base de données D). Par conséquent, l'algorithme d'apprentissage s'exécute 5 fois sur 5 bases d'apprentissage différentes où chaque exécution produit le deuxième modèle M2. Après, rajouter les règles du M1 au M2 pour obtenir le troisième modèle M3. Puis, appliquer une méthode de post-traitement sur M3 pour tester ses performances sur la partition concernée. Enfin, les performances finales sont égales à la moyenne des 5 performances. Par exemple, si la mesure de performance utilisée est G-Mean ***(vue dans le chapitre 3, section 4.2.1)*** alors G-Mean finale est calculée comme suit :

$$\boldsymbol{G-Mean} = \frac{\sum_{i=1}^{5} \boldsymbol{G-Mean}\ (\boldsymbol{partition_i})}{\mathbf{5}} \qquad \textit{(5. 3)}$$

Nous notons que les meilleurs résultats pour chaque base de données sont mis en gras.

4. Ajustement des paramètres de l'algorithme génétique :

L'efficacité des algorithmes génétiques est liée fortement au choix judicieux de ses paramètres. Afin de sélectionner la combinaison des paramètres qui produit de bonnes

performances, nous avons fixé la probabilité du croisement ainsi que la probabilité de mutation respectivement à 0.8 et 0.1. Ensuite, nous avons fixé le nombre de générations à 10, 50 et 100, nous avons affecté pour chaque valeur, les valeurs 20, 50, 70 et 100 à la taille de la population. Les résultats obtenus sont illustrés dans le ***tableau 5.3***.

Tableau 5.3. ***G-Mean obtenu pour différentes valeurs de générations et différentes valeurs de la taille de population***

Nombre de générations	**10**				**100**				**50**			
Taille de la population	**20**	**50**	**70**	**100**	**20**	**50**	**70**	**100**	**20**	**50**	**70**	**100**
iris0	**0.950**	0.940	**0.950**	0.940	**0.950**	0.903	**0.940**	**0.950**	**0.950**	**0.950**	**0.950**	**0.950**
Glass016vs2	**0.840**	0.789	0.734	0.789	0.766	0.705	0.804	0.789	0.720	0.815	**0.840**	0.815
Glass1	0.954	**0.961**	**0.961**	**0.961**	0.948	0.936	0.948	0.948	0.922	**0.961**	**0.961**	0.922
Glass0	0.958	**0.965**	0.958	**0.965**	**0.965**	0.876	0.958	0.944	0.951	**0.965**	0.930	0.930
Glass0123vs456	**0.951**	0.932	0.932	0.923	0.891	0.867	0.873	0.923	**0.951**	0.942	0.891	0.913
Glass6	0.863	0.896	0.863	**0.912**	0.829	0.800	0.860	0.880	0.844	0.829	**0.912**	0.846
Glass2	0.744	**0.789**	0.670	0.705	0.670	0.647	0.762	0.734	0.840	0.766	0.703	0.780
Glass4	0.688	0.753	0.753	0.733	0.688	0.615	0.688	0.721	0.721	**0.784**	0.705	**0.784**
New-thyroid2	0.914	**0.927**	0.887	**0.927**	0.914	**0.927**	0.899	0.912	0.866	0.881	0.914	0.899
New-thyroid1	0.901	**0.914**	0.901	**0.914**	0.899	0.899	**0.914**	0.899	0.899	0.859	0.869	0.897
Ecoli0vs1	**0.968**	**0.968**	**0.968**	**0.968**	**0.968**	0.937	0.942	0.961	**0.968**	**0.968**	**0.968**	**0.968**
Haberman	0.945	0.939	0.946	0.964	0.957	0.941	0.939	0.958	0.946	**0.970**	0.952	0.958
Ecoli1	**0.968**	**0.968**	**0.968**	**0.968**	**0.968**	0.937	**0.968**	**0.968**	**0.968**	**0.968**	0.886	**0.968**
Ecoli2	**0.952**	**0.952**	**0.952**	**0.952**	0.895	0.907	0.943	0.905	0.942	0.952	0.942	0.943
Ecoli3	0.912	**0.927**	**0.927**	**0.927**	0.912	0.805	0.871	**0.927**	**0.927**	**0.927**	0.886	0.873
Ecoli4	**0.866**	**0.866**	**0.866**	**0.866**	0.836	0.700	0.836	0.764	**0.866**	0.806	0.821	**0.866**
Yeast1vs7	**0.915**	**0.915**	**0.915**	**0.915**	**0.915**	0.838	0.846	0.786	0.900	0.851	**0.915**	0.882
Yeast2vs8	**0.866**	**0.866**	**0.866**	**0.866**	**0.866**	0.750	0.821	**0.866**	**0.866**	**0.866**	0.798	**0.866**
Yeast2vs4	**0.951**	**0.951**	**0.951**	**0.951**	0.922	0.905	0.923	0.932	0.942	0.942	0.941	0.903
Yeast05679vs4	**0.951**	**0.951**	**0.951**	**0.951**	0.932	0.792	0.931	0.932	0.932	0.921	0.923	**0.951**
Wisconsin	0.975	0.977	0.977	0.979	0.979	0.964	0.971	0.963	**0.985**	0.981	0.967	0.977
Yeast1458vs7	**0.915**	**0.915**	**0.915**	**0.915**	0.900	0.838	0.861	0.836	**0.915**	0.861	0.868	0.879
Abalone9vs18	**0.941**	**0.941**	**0.941**	0.929	0.929	0.886	0.930	**0.941**	**0.941**	**0.941**	0.917	0.929
Pima	**0.987**	0.979	0.972	0.976	0.972	0.975	0.972	0.959	0.974	0.978	0.974	0.960
Vehicle1	0.928	0.918	0.918	0.916	**0.939**	0.901	0.919	0.919	0.932	0.936	0.898	0.914
Vehicle2	0.930	0.953	**0.962**	0.943	0.955	0.921	0.939	0.944	0.932	0.921	0.935	0.932
Vehicle3	0.919	0.903	0.919	0.912	0.910	**0.935**	0.908	0.914	0.912	0.898	0.928	0.872
Vehicle0	0.941	0.933	**0.968**	0.948	0.933	0.941	0.928	0.953	0.948	0.938	0.950	0.938
Yeast1289vs7	0.900	**0.915**	**0.915**	**0.915**	0.884	0.806	0.900	0.851	**0.915**	**0.915**	**0.915**	0.882
Vowel0	0.917	0.882	0.858	0.913	0.907	0.830	0.934	0.858	0.855	0.905	0.884	**0.973**
Yeast1	**0.994**	**0.994**	**0.994**	**0.994**	**0.994**	0.988	**0.994**	**0.994**	0.993	**0.994**	**0.994**	0.991
Yeast3	**0.985**	**0.985**	**0.985**	**0.985**	**0.985**	0.970	0.979	0.973	**0.985**	0.982	0.979	0.979
Yeast4	**0.951**	**0.951**	**0.951**	**0.951**	0.922	0.905	0.922	0.942	0.931	**0.951**	0.942	**0.951**
Yeast5	**0.944**	**0.944**	**0.944**	0.932	0.932	0.804	0.911	0.922	0.922	0.884	0.920	0.922
Yeast6	**0.927**	**0.927**	**0.927**	0.912	**0.927**	0.861	0.871	0.901	**0.927**	0.901	**0.927**	0.901
Shuttle0vs4	0.956	0.972	0.972	**0.976**	0.968	0.930	0.956	0.956	0.96	0.972	**0.976**	0.964
Abalone19	0.852	**0.921**	**0.921**	0.834	0.891	0.848	0.906	0.906	**0.921**	0.906	**0.921**	**0.921**
Page-blocks0	0.991	0.971	0.989	**0.993**	0.990	0.991	0.987	**0.993**	0.983	0.983	0.986	0.979

Pour visualiser les résultats obtenus et déduire les meilleurs paramètres qui produisent de bonnes performances, nous avons utilisé le test de rangs alignés de Freidman. Les moyennes des rangs associées à toutes les combinaisons sont données dans le ***tableau 5.4***

Tableau 5. 4. ***Les moyennes des rangs pour différentes valeurs de générations et différentes valeurs de la taille de population***

Nombre de générations	***10***				***50***				***100***			
Taille de la Population	***20***	***50***	***70***	***100***	***20***	***50***	***70***	***100***	***20***	***50***	***70***	***100***
Rang moyen	1.973	***1.657***	1.710	1.842	1.973	2.157	2.342	2.368	1.815	3.447	2.210	1.947

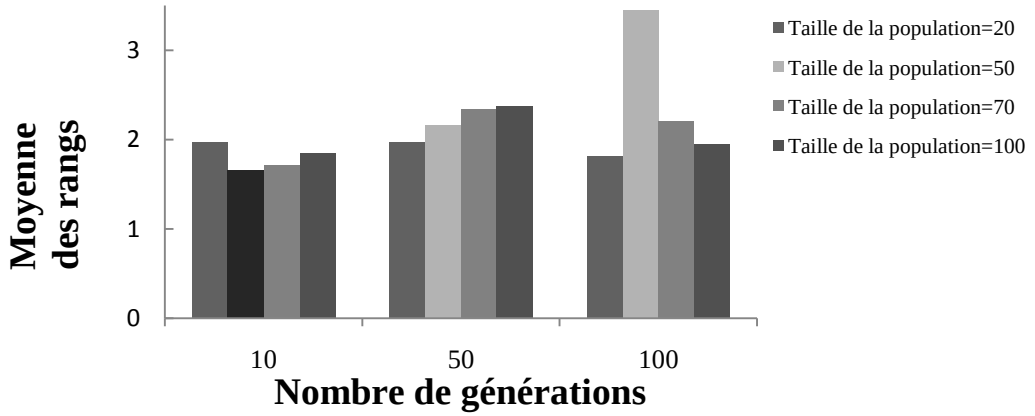

Figure 5.1. ***La moyenne des rangs pour différentes valeurs de générations et différentes valeurs de la taille de la population***

Nous remarquons que la moyenne des rangs la plus faible est associée au nombre de générations égal à 10 et à la taille de la population égale à 50. Par conséquent, ces deux valeurs produisent les meilleurs résultats.

En résumé, les paramètres de l'algorithme génétique utilisés sont donnés dans le ***tableau 5.5***.

Tableau 5. 5. ***Les paramètres de l'algorithme génétique utilisé***

Paramètres	Valeurs
Taille de la population	50
Nombre de générations	10
Probabilité de croisement	0.8
Probabilité de mutation	0.1

5. Ajustement des modules de la méthode ILS :

L'efficacité de la méthode ILS est liée fortement au choix judicieux de ses modules. Donc, nous avons testé les 03 critères d'acceptation : la marche aléatoire MA, le meilleur critère M et le critère alternatif MA/M. Ensuite, pour chaque critère, nous avons intégré ILS dans le

cycle génétique ssGA après croisement ou après mutation. Par conséquent, nous avons obtenu les 06 combinaisons présentées dans le ***tableau 5.6***.

Tableau 5.6. ***Les combinaisons des modules de la méthode ILS***

Les cas	*Abréviation*
USGA_ILS (Croisement+Marche Aléatoire)	CMA
USGA_ILS (Mutation+ Marche Aléatoire)	MMA
USGA_ILS (Croisement+Meilleur)	CM
USGA_ILS (Mutation+ Meilleur)	MM
USGA_ILS (Croisement+ Marche Aléatoire / Meilleur)	CMA/M
USGA_ILS (Mutation+Marche Aléatoire/ Meilleur)	MMA/M

Pour chaque combinaison, nous avons testé différentes valeurs de degré de perturbation. Nous rappelons que le degré de perturbation D représente le nombre d'attributs modifiés. Les degrés sont les suivants :

- D=25% : c'est-à-dire le degré est égal au nombre d'attributs/4.
- D=50% : c'est-à-dire le degré est égal au nombre d'attributs/2.
- D=75% : c'est-à-dire le degré est égal au 3*nombre d'attributs/4.

Aussi, pour chaque combinaison et pour chaque degré de perturbation, nous avons testé deux valeurs du nombre d'itérations de la méthode ILS qui sont 5 et 10. Le ***tableau 5.7*** montre la moyenne des rangs associée à chaque cas.

Tableau 5.7. ***Les moyennes des rangs des combinaisons de la méthode ILS***

Nombre d'itérations	***5***			***10***		
Degré (%)	***25***	***50***	***75***	***25***	***50***	***75***
CMA	3.860	3.920	5.550	4.390	7.230	4.500
MMA	4.184	3.184	3.236	4.973	4.105	5.315
CM	3.026	3.868	7.289	4.553	5.500	5.079
MM	5.631	3.157	3.394	4.395	4.789	5.053
CMA/M	5.052	**3.000**	5.736	4.395	**3.921**	4.789
MMA/M	3.947	4.289	5.236	3.368	4.368	4.000

Pour les itérations 5 et 10, nous remarquons que les meilleurs résultats sont obtenus pour la stratégie d'intégration de la méthode ILS après croisement, au critère d'acceptation alternatif et au degré de perturbation moyen D=50% car la moyenne des rangs qui correspond à ce cas est la plus faible.

Ces résultats sont dus au critère d'acceptation alternatif qui favorise la diversification et l'intensification et aussi au degré de perturbation moyen qui permet de s'échapper des optimums locaux et de diversifier la recherche sans une relance aléatoire de la méthode qui permet la convergence. A noter aussi, que l'intégration de la méthode ILS après croisement est la meilleure car la perturbation a une relation avec la stratégie d'intégration de la méthode

ILS : si la mutation est appliquée pour chaque attribut et ILS est intégrée après la mutation aussi, alors le degré de perturbation augmente et par conséquent la convergence n'est pas permise.

Nous remarquons aussi que la moyenne des rangs correspondante au nombre d'itérations égal à 5 sont en général inférieurs à la moyenne des rangs correspondante au nombre d'itérations égal à 10.

En conclusion, les meilleurs modules de la méthode ILS sont :

- Degré de perturbation=50%
- Nombre d'itérations=5
- Le critère d'acceptation alternatif (MA /M).
- Stratégie d'intégration de la méthode ILS est après le croisement.

6. Détails des résultats obtenus par les approches USGA et USGA_ILS :

Afin de montrer la robustesse de nos approches USGA et USGA_ILS, nous avons utilisé d'autres mesures de performances: le taux de vrais positifs TPrate, le taux de vrais négatifs TNrate et F-Mesure ***(vues dans le chapitre 3, sections 4.1.2, 4.1.3 et 4.2.2 respectivement).*** Les valeurs de ces mesures liées à chaque base de données sont données dans le ***tableau 5.8***.

Tableau 5.8.*TPrate, TNrate et F-Mesure obtenus par USGA et USGA_ILS pour les bases de données déséquilibrées*

	USGA			*USGA_ILS*		
Datasets	***TPrate***	***TNrate***	***F-Mesure***	***TPrate***	***TNrate***	***F-Mesure***
Glass1	1.000	0.924	0.962	1.000	0.936	0.968
Ecoli0vs1	1.000	0.937	0.968	1.000	0.937	0.968
Wisconsin	0.983	0.972	0.977	1.000	0.989	0.980
Pima	1.000	0.978	0.988	0.996	0.978	0.987
Iris0	1.000	0.884	0.943	1.000	0.903	0.952
Glass0	1.000	0.931	0.965	1.000	0.931	0.965
Yeast1	1.000	0.988	0.994	1.000	0.994	0.988
Vehicle2	0.979	0.929	0.953	1.000	0.978	0.988
Vehicle3	0.978	0.833	0.910	0.995	0.972	0.983
Haberman	0.975	0.905	0.940	0.987	0.941	0.963
Vehicle1	0.986	0.854	0.916	1.000	0.973	0.986
Glass0123vs456	1.000	0.869	0.936	1.000	0.905	0.953
Vehicle0	0.931	0.936	0.931	0.984	0.971	0.977
Ecoli1	1.000	0.937	0.968	1.000	0.937	0.968
New-thyroid2	1.000	0.861	0.933	0.971	0.861	0.918
New-thyroid1	0.971	0.861	0.918	1.000	0.861	0.933
Ecoli2	1.000	0.907	0.954	1.000	0.907	0.954
Glass6	0.965	0.833	0.903	1.000	0.833	0.920
Yeast3	1.000	0.970	0.984	1.000	0.970	0.984
Ecoli3	1.000	0.861	0.933	1.000	0.861	0.933
Page-blocks0	0.981	0.960	0.970	0.998	0.989	0.993
Yeast2vs4	1.000	0.905	0.953	1.000	0.886	0.944
Yeast05679vs4	1.000	0.905	0.953	1.000	0.905	0.953
Vowel0	0.855	0.911	0.880	1.000	0.946	0.972
Glass016vs2	0.882	0.705	0.810	1.000	0.647	0.850
Glass2	0.882	0.705	0.810	1.000	0.588	0.829
Ecoli4	1.000	0.750	0.888	1.000	0.750	0.888
Yeast1vs7	1.000	0.838	0.923	1.000	0.838	0.923
Shuttle0vs4	0.983	0.961	0.971	1.000	0.961	0.980
Glass4	0.923	0.615	0.800	1.000	0.538	0.812
Abalone9vs18	1.000	0.886	0.943	1.000	0.886	0.943
Yeast1458vs7	1.000	0.838	0.923	1.000	0.838	0.923
Yeast2vs8	1.000	0.750	0.888	1.000	0.750	0.888
Yeast4	1.000	0.905	0.953	1.000	0.905	0.953
Yeast1289vs7	1.000	0.838	0.923	1.000	0.838	0.923
Yeast5	1.000	0.891	0.946	1.000	0.891	0.946
Yeast6	1.000	0.861	0.933	1.000	0.861	0.933

Les résultats obtenus dans le ***tableau 5.8*** montrent l'efficacité et la robustesse des approches USGA et USGA_ILS vu que le pourcentage des instances minoritaires (TPrate) est égal ou proche de 100% et F-Mesure est proche de 1.

7. Etudes comparatives :

Afin de montrer les performances de nos approches proposées et les autres approches utilisées dans l'étude comparative, nous divisons cette section en trois sous sections :

Les sous sections 7.1, 7.2 et 7.3 montrent les résultats obtenus par les différentes approches pour les bases de données faiblement, moyennement et fortement déséquilibrées respectivement.

Dans chaque sous section, nous présentons d'abord les résultats de toutes les approches étudiées en termes de G-Mean. A noter que les meilleurs résultats obtenus pour chaque base de données sont mis en gras. Ensuite, nous procédons aux études comparatives suivantes :

- Une étude comparative binaire entre USGA et chacune des approches comparées.
- Une étude comparative multiple entre USGA et toutes les approches comparées à la fois.
- Une étude comparative binaire entre USGA_ILS et chacune des approches comparées.
- Une étude comparative multiple entre USGA_ILS et toutes les approches comparées à la fois.
- Une étude comparative entre USGA et USGA_ILS.

Pour les études comparatives binaires, nous utilisons la méthode de test de rang signé apparié de Wilcoxon, où R^+ est la somme des rangs associés à l'approche USGA ou à USGA_ILS et R^- est la somme des rangs associés à l'approche comparée.

7.1. Pour les bases de données faiblement déséquilibrées :

Tableau 5.9.*G-Mean obtenu pour les bases de données faiblement déséquilibrées*

Base de données	*USGA*	*USGA_ILS*	*CHC*	*SMB*	*AC2*	*UB1*	*C4.5*	*PART*	*RIP*
Glass1	**0.961**	**0.967**	0.779	0.799	0.782	0.752	0.731	0.734	0.704
Ecoli0vs1	0.968	0.968	0.979	0.968	0.969	0.979	**0.980**	**0.980**	**0.980**
Wisconsin	**0.977**	**0.989**	0.966	0.963	0.965	0.961	0.948	0.940	0.952
Pima	**0.989**	**0.987**	0.698	0.743	0.707	0.758	0.706	0.720	0.718
Iris0	0.940	0.950	**1.000**	0.989	0.989	0.989	0.980	0.980	0.757
Glass0	**0.965**	**0.965**	0.779	0.814	0.801	0.838	0.791	0.797	0.757
Yeast1	**0.994**	**0.994**	0.655	0.705	0.637	0.722	0.604	0.616	0.647
Vehicle2	0.953	**0.988**	0.778	**0.977**	0.972	0.961	0.938	0.945	0.961
Vehicle3	**0.903**	**0.984**	0.448	0.741	0.732	0.787	0.679	0.461	0.582
Haberman	**0.939**	**0.964**	0.385	0.625	0.416	0.661	0.454	0.444	0.500
Vehicle1	**0.918**	**0.986**	0.448	0.744	0.751	0.760	0.622	0.470	0.674

7.1.1. Etude comparative entre l'approche USGA et les autres approches :

Tableau 5. 10. ***Test de rang signé apparié de Wilcoxon de l'approche USGA et les approches comparées pour les bases de données faiblement déséquilibrées***

Les approches Comparées	R^+	R^-	*T*	*Hypothèse (α=0.05)*	*L'approche testée*
USGA vs. C45	58	8	8	Rejetée par USGA	USGA
Our vs. RIP	61	5	5	Rejetée par USGA	USGA
USGA vs. PART	58	8	8	Rejetée par USGA	USGA
USGA vs. AC2	57	9	9	Rejetée par USGA	USGA
USGA vs. SMB	57	9	9	Rejetée par USGA	USGA
USGA vs. CHC	61	5	5	Rejetée par USGA	USGA
USGA vs. UB1	58	8	8	Rejetée par USGA	USGA

Pour les bases de données faiblement déséquilibrées, l'approche USGA est la meilleure dans tous les cas car le test statistique T est inférieur à la valeur critique (10 pour 11 bases de données et α=0.05.) et aussi la moyenne des rangs pour USGA est la plus petite ***(voir les figures 5.2 et 5.3)***.

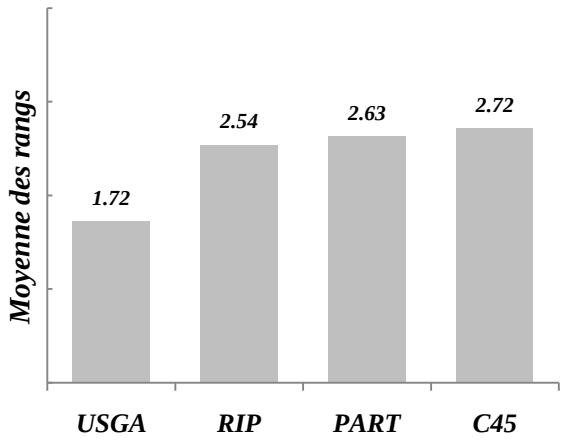

Figure 5.2. ***Moyenne des rangs pour les bases de données faiblement déséquilibrées des approches classiques et USGA***

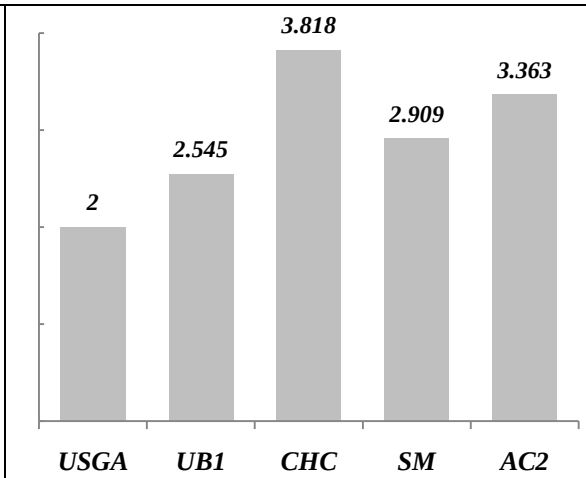

Figure 5.3. ***Moyenne des rangs pour les bases de données faiblement déséquilibrées des approches récentes et USGA***

7.1.2. Etude comparative entre l'approche USGA_ILS et les autres approches

Tableau 5. 11. ***Test de rang signé apparié de Wilcoxon de l'approche USGA_ILS et les approches comparées pour les bases de données faiblement déséquilibrées***

Les approches comparées	R^+	R^-	*T*	*Hypothèse (α=0.05)*	*L'approche testée*
USGA_ILS vs. C45	62	4	4	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. RIP	61	5	5	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. PART	61	5	5	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. AC2	61	5	5	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. SMB	60	6	6	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. CHC	62	4	4	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. UB1	61	5	5	Rejetée par USGA_ILS	USGA_ILS

Pour les bases de données faiblement déséquilibrées, l'approche USGA_ILS est la meilleure dans tous les cas car le test statistique T est inférieur à la valeur critique (10 pour 11 bases de données et α=0.05.) et aussi la moyenne des rangs pour USGA_ILS est la plus petite ***(voir les figures 5.4 et 5.5)***.

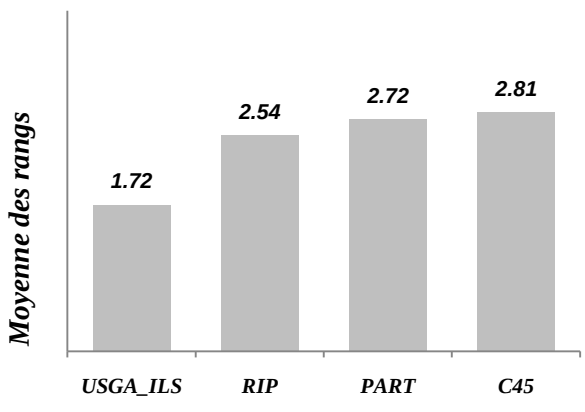

Figure 5.4. ***Moyenne des rangs pour les bases de données faiblement déséquilibrées des approches classiques et USGA_ILS***

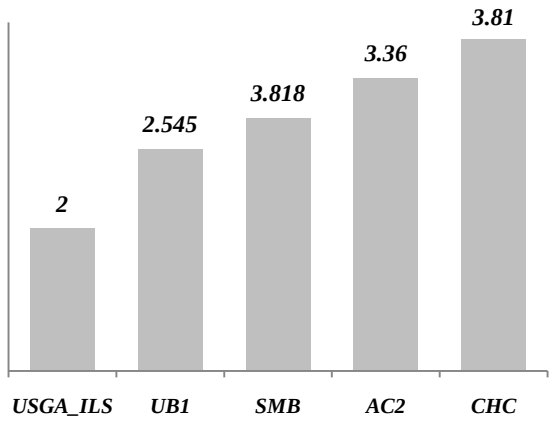

Figure 5.5. ***Moyenne des rangs pour les bases de données faiblement déséquilibrées des approches récentes et USGA_ILS***

7.1.3. Etude comparative entre les approches USGA et USGA_ILS :

Tableau 5. 12. ***Test de rang signé apparié de Wilcoxon des approches USGA et USGA_ILS pour les bases de données faiblement déséquilibrées***

Les approches comparées	R^+	R^-	*T*	*Hypothèse (α=0.05)*	*L'approche testée*
USGA_ILS vs. USGA	63	3	3	Rejetée par USGA_ILS	USGA_ILS

L'approche mémétique USGA_ILS est meilleure que l'approche génétique pure USGA car le test statistique T est inférieur à la valeur critique et aussi la moyenne des rangs pour USGA_ILS est la plus petite ***(voir la figure 5.6)***.

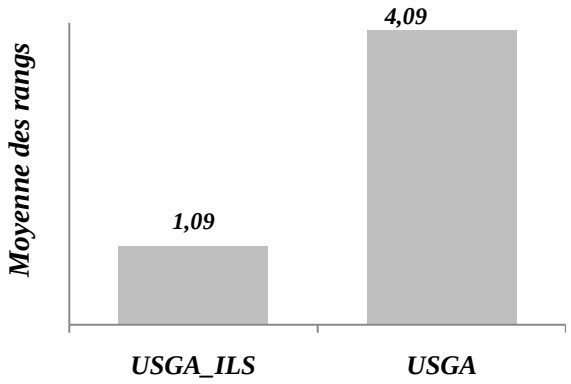

Figure 5. 6. ***Moyenne des rangs pour les bases de données faiblement déséquilibrées des approches USGA et USGA_ILS***

7.2. Pour les bases de données moyennement déséquilibrées:

Tableau 5. 13. ***G-Mean obtenu pour les bases de données moyennement déséquilibrées***

Bases de données	*USGA*	*USGA_ILS*	*CHC*	*SMB*	*AC2*	*UB1*	*C4.5*	*PART*	*RIP*
Glass0123vs456	**0.932**	**0.951**	0.909	0.900	0.901	0.887	0.877	0.473	0.904
Vehicle0	0.933	**0.978**	0.841	**0.963**	0.943	0.951	0.918	0.935	0.932
Ecoli1	**0.968**	**0.968**	0.810	0.876	0.876	0.896	0.809	0.828	0.880
New-thyroid2	0.927	0.914	0.959	**0.968**	0.956	0.949	0.868	0.931	0.955
New-thyroid1	0.914	0.927	0.982	**0.983**	0.945	0.963	0.887	0.938	0.900
Ecoli2	**0.952**	**0.952**	0.896	0.904	0.884	0.898	0.855	0.888	0.838
Glass6	0.896	**0.912**	0.890	0.833	0.887	**0.897**	0.898	0.898	0.900
Yeast3	**0.985**	**0.985**	0.804	0.892	0.910	0.932	0.868	0.802	0.875
Ecoli3	**0.927**	**0.927**	0.789	0.809	0.847	0.875	0.702	0.731	0.733
Page-blocks0	**0.971**	**0.993**	0.681	0.934	0.874	0.956	0.919	0.906	0.915
Yeast2vs4	0.951	0.941	0.783	0.874	0.915	**0.953**	0.817	0.811	0.827
Yeast05679vs4	**0.951**	**0.951**	0.630	0.769	0.746	0.786	0.596	0.656	0.838

7.2.1. Etude comparative entre l'approche USGA et les autres approches :

Tableau 5. 14. ***Test de rang signé apparié de Wilcoxon de l'approche USGA et les approches comparées pour les bases de données moyennement déséquilibrées***

Les approches comparées	R^+	R^-	*T*	*Hypothèse (α=0.05)*	*L'approche testée*
USGA vs. C4.5	77	1	1	Rejetée par USGA	USGA
USGA vs. RIP	72	6	6	Rejetée par USGA	USGA
USGA vs. PART	69	9	9	Rejetée par USGA	USGA
USGA vs. AC2	69	9	9	Rejetée par USGA	USGA
USGA vs. SMB	66	12	12	Rejetée par USGA	USGA
USGA vs. CHC	70	8	8	Rejetée par USGA	USGA
USGA vs. UB1	59	19	19	***Non rejetée***	***Pas de différence***

Pour les bases de données moyennement déséquilibrées, USGA est meilleure que les approches CHC, AC2, SMB, PART et RIP car le test statistique T est inférieur à la valeur critique (13 pour 12 bases de données et α=0.05.) et aussi la moyenne des rangs pour USGA est la plus faible comme il est illustré dans les ***figures 5.7 et 5.8***. Cependant, il n y'a pas de différence entre USGA et l'approche UB1 parce que le test statistique est supérieure à la valeur critique.

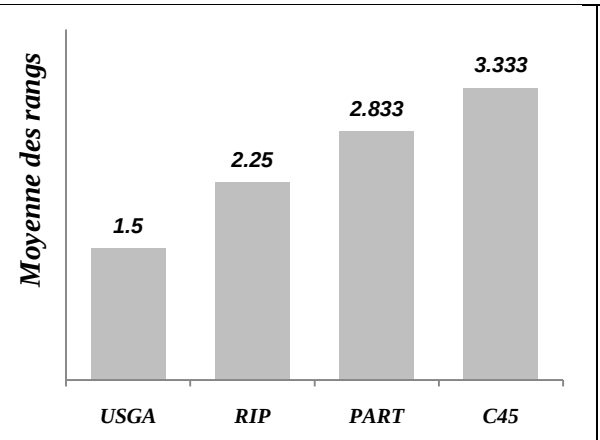

Figure 5.7. ***Moyenne des rangs pour les bases de données moyennement déséquilibrées des approches classiques et USGA***

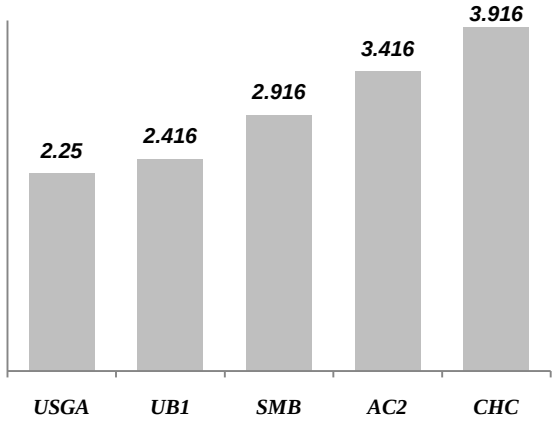

Figure 5.8. ***Moyenne des rangs pour les bases de données moyennement déséquilibrées des approches récentes et USGA***

7.2.2. Etude comparative entre l'approche USGA_ILS et les autres approches :

Tableau 5. 15. ***Test de rang signé apparié de Wilcoxon de l'approche USGA_ILS et les approches comparées pour les bases de données moyennement déséquilibrées***

Les approches comparées	R^+	R^-	*T*	*Hypothèse (α=0.05)*	*L'approche testée*
USGA_ILS vs. C4.5	74	4	4	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. RIP	74	4	4	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. PART	74	4	4	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. AC2	74	4	4	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. SMB	71	7	7	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. CHC	72	6	6	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. UB1	70	8	8	Rejetée par USGA_ILS	USGA_ILS

Pour les bases de données moyennement déséquilibrées, USGA_ILS est meilleure que les approches CHC, AC2, SMB, PART et RIP car le test statistique T est inférieur à la valeur critique (13 pour 12 bases de données et α=0.05.) et aussi la moyenne des rangs pour USGA_ILS est la plus petite ***(voir les figures 5.9 et 5.10)***.

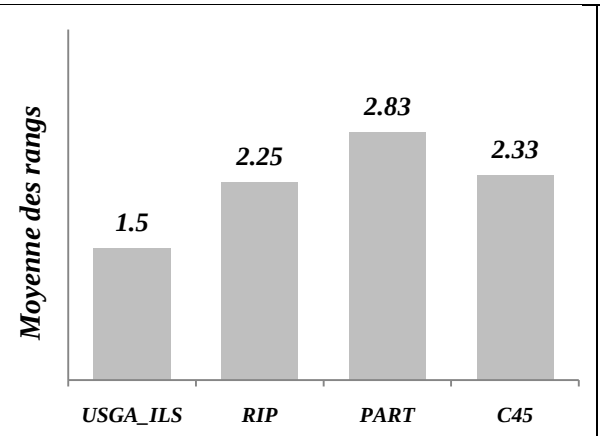

Figure 5.9. ***Moyenne des rangs pour les bases de données moyennement déséquilibrées des approches classiques et USGA_ILS***

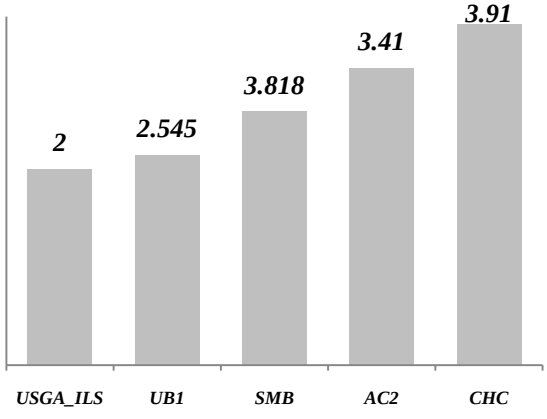

Figure 5.10. ***Moyenne des rangs pour les bases de données moyennement déséquilibrées pour des approches récentes et USGA_ILS***

7.2.3. Etude comparative entre les approches USGA et USGA_ILS :

Tableau 5. 16. ***Test de rang signé apparié de Wilcoxon des approches USGA et USGA_ILS pour les bases de données moyennement déséquilibrées***

Les approches comparées	R^+	R^-	*T*	*Hypothèse (α=0.05)*	*L'approche testée*
USGA_ILS vs. USGA	55	8	8	Rejetée par USGA_ILS	USGA_ILS

L'approche mémétique USGA_ILS est meilleure que l'approche génétique pure USGA car le test statistique T est inférieur à la valeur critique et la moyenne des rangs pour USGA_ILS est la plus petite ***(voir la figure 5.11)***.

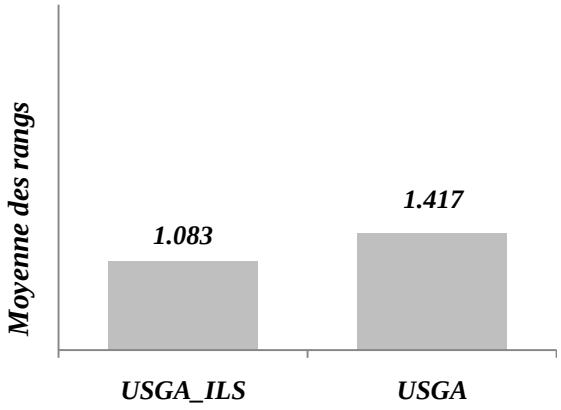

Figure 5. 11. ***Moyenne des rangs pour les bases de données moyennement déséquilibrées des approches USGA et USGA_ILS***

7.3. Pour les bases de données fortement déséquilibrées

Tableau 5. 17. ***G-Mean obtenu pour les bases de données fortement déséquilibrées***

Bases de données	*USGA*	*USGA_ILS*	*CHC*	*SMB*	*AC2*	*UB1*	*C4.5*	*RIP*	*PART*
Vowel0	0.882	**0.970**	0.896	**0.991**	0.968	0.947	0.939	0.910	0.945
Glass016vs2	**0.789**	**0.804**	0.000	0.542	0.408	0.663	0.071	0.241	0.241
Glass2	0.789	**0.766**	0.000	0.754	0.671	**0.808**	0.476	0.625	0.237
Ecoli4	0.866	0.866	**0.918**	0.872	0.927	0.883	0.701	0.624	0.829
Yeast1vs7	**0.915**	**0.915**	0.162	0.624	0.698	0.763	0.480	0.311	0.547
Shuttle0vs4	0.972	0.980	0.995	**1.000**	0.941	0.983	0.999	0.999	1.000
Glass4	0.753	0.733	0.000	0.898	**0.864**	0.829	0.999	0.866	0.817
Abalone9vs18	**0.941**	0.941	0.348	0.700	0.025	0.763	0.999	0.482	0.574
Yeast1458vs7	**0.915**	**0.915**	0.000	0.038	0.362	0.615	0.000	0.180	0.000
Yeast2vs8	**0.866**	**0.866**	0.728	0.718	0.312	0.775	0.000	0.000	0.670
Yeast4	**0.951**	**0.951**	0.139	0.656	0.008	0.837	0.502	0.556	0.520
Yeast1289vs7	**0.915**	**0.915**	0.000	0.580	0.565	0.665	0.316	0.364	0.363
Yeast5	**0.944**	0.944	0.704	0.905	0.875	0.957	0.822	0.793	0.875
Yeast6	**0.927**	**0.927**	0.386	0.793	0.595	0.854	0.732	0.712	0.607
Abalone19	**0.852**	**0.921**	0.000	0.081	0.080	0.658	0.000	0.000	0.000

7.3.1. Etude comparative entre l'approche USGA et les autres approches :

Tableau 5. 18. ***Test de rang signé apparié de Wilcoxon de l'approche USGA et les approches comparées pour les bases de données fortement déséquilibrées***

Les approches comparées	R^+	R^-	*T*	*Hypothèse (α=0.05)*	*L'approche testée*
USGA vs. C4.5	107	13	13	Rejetée par USGA	USGA
USGA vs. RIP	114	6	6	Rejetée par USGA	USGA
USGA vs. PART	111	8	8	Rejetée par USGA	USGA
USGA vs. AC2	109	11	11	Rejetée par USGA	USGA
USGA vs. SMB	105	15	15	Rejetée par USGA	USGA
USGA vs. CHC	111	6	6	Rejetée par USGA	USGA
USGA vs. UB1	99	21	21	Rejetée par USGA	USGA

Pour les bases de données fortement déséquilibrées, USGA est la meilleure dans tous les cas car le test statistique T est inférieur à la valeur critique (25 pour 15 bases de données et α=0.05.) et aussi la moyenne des rangs pour USGA est la plus petite ***(voir les figures 5.12 et 5.13)***.

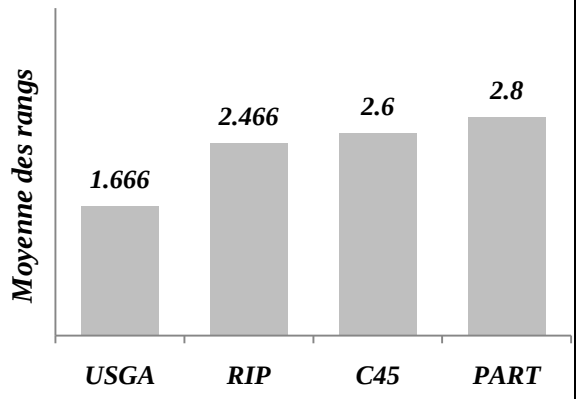

Figure 5. 12. ***Moyenne des rangs pour les bases de données fortement déséquilibrées des approches classiques et USGA***

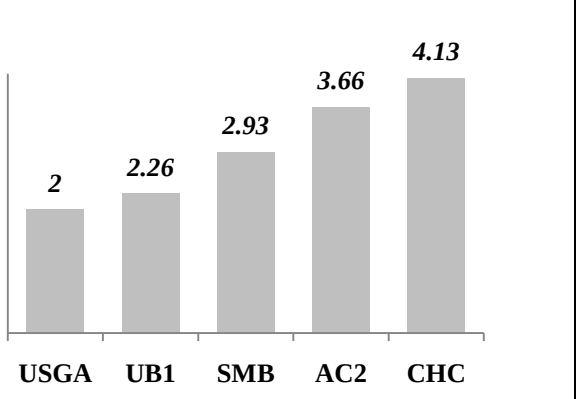

Figure 5. 13. ***Moyenne des rangs pour les bases de données fortement déséquilibrées des approches récentes et USGA***

7.3.2. Etude comparative entre l'approche USGA_ILS et les autres approches :

Tableau 5. 19. ***Test de rang signé apparié de Wilcoxon de l'approche USGA_ILS et les approches comparées pour les bases de données fortement déséquilibrées***

Les approches comparées	R^+	R^-	*T*	*Hypothèse (α=0.05)*	*L'approche testée*
USGA_ILS vs. C4.5	113	7	7	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. RIP	110	10	10	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. PART	114	6	6	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. AC2	111	9	9	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. SMB	108	12	12	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. CHC	111	9	9	Rejetée par USGA_ILS	USGA_ILS
USGA_ILS vs. UB1	109	11	11	Rejetée par USGA_ILS	USGA_ILS

Pour les bases de données fortement déséquilibrées, USGA_ILS est la meilleure dans tous les cas car le test statistique T est inférieur à la valeur critique (25 pour 15 bases de données et α=0.05.) et aussi la moyenne des rangs pour USGA_ILS est la plus petite ***(voir les figures 5.14 et 5.15)***.

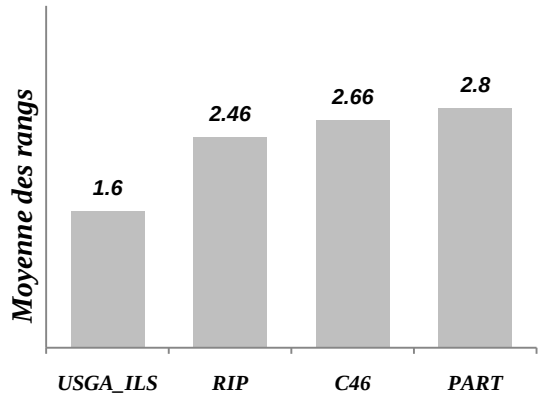

Figure 5.14. ***Moyenne des rangs pour les bases de données fortement déséquilibrées des approches classiques et USGA_ILS***

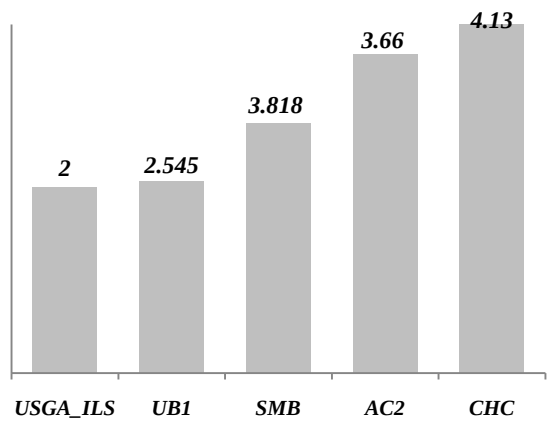

Figure 5.15. ***Moyenne des rangs pour les bases de données fortement déséquilibrées des approches récentes et USGA_ILS***

7.3.3. Etude comparative entre les approches USGA et USGA_ILS :

Tableau 5. 20. ***Test de rang signé apparié de Wilcoxon test des approches USGA et USGA_ILS pour les bases de données fortement déséquilibrées.***

Les approches comparées	R^+	R^-	*T*	*Hypothèse (α=0.05)*	*L'approche testée*
USGA_ILS vs. USGA	61	25	24	Rejetée par USGA_ILS	USGA_ILS

L'approche mémétique USGA_ILS est meilleure que l'approche génétique pure USGA car le test statistique Test inférieur à la valeur critique et la moyenne des rangs pour USGA_ILS est la plus petite telle qu'il est illustré dans la ***figure 5.16***.

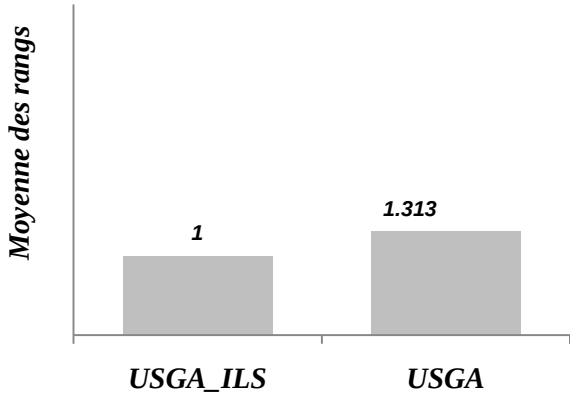

Figure 5. 16. ***Moyenne des rangs pour les bases de données fortement déséquilibrées des approches USGA et USGA_ILS***

8. Discussion des résultats obtenus :

- **Pour l'approche génétique pure :**

Après l'analyse des résultats obtenus, nous concluons que les résultats dépendent de la valeur du ratio du déséquilibre IR.

L'approche génétique pure USGA est la meilleure pour les bases de données faiblement et fortement déséquilibrées. Cependant, pour les bases de données moyennement déséquilibrées, nos résultats sont meilleurs ou similaires.

En fait, ces résultats sont dus au principe de notre approche qui utilise une méthode intelligente de sous-échantillonnage dans la première phase. Cette méthode conserve les concepts des instances majoritaires supprimées dans le premier modèle M1. Par conséquent, les règles du M1 vont être rajoutées au modèle M2 pour réduire le taux de mal classification des instances majoritaires.

En outre, ces résultats sont dus aussi à l'utilisation de deux phases évolutionnaires qui utilisent deux fonctions objectives différentes. Chacune favorise les instances majoritaires ou minoritaires.

En plus, la phase de post-traitement du modèle final M3 supprime les règles contradictoires et spécifiques pour éviter le chevauchement entre les règles et par conséquent, les performances de notre modèle final augmentent.

- **Pour l'approche mémétique:**

Les résultats obtenus par l'approche mémétiques sont les meilleurs. En fait, la méthode de recherche locale itérée ILS permet l'exploration de l'espace de recherche. En outre, nous avons testé toutes les combinaisons des modules de la méthode ILS dont l'interaction entre eux produit de bons résultats.

9. Evaluation du temps d'exécution:

Afin d'évaluer le temps d'exécution des approches étudiées, nous avons calculé le temps moyen de chaque groupe de bases de données qui ont le même nombre d'instances et le même nombre d'attributs. Les valeurs obtenues sont présentées dans le ***tableau 5.21***.

Tableau 5. 21. Temps d'exécution des approches étudiées en millisecondes (ms)

#Insts.	*#Atts*	*USGA*	*USGA_ILS*	*C4.5*	*RIP*	*PART*	*AC2*	*SMB*	*UB1*
150	4	002.00	07.00	0.02	0.02	0.01	03.00	00 9.00	03.00
214	9	008.00	21.00	0.01	0.16	0.01	05.66	023.00	07.33
306	3	002.00	08.00	0.02	0.02	0.02	03.00	014.00	03.00
336	7	005.00	17.75	0.02	0.02	0.02	06.00	026.75	04.00
988	13	073.00	36.00	0.02	0.08	0.03	05.00	126.00	08.00
1428	8	008.00	58.00	0.02	0.46	0.04	20.60	148.60	20.20
1829	9	075.00	62.00	0.02	0.03	0.02	02.00	012.00	03.00
4174	8	175.00	48.00	0.03	0.03	0.08	18.00	547.00	10.00

Comme il est montré dans le ***tableau 5.21*** et dans la visualisation graphique présentée dans la ***figure 5.17***, nous remarquons que le temps d'exécution ne dépend pas seulement du nombre d'instances et du nombre d'attributs. Mais, il dépend aussi de la méthode d'extraction des règles de classification :

- Le temps d'exécution des approches classiques est le plus faible car la méthode d'extraction des règles est à un seul modèle.
- Cependant, le temps d'exécution de l'approche SMB est élevé puisqu'elle utilise la méthode de prétraitement SMOTE d'une part et d'autre part elle utilise une méthode à plusieurs modèles.
- Le temps d'exécution de l'approche génétique pure USGA est moyen par rapport aux autres approches.
- Le temps d'exécution de l'approche USGA_ILS est le plus élevé à cause de l'intégration de la méthode de recherche locale itérée qui appelle à son tour une autre méthode de recherche locale simple.

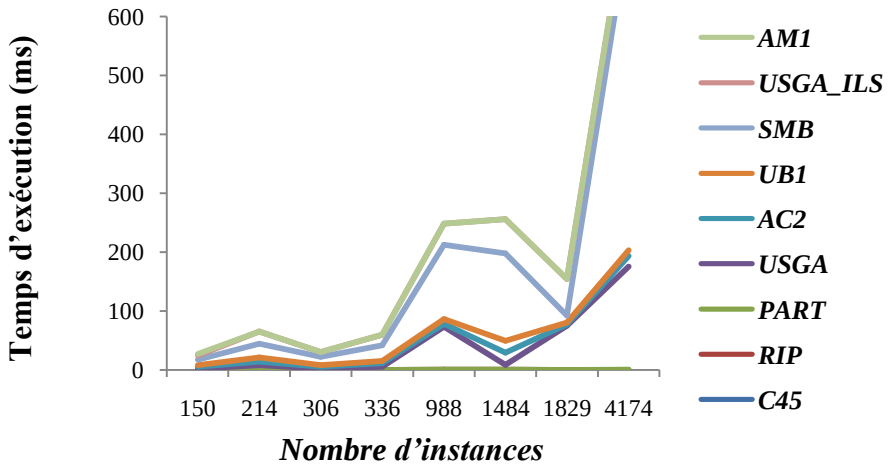

Figure 5. 17. ***Le temps d'exécution des approches étudiées***

10. Analyse de la complexité de nos approches proposées:

10.1. La complexité du cycle génétique ssGA :

L'étape la plus coûteuse d'un processus évolutionnaire est l'évaluation de la fonction objective de chaque individu.

Le coût d'évaluation est proportionnel à la taille de la base de données (n). Par conséquent, la complexité d'évaluation d'un individu est O(n). Dans ssGA, nous avons les évaluations suivantes :

1. La population initiale de taille taille_pop est évaluée une seule fois au début de l'algorithme. Donc, la complexité d'évaluation de la population initiale est donnée dans l'équation (5. 5):

$$\boldsymbol{Complexité\ (Population) = O(n * taille\_pop) \approx O(n)} \qquad \textbf{(5. 5)}$$

2. Les deux enfants résultants du croisement sont évalués. La complexité d'évaluation est donnée dans l'équation (5. 6):

$$\boldsymbol{Complexité(Deux\ individus) = O(2 * n) \approx O(n)} \qquad \textbf{(5. 6)}$$

3. L'enfant muté est aussi évalué avec la complexité donnée dans l'équation (5. 7):

$$\boldsymbol{Complexité(Individu) = O(n)} \qquad \textbf{(5. 7)}$$

Donc, la complexité d'un cycle génétique ssGA est : O(n) + O(n) + O(n). Par conséquent, la complexité globale est donnée dans l'équation (5.8):

$$\boldsymbol{Complexité(Cycle\ génétique) = O(n)} \qquad \textbf{(5. 8)}$$

Pour un nombre de générations égal à Max_gen, la complexité est donnée dans l'équation (5.9):

$$\boldsymbol{Complexité(Générations) = O(n * Max\_gen) \approx O(n)} \qquad \textbf{(5. 9)}$$

10.2. La complexité de l'approche USGA :

La complexité des phases de l'approche USGA est donnée dans les deux sous sections suivantes :

Phase 1: le nombre d'instances majoritaires qui vont être supprimées est égal approximativement à ***n=Min*(IR-1).*** Dans le pire des cas, une règle est générée pour chaque instance majoritaire après 10 itérations. La complexité de cette phase est donnée dans l'équation (5.10):

$$\boldsymbol{Complexité(Phase1) = O(n^2) * O(n) \approx O(n^3)} \qquad \textbf{(5. 10)}$$

Phase 2: dans cette phase, la taille de la base d'apprentissage est égale à n. Dans le pire des cas, une règle est générée pour chaque instance. Donc, le cycle génétique est appelé n fois. La complexité dans la phase 2 est donnée dans l'équation (5.11)

$$\boldsymbol{Complexité(Phase2) = O(n) * O(n) \approx O(n^2)} \qquad \textbf{(5.11)}$$

Au total, la complexité de notre approche USGA est donnée dans l'équation (5.12) :

$$\boldsymbol{Complexité(USGA) = O(n^3) + O(n^2) \approx O(n^3)} \qquad \textbf{(5. 12)}$$

Nous concluons que l'approche USGA a ***une complexité polynomiale*** d'ordre 3.

10.3. La complexité de l'approche mémétique USGA_ILS :

L'approche mémétique USGA_ILS a la même complexité que l'approche USGA car le nombre d'évaluations supplémentaires est négligent.

11. Conclusion :

Dans ce chapitre, nous avons exposé les résultats des expérimentations que nous avons réalisées par l'utilisation de tous les outils nécessaires. Les tests ont porté sur nos deux approches évolutionnaires proposées en utilisant 38 bases de données bi-classes. Concernant l'approche génétique pure USGA, les tests montrent clairement sa fiabilité et sa robustesse à bien découvrir des règles qui représentent les instances majoritaires et minoritaires et à produire de bonnes performances vues les valeurs élevées de taux de classification par rapport aux approches utilisées dans les études comparatives. En ce qui concerne l'approche mémétique USGA_ILS, les tests montrent qu'elle est meilleure que l'approche génétique pure.

Malgré les limites de nos approches en termes de coût de calcul, notre approche surpasse les autres en termes de G-Mean, ce qui permet une grande précision prédictive rendant l'augmentation du coût négligeable.

CONCLUSION GENERALE ET PERSPECTIVES

Le problème de la classification dans les bases de données déséquilibrées est un sujet de recherche actif en intelligence artificielle, en apprentissage automatique et en exploration de données. Ceci est dû aux deux facteurs. En premier lieu, les cas rares apparaissent dans plusieurs domaines tels que la détection des fraudes et le diagnostic médical. En second lieu, les approches et les méthodes de la classification complète ignorent les instances minoritaires ou ils les considèrent comme des instances bruyantes et ils tendent à extraire des règles spécifiques aux instances majoritaires. Par conséquent, la plupart des modèles ont de mauvaises performances pour les bases de données déséquilibrées, car ils sont conçus pour minimiser le taux d'erreur global.

Plusieurs méthodes et approches ont été proposées pour traiter le problème de la classification dans les bases de données déséquilibrées. Ces méthodes ont été divisées en trois niveaux : le niveau données, le niveau algorithme et le niveau hybride.

Au niveau données, la distribution des instances est altérée par le sous-échantillonnage qui supprime les instances majoritaires ou le sur-échantillonnage qui duplique les instances minoritaires ou par l'échantillonnage hybride.

Au niveau algorithme, les solutions proposées sont divisées en deux catégories : la modification des algorithmes de la classification complète pour les adapter à la classification partielle et la proposition des algorithmes spécifiques.

Au niveau hybride, quelques algorithmes et approches de la classification complète doivent être combinés avec les méthodes d'échantillonnage pour qu'ils soient adaptables à la classification partielle.

Dans notre thèse, nous avons proposé une approche hybride à trois phases, la première phase utilise le sous-échantillonnage guidé par les règles de classification pour équilibrer la base de données sans la perte des concepts contenus dans les instances majoritaires supprimées. La deuxième phase sert à extraire des règles de classification représentant les instances majoritaires et minoritaires. Enfin, la troisième phase est une phase de post-traitement, qui sert à supprimer les règles contradictoires et les règles spécifiques.

La classification est un problème NP-difficile qui ne peut pas être résolu par une méthode exacte. Dans notre thèse, les règles de classification ont été extraites à l'aide des algorithmes génétiques et mémétiques. Par conséquent, nous avons proposé deux approches qui ont le même principe mais avec deux métaheuristiques différentes. La première approche que nous avons appelée USGA utilise les algorithmes génétiques purs dans les deux phases. Tant dès que la deuxième approche USGA_ILS utilise les

algorithmes génétiques dans la première phase et les algorithmes génétiques hybridés avec la méthode de recherche locale itérée ILS dans la deuxième phase.

Pour tester nos approches proposées ainsi que l'utilité d'intégration de la méthode de recherche locale itérée ILS, nous avons mené des tests sur 38 bases de données de différentes valeurs du ratio de déséquilibre IR. Les résultats ont été obtenus par l'utilisation des mesures de performances des modèles les plus appropriées.

Les résultats des expérimentations obtenus montrent la fiabilité et la robustesse de nos approches évolutionnaires proposées. Ils prouvent aussi que les résultats obtenus par l'approche mémétique USGA_ILS sont meilleurs que ceux obtenus par l'approche génétique pure USGA. Ceci est dû à l'utilité d'intégration de la méthode de recherche locale itérée dans le cycle génétique et le choix de la bonne combinaison des ses modules.

Les études comparatives ont été faites avec nos approches et quelques approches classiques telles que C45, PART et RIPPER et quelques approches récentes telles que SMB, UB1, AC2 et CHC en faisant appel aux tests statistiques. Cette étude a prouvé l'efficacité et la robustesse de nos approches.

Perspectives :

- Traiter le problème de la classification dans les bases de données déséquilibrées contenant plusieurs classes.
- Traiter les bases de données volumineuses en utilisant le parallélisme.
- Traiter le problème de la classification dans les bases de données bi-classes et multi-classes en étudiant d'autres facteurs tels que la présence de petites disjonctions des instances minoritaires et le chevauchement entre les classes.

REFERENCES BIBLIOGRAPHIQUES

[1] J. Han and M. Kamber. “Data Mining Concepts and Techniques”, 2006.
[2] M. Tavallaee, N. Stakhanova and A. Ghorbani. “Toward credible evaluation of anomaly-based intrusion-detection methods”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 5, pp. 516-524, 2010.
[3] G. Batista, R. C. Prati and M. C. Monard. “A study of the behaviour of several methods for balancing machine learning training data”, *ACM SIGKDD Explorations Newsletter*,- Special issue on learning from imbalanced datasets, vol. 6, no. 1, pp. 20-29, 2004.
[4] N. Japkowicz and S. Stephen. “The class imbalance problem: A systematic study”, *Intelligent Data Analysis*, vol. 6, no.5, pp. 429-449, 2002.
[5] N. Japkowicz, R. C. Holte, C. X. Ling and S. Matwin S. (Eds.). Learning from Imbalanced Data Sets Workshop, technical report WS-00-05, 2000.
[6] G. M. Weiss and F. Provost, “Learning when training data are costly: The effect of class distribution on tree induction”, *Artificial Intelligence Research archive*, vol. 19, no. 1, pp. 315-354, 2003.
[7] Y. Tang, Y. Zhang, N. V. Chawla and S. Krasser. “SVMs Modeling for Highly Imbalanced Classification”,*IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 1, pp. 281-288, 2009.
[8] R. Alejo, V. García, J. M. Sotoca, R. A. Mollineda, and J. S. Sánchez. “Improving the performance of the RBF neural networks trained with imbalanced samples”, Lecture Notes in Computer Science, vol. 4507, pp. 162-169, 2007.
[9] X. Fu, L. Wang, K. S. Chua and F. Chu. “Training rbf neural networks on unbalanced data”, Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'OZ), vol. 2, pp. 1016-1020, 2002.
[10] Y. L. Murphey, H. Wang, G. Ou and L.A. Feldkamp. “OAHO: an Effective Algorithm for Multi-Class Learning from Imbalanced Data”, IEEE International Joint Conference onNeural Networks, pp. 406-411, 2007.
[11] K. Yoon and S. Kwek. “A data reduction approach for resolving the imbalanced data issue in functional genomics”, *Neural Computing and Applications*, vol. 16, pp. 295-306, 2007.
[12] Z. H. Zhou and X.Y. Liu. “Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63-77, 2006.
[13] H. Yu, J. Ni and J. Zhao. “ACOSampling: An ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data”, Neurocomputing, vol. 101, pp. 309-318, 2013.
[14] G. Qiong, W. Xian-Ming, W. Zhao, N. Bing and X. Chun-Sheng. “An Improved SMOTE Algorithm Based on Genetic Algorithm for Imbalanced Data Classification”, *Digital Information Management*, vol. 14, no. 2, pp. 92-103. 2016.
[15] K. Ali, S. Manganaris, and R. Srikant. “Partial classification using association rules”, proceedings of the third International conference on Knowledge Discovery and Data Mining (KDD’97), pp. 115-118, 1997.
[16] P. Riddle, R. Segal and O. Etzioni. “Representation design and brute-force induction in a Boeing manufacturing domain”, *Applied Artificial Intelligence*, vol. 8, no. 1, pp. 125-147, 1994.
[17] A. Fernández, S. García, M. J, del jesus and F. Herrera. “A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets”, *Fuzzy Sets and Systems*, vol. 159, no. 18, pp. 2387-2398, 2007.
[18] G. M. Weiss. “Mining with Rarity: A Unifying Framework”, *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7-19, 2004.

[19] S. García and F. Herrera. "Evolutionary Undersampling for Classification with Imbalance Datasets: Proposals and Taxonomy", *Evolutionary Computation,* vol. 17, no. 3, pp. 275-306, 2009.
[20] E. Aarts and J. K. Lenstra. "Local Search in Combinatorial Optimization", 1997.
[21] C. H. Papadimitriou. "The complexity of combinatorial optimization problems", PhD thesis, 1976.
[22] H. R. Lourenço, O. Martin, and T. Stutzle. "Iterated local search". In F. Glover and G. Kichenberger, editors, Handbook of Metaheuristics, pp. 321-353, 2003.
[23] H. Lourenço, O. Martin, and T. stutzle. "A gentle introduction to iterated local search", Proceedings of the fourth Metaheuristics International Conference (MIC'2001), vol. 1, 2001.
[24] G. Leguizamón and Z. Michalewicz. "A new version of Ant System for subset problems". Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99), pp. 1459-1464, 1999.
[25] J. H. Holland. "Outline for a logical theory of adaptive systems", *Association of Computing Machinery,* vol. 9, no. 3, pp. 297-314, 1962.
[26] D. E. Goldberg. "Genetic Algorithms in Search, Optimization, and Machine Learning", 1989.
[27] Y. COUEQUE, J. OHLER et S. TOLLARI. "Algorithmes génétiques pour résoudre le problème du commis voyageur". Travail de recherche mis sur INTERNET, avril 2002 *http://lsis.univ-tln.fr/~tollari/TER/AlgoGen1/node5.html.*
[28] M. YAGOUNI. Contribution à la résolution des problèmes complexes de l'Optimisation Combinatoire. Thèse doctorat, pp. 83, 2017.
[29] A. E. Eiben, J. E. Smith. "Introduction to Evolutionary Computing", 2003.
[30] P. Moscato. "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms",Technical Report C3P 826, Caltech Concurrent Computation Program, 1989.
[31] D. A. Montague, Fraud Prevention Techniques for Credit Card Fraud. NY, New York: Spring-field Press, 2006.
[32] A. Abelovszky, Plastic Card Fraud and Safety Measure. London, England: National Press, 2008. Book
[33] S. Thorat, S. Kute. "Medical Data Mining Life Cycle and its Role in Medical Domain". International Journal of Computer Science and Information Technologies (IJCSIT), vol. 5, no. 4, pp. 5751-5755, 2014.
[34] O. Etzioni. "The World-Wide Web: quagmire or gold mine?." Communications of the ACM, Vol. 39, No. 11, pp. 65-68, 1996.
[35] J.Li, L. Wong and Q.Yang. "Data mining in Bioinformatics", *IEEE Intelligent System,* vol. 20, no. 6, pp. 16-18, 2005.
[36] H.Liu, J. Li and L. Wong. "Use of Extreme Patient Samples for Outcome Prediction from Gene Expression Data", *Bioinformatics,* vol. 21, no. 16, pp. 3377-3384, 2005.
[37] A. A. Freitas and S. H. Lavington. "Mining Very Large Databases with Parallel Processing", 1998.
[38] D. J. Hand. "Construction and Assessment of Classification Rules", 1997.
[39] C.Fraley and A.E. Raftery. "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis", Technical Report N. 329, 1998.
[40] J. A. Hartigan and M. A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm", *Royal Statistical Society. Series C (Applied Statistics),* vol. 28, no. 1, pp. 100-108, 1979.
[41] S. P. Lloyd. Least squares quantization in PCM. IEEE transactions on information theory, vol. IT-28, no. 2, pp. 129-137, 1982.

[42] L. Kaufman and P. J. Rousseeuw. "Clustering by means of medoids", *Statistical Data Analysis based on the L1 Norm,* edited by Y. Dodge, pp. 405-416, 1987.
[43] M. Rathi. "Regression modeling technique on data mining for prediction of CRM", international conference on advances in information and Computing Technologies, pp. 195-199, 2010.
[44] A. Goldenberg, J. Kubica, P. Komarek, A. Moore and J. Schneider. "A comparison of statistical and machine learning algorithms on the task of link completion", KDD Workshop on Link Analysis for Detecting Complex Behavior, pp. 8, 2003.
[45] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", Proceedings of the 1993 ACM SIGMOD international conference on Management of data (SIGMOD'93), vol. 22, no. 2, pp. 207-216, 1993.
[46] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations", Proceedings of the 1997 ACM SIGMOD international conference on Management of data (SIGMOD '97), vol. 26, no. 2, pp. 265-276, 1997.
[47] R. Agrawal, R. Srikant. "Fast algorithms for mining association rules", Proceedings of the twentieth International Conference on Very Large Data Bases (VLDB '94), vol. 1215, pp. 487- 499, 1994.
[48] Z. Yang, W. Tang, A. Shintemirov, and Q.Wu. "Association rule mining-based dissolved gas analysis for fault diagnosis of power transformers", *IEEE Transactions Systems, Man, and Cybernetics*, part C, vol. 39, no. 6, pp. 597- 610, 2009.
[49] H. Lu, R. Setiono and H Liu. "Effective Data Mining Using Neural Network". *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 957-961, 1996.
[50] M. V. Fidelis, H. S. Lopes and A. A. Freitas. "Discovering comprehensible classification rules with a genetic algorithm", Proceedings of the 2000 Congress on Evolutionary Computation, vol. 1, pp. 805-810, 2000.
[51] I. H. Witten andE. Frank. "Data Mining Practical Machine Learning Tools and Techniques", 2005.
[52] K. A**.** DeJong, W. M., Spears and F. F, Gordon. "Using genetic algorithms for concept learning", *Machine Learning,* vol. 13, no. 2-3, pp. 161-188, 1993.
[53] J. S. Aguilar-Ruiz, J.C. Riquelme and M. Toro. "Evolutionary Learning of Hierarchical Decision Rules", *IEEE Transaction on Systems, Man, and Cybernetics*, Part B: Cybernetics, vol. 33, no. 2, pp. 324-331, 2003.
[54] A. A.Freitas. "A Survey of Evolutionary Algorithms for Data Mining and Knowledge", in A. Ghosh and S. Tsutsui, Advances in Evolutionary Computing, pp. 819-845, 2003.
[55] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Data Mining Researchers", Technical Report HPL-2003-4, HP Labs, 2003.
[56] T. Fawcett. "An Introduction to ROC Analysis", *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861-874, 2006.
[57] J. P.Egan. "Signal detection theory and ROC analysis", 1975.
[58] J.A.Swets, R.M. Dawes and J. Monahan. "Better decisions through science", *Scientific American*, vol. 283, no. 4, pp. 82-87, 2000.
[59] T. Mitchell. "Machine Learning", 1997.
[60] P. Adriaans and D. Zantinge. "Data Mining", 1996.
[61] S. Tuffery. "Data Mining et Statistique Décisionnelle: l'intelligence dans les bases de données", 2005.
[62] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. "Classification and regression trees", 1984.
[63] J. R. Quinlan. "C4.5: Programs for Machine Learning", 1993.

[64] F. Esposito, D. Malerba and G. Semeraro. "A comparative Analysis of Methods for Pruning Decision Trees", *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 5, pp. 476-491, 1997.
[65] C. E. Shannon. "A mathematical theory of communication", *Bell System Technical Journal*, vol. 27, no. 3, pp. 379-423, 1948.
[66] V. Vapnik. "The Nature of Statistical Learning Theory", 1995.
[67] R. E. Schapire, "The strength of weak learnability", *Machine Learning*, vol. 5, no. 2, pp. 197-227, 1990.
[68] Y. Freund and R. E. Schapire. "A decision theoretic generalization of on-line learning and an application of boosting", *Computer and System Sciences*, vol. 55, no. 1, pp.119-139, 1997.
[69] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions", *Machine Learning*, vol. 37, no. 3, pp. 297-336, 1999.
[70] L. Breiman. "Bagging predictors", *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.
[71] P. D. Turney. "Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm", *Artificial Intelligence Research*, vol. 2, pp. 369-409, 1995.
[72] P. D. Turney. "Types of cost in inductive concept learning", Proceedings of the Workshop on Cost Sensitive Learning at the Seventeenth International Conference on Machine Learning, pp. 15-21, 2000.
[73] C. Elkan. "The Foundations of Cost-Sensitive Learning", Proceedings of the Seventeenth International Joint Conference of Artificial Intelligence, pp. 973-978, 2001.
[74] B.Zadrozny and C. Elkan. "Learning and Making Decisions When Costs and Probabilities are Both Unknown", Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, pp. 204-213, 2001.
[75] D. Lizotte, O. Madani and R. Greiner. "Budgeted Learning of Naïve Bayes Classifiers", Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence, pp. 378-385, 2003.
[76] K.M. Ting. "Inducing Cost-Sensitive Trees via Instance Weighting", Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery, pp. 1393-147, 1998.
[77] C.X. Ling, Q. Yang, J. Wang and S. Zhang. "Decision Trees with Minimal Costs", Proceedings of the twenty-first International Conference on Machine Learning (ICML'2004), vol. 69, pp. 544-551, 2004.
[78] B.Zadrozny, J. Langford and N. Abe. "Cost sensitive learning by Cost-Proportionate instance Weighting", Proceedings of the third International Conference on Data Mining, pp. 435, 2003.
[79] V. S. Sheng, C. X. Ling C.X. "Roulette Sampling for Cost-Sensitive Learning", Proceedings of the European Conference on Machine Learning (ECML-2007), vol. 4701, pp. 724-731, 2007.
[80] D. Michie, D. J. Spiegelhalter and C. C. Taylor. "Machine Learning, Neural and Statistical Classification", 1994.
[81] P. Domingos. "MetaCost: A General Method for Making Classifiers Cost-Sensitive", Proceedings of the fifth ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD'99), pp. 155-164, 1999.
[82] J. H. Holland. "Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems", In R. Michalski, J. Carbonell, and T. Mitchell (Eds.),Machine Learning: An Artificial Intelligence Approach, vol. 2, pp. 275-304, 1986.

[83] S. F. Smith. "Flexible learning of problem solving heuristics through adaptive search", Proceedings of the Eighth International Joint Conference on Artificial intelligence (IJCAI'83), vol. 1, pp. 421-425, 1983.

[84] J. Bacardit. "Pittsburgh Genetics-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time", PhD dissertation, 2004.

[85] J. Bacardit and J. M. Garrell. "Evolving multiple discretizations with adaptive intervals for a pittsburgh rule-based learning classifier system", Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2003), pp. 1818-1831, 2003.

[86] J. Rissanen. "Modeling by shortest data description", *Automatica*, vol. 14, no. 5, pp. 465-471, 1978.

[87] J. Bacardit and J. M. Garrell. "Incremental learning for pittsburgh approach classifier systems", Proceedings of the "Segundo Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados.", pp. 303-311, 2003.

[88] T.Soule and J. A. Foster. "Effects of code growth and parsimony pressure on populations in genetic programming", *Evolutionary Computation*, vol. 6, no. 4, pp. 293-309, 1998.

[89] J. Bacardit and J. M. Garrell. "Bloat control and generalization pressure using the minimum description length principle for a pittsburgh approach learning classifier system", Proceedings of the sixth International Workshop on Learning Classifier Systems, 2003.

[90] G. Wu and E.Y. Chang. "Adaptive Feature-Space Conformal Transformation for Imbalanced-Data Learning", Proceedings of the Twentieth on International Conference on Machine Learning (ICML'03), pp. 816-823, 2003.

[91] S. Steward. "Lighting the way in '97", *Cellular Business*, pp. 23, 1997.

[92] T. Fawcett, F. Provost. "Adaptive Fraud Detection, Data Mining and Knowledge Discovery". vol.1, no. 3, pp. 291-316, 1997.

[93] G. Cohen, M. Hilario, H. Sax, S. Hugonnet and A. Geissbühler. "Learning from imbalanced data in surveillance of nosocomial infection", *Artificial Intelligence in Medicine*, vol. 37, no. 1, pp. 7-18, 2006.

[94] K. Ezawa, M. Singh and S. W. Norton. "Learning goal oriented Bayesian networks for telecommunications risk management", Proceedings of the Thirteenth International Conference on Machine Learning, pp. 139-147, 1996.

[95] M. Kubat, R. Holte and S. Matwin. "Machine learning for the detection of oil spills in satellite radar images", *Machine Learning*, vol. 30, no. 2-3, pp. 195-215, 1998.

[96] S. Daskalaki, I. Kopanas and N. Avouris. "Evaluation of classifiers for an uneven class distribution problem", *Applied Artificial Intelligence*, vol. 20, no. 5, pp. 38-417, 2006.

[97] J. Huang and C. X. Ling. "Using AUC and accuracy in evaluating learning algorithms", *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299-310, 2005.

[98] T. C. W. Landgrebe, P. Paclick, R. P. W. Duin and A. P. Bradley. "Precision-recall operating characteristic (P-ROC) curves in imprecise environments", Proceedings of the eighteenth International Conference on Pattern Recognition, vol. 4, pp. 123-127, 2006.

[99] F. Provost and T. Fawcett. "Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions", Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97), pp. 43-48, 1997.

[100] M. Joshi. "On evaluating performance of classifiers for rare classes", Proceedings of IEEE International Conference on Data Mining, pp. 641-644, 2002.

[101] M. Buckland and F. Gey. "The Relationship between Recall and Precision", *American Society for Information Science*, vol. 45, no. 1, pp.12-19, 1994.

[102] M. Kubat and S. Matwin. "Addressing the Curse of Imbalanced Training Sets: One-Sided Selection", Proceedings of the Fourteenth International Conference on Machine Lesarning (ICML'97), pp. 179-186, 1997.

[103] H. Guo and H. L. Viktor. "Learning from Imbalanced Data Sets with Boosting and Data Generation: The Databoost-IM Approach", *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 30-39, 2004.

[104] G. Wu and E.Y. Chang, "KBA: Kernel Boundary Alignment Considering Imbalanced Data Distribution", *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 786-795, 2005.

[105] H. He and E. García. "Learning from imbalanced data", *IEEE Transactions on Data and Knowledge Engineering*, vol. 21, no. 9, pp. 1263-1284, 2009.

[106] J. Zhang and I. Mani, "KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction", Proceeding of International Conference on Machine Learning (ICML 2003), Workshop on Learning from Imbalanced Data Sets, 2003.

[107] C. Drummond and R. Holte. "C4.5, Class Imbalance, and Cost Sensitivity: Why Under-sampling beats Over-sampling", Proceeding of International Conference on Machine Learning (ICML 2003), Workshop on Learning from Imbalanced Data Sets, 2003, pp. 1-8 added

[108] R. C. Holte, L. E. Acker and B. W. Porter. "Concept learning and the problem of small disjuncts", Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89), vol. 1,pp. 813-818, 1989.

[109] D. Mease, A. J. Wyner and A. Buja. "Boosted Classification Trees and Class Probability/Quantile Estimation," *Machine Learning Research*, vol. 8, pp. 409-439, 2007.

[110] X. Y. Liu, J. Wu and Z. H. Zhou. "Exploratory Under Sampling for Class Imbalance Learning", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539-550, 2006.

[111] T. M. Cover and P. E. Hart. "Nearest neighbor pattern classification", *IEEE Transaction on Information Theory*, vol. 13, no. 1, pp. 21-27, 1967.

[112] I. Tomek. "Two Modifications of CNN", *IEEE Transaction System, Man, Cybernetics*, vol. 6, no. 11, pp. 769-772, 1976.

[113] P. Hart. "The condensed nearest neighbor rule", *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515-516, 1968.

[114] D. R. Wilson and T. R. Martinez. "Reduction Techniques for Instance-Based Learning Algorithms", *Machine Learning*, vol. 38, no. 3,pp. 257-286, 2000.

[115] D. W. Aha, D. Kibler and M. K. Albert. "Instance-Based Learning Algorithms", *Machine Learning*, vol. 6, no. 1, pp. 37-66, 1991.

[116] J. Laurikkala. "Improving identification of difficult small classes by balancing class distribution", in: S. Quaglini, P. Barahona and S. Andreassen (eds), Artificial Intelligence in Medicine. AIME 2001, Lecture Notes in Computer Science, vol. 2101, pp. 63-66, 2001.

[117] D. L. Wilson. "Asymptotic properties of nearest neighbor rules using edited data", *IEEE Transactions on Systems Man and Cybernetics*, vol. 2, no. 3, pp.408-421, 1972.

[118] S. Yen, Y. Lee, C. Lin and J. Ying. "Investigating the Effect of Sampling Methods for Imbalanced Data Distributions", IEEE International Conference on Systems, Man, and Cybernetics, pp. 4163-4168, 2006.

[119] L. I. Kuncheva and J. C. Bezdek. "Nearest prototype classification: Clustering, genetic algorithms, or random search?" *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 28, no. 1, pp. 160-164, 1998.

[120] J. R. Cano, F. Herrera and M. Lozano. "Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study", *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 561-575, 2003.

[121] L. J. Eshelman. "The CHC adaptive search algorithm: How to safe search when engaging in nontraditional genetic recombination", in G. J. E. Rawlings (Ed.), Foundations of genetic algorithms, pp. 265-283, 1991.

[122] S. Y. Ho, C. C. Liu, and S. Liu. "Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm". *Pattern Recognition Letters*, vol. 23, no. 13, pp. 1495-1503, 2002.

[123] S. García, J. R. Cano, A. Fernandez and F. Herrera. "A proposal of evolutionary prototype selection for class imbalance problems", International Conference on Intelligent Data Engineering and Automated Learning IDEAL, vol. 4224, pp. 1415-1423, 2006.

[124] A. Colorni, M. Dorigo and V. Maniezzo. "Distributed optimization by ant colonies", Proceedings of the first European Conference on Artificial Life, vol. 142, pp.134-142, 1991.

[125] H. L. Yu, G .C. Gu, H. B. Liu, J. Shen and J. Zhao. "A modified ant colony optimization algorithm for tumor marker gene selection", Genomics Proteomics Bioinformatics, vol. 7, no. 4, pp. 200-208, 2009.

[126] P. Yang, L. Xu, B. Zhou, Z. Zhang and A. Y. Zomaya. "A particle swarm based hybrid system for imbalanced medical data sampling", *BMC Genomics*, vol. 10, no. 3, pp. S34, 2009.

[127] N. V. Chawla, K.W. Bowyer, L. O. Hall and W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique", *Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002.

[128] B. X. Wang and N. Japkowicz. "Imbalanced Data Set Learning with Synthetic Samples", Proceedings of IRIS Machine Learning Workshop, 2004.

[129] S. Hu, Y. Liang, L. Ma and Y. He. "MSMOTE: Improving classification performance when training data is imbalanced", Computer Science and Engineering, International Workshop (IWCSE), vol. 2, pp. 13-17, 2009.

[130] H. Han, W. Wang and B. Mao. "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning", Proceedings of the International Conference on Intelligent Computing (ICIC'05), vol. Part I, pp. 878-887, 2005.

[131] H. He, Y. Bai, E. A. Garcia and S. Li. "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning", Proceedings of the International Joint Conference on Neural Networks, pp. 1322-1328, 2008.

[132] T. Jo and N. Japkowicz. "Class Imbalances versus Small Disjuncts", *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 40-49, 2004.

[133] P. A. Flach. "The Geometry of ROC Space: Understanding Machine Learning Metrics through ROC Isometrics", International Conference on Machine Learning (ICML'2003), pp. 194-201, 2003.

[134] D. A. Cieslak and N. V. Chawla. "Learning decision trees for unbalanced data", in W. Daelemans, B. Goethals and K. Morik (Eds), Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 5211, pp. 241-256, 2008.

[135] B. Raskutti and A. Kowalczyk. "Extreme Re-Balancing for SVMs: A Case Study", ACM *SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 60-69, 2004.

[136] G. Wu and E. Chang. "Class-Boundary Alignment for Imbalanced Data Set Learning", proceedings of International Conference on Data Mining (ICDM'03), Workshop Learning from Imbalanced Datasets II, 2003.

[137] J. ZHANG, E. Eloedorn, L. Rosen and D. Venese. "Learning Rules from Highly Umbalanced Data Set", Proceeding of the fourth IEEE International Conference on Data Mining (ICDM'04), pp. 571-574, 2004.

[138] T. B. Ho, D. Nguyen and S. Kawasaki. "Mining Prediction Rules from Minority Classes", Proceedings of the fourth International Conference on Applications of Prolog, pp. 254-265, 2001.

[139] J. Furnkranz. "Separate-and-conquer rule learning", *Artificial Intelligence Review*, vol. 13, no. 1, pp. 3-54, 1999.

[140] Z. H. Zhou. "Ensemble methods: foundations and algorithms", 2012.

[141] M. Galar, A. Fernández, E. Barrenechea, H. Bustince and F. Herrera. "A review on Ensembles for the class Imbalance Problem: Bagging, Boosting and Hybrid-Based Approaches", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 4, pp. 463-484, 2012.

[142] N. V. Chawla, A. Lazarevic, L. O. Hall and K. W. Bowyer. "SMOTEBoost: Improving prediction of the minority class in boosting", in N. Lavrač, D. Gamberger, L. Todorovski and H. Blockeel, (eds) Knowledge Discovery in Databases: PKDD 2003, Lecture Notes in Computer Science, vol. 2838 , pp. 107-119, 2003.

[143] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse and A. Napolitano. "RUSBoost: A hybrid approach to alleviating class imbalance", *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 1, pp. 185-197, 2010.

[144] S. Wang and X. Yao. "Diversity analysis on imbalanced data sets by using ensemble models", IEEE Symposium Series on Computational Intelligence and Data Mining (IEEECIDM 2009) 2009, pp. 324-331, 2009.

[145] R. Barandela, R. Valdovinos and R. Sánchez. "New applications of ensembles of classifiers", Pattern *Analysis and Applications*, vol. 6, no. 3, pp. 245-256, 2003.

[146] Y. Sun, M. S. Kamel, A. K. C. Wong and Y. Wang. "Cost-sensitive boosting for classification of imbalanced data", *Pattern Recognition*, vol. 40, no. 12, pp. 3358-3378, 2007.

[147] R. Akbani, S. Kwek, and N. Japkowicz. "Applying Support Vector Machines to Imbalanced Data Sets," Lecture Notes in ComputerScience, vol. 3201, pp. 39-50, 2004.

[148] K. Veropoulos, C. Campbell and N. Cristianini. "Controlling the sensitivity of support vector machines", Proceedings of the International Joint Conference on AI, pp. 55-60, 1999.

[149] F. Vilarino, P. Spyridonos, P. Radeva and J. Vitria. "Experiments with SVM and Stratified Sampling with an Imbalanced Problem: Detection of Intestinal Contractions", in: S. Singh, M. Singh, C. Apte and P. Perner (eds), Pattern Recognition and Image Analysis. ICAPR 2005, Lecture Notes in Computer Science, vol. 3687, pp. 783-791, 2005.

[150] P. Kang and S. Cho. "EUS SVMs: Ensemble of Under sampled SVMs for Data Imbalance Problems", In: I. King, J. Wang, LW. Chan and D. Wang (eds), Neural Information Processing, ICONIP 2006. Lecture Notes in Computer Science, vol. 4232, pp. 837-846, 2006.

[151] B. X. Wang and N. Japkowicz. "Boosting Support Vector Machines for Imbalanced Data Sets", In: A. An, S. Matwin, Z. WRaś, D. Ślęzak (eds) Foundations of Intelligent Systems. ISMIS 2008? Lecture Notes in Computer Science, vol. 4994, pp. 38-47, 2008.

[152] M. Golea. "On the complexity of rule extraction from neural networks and network querying", proceedings of the Rule Extraction From Trained Artificial Neural Networks Workshop, Society For the Study of Artificial Intelligence and Simulation of Behavior Workshop Series (AISB'96), pp.51-59, 1996.

[153] R. R. Yager and L. A. Zadeh. "Fuzzy Sets, Neural Networks and Soft Computing", 1994.
[154] Z. Pawlak. "Rough Sets: Theoretical Aspects of Reasoning about Data", 1991.
[155] P. Clark and T. Niblett. "The cn2 induction algorithm", *Machine Learning*, vol. 3, no. 4, pp. 261-283, 1989.
[156] **A.** Asuncion and D. J. Newman. "UCI Machine Learning Repository", [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science, 2007.
[157] R. S. Michalski. "A theory and methodology of inductive learning", *Artificial Intelligence*, vol. 20, no. 2, pp. 111-161, 1983.
[158] R. S. Michalski and K. A. Kaufman. "Data mining and knowledge discovery: A review of issues and a multi-strategy approach", in: R. S. Michalski, I. Bratko, A. Bratko (Eds.), Machine Learning and Data Mining: Methods and Applications, pp. 71-112, 1998.
[159] V. Dhar, D. Chou and F. Provost. " Discovering interesting patterns for investment decision making with GLOWER - a Genetic Learner Overlaid with Entropy Reduction", Data *Mining and Knowledge Discovery*, vol. 4, no. 4, pp. 251-280, 2000.
[160] S. Benkhider and H. Drias. "A New Memetic Approach for the classification rules extraction problem", *Data Mining, Modeling and Management*, vol. 5, no. 4, pp 318-332.
[161] G. Syswerda. "A Study of Reproduction in Generational and Steady-State Genetic Algorithms", Foundations of genetic algorithms, vol. 1, pp. 94-101, 1991.
[162] A. Mahani, A. R. Baba-Ali. "A new rule-based knowledge extraction approach for imbalanced datasets". Knowledge and Information Systems. DOI: 10.1007/s10115-019-01330-9, 2019.
[163] S. Geisser. "The predictive sample reuse method with applications", *American Statistical Association*, vol. 70, no. 350, pp. 320-328, 1975.
[164] A. Mahani, S. Benkhider and A.R. Baba-Ali. "Extraction Of Classification Rules from imbalanced datasets using GA_ILS. Proceeding of the 14th International Conference on Applied Computing (AC 2017), pp. 289-294, 2017.
[165] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. Fernández and F. Herrera. "KEEL: A software tool to assess evolutionary algorithms for data mining problems", *Soft Computing*, vol. 13, no. 3, pp. 307-318, 2008.
[166] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez and F. Herrera. "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework", *Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255-287, 2011.
[167] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan,A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, D. Steinberg."Top 10algorithms in data mining", *Knowledge and Information Systems* 14, pp. 1-37, 2007.
[168] W. W. Cohen. "Fast effective rule induction", Proceedings of the Twelfth International Conference on Machine Learning, pp. 115-123, 1995.
[169] E. Frank and I. H. Witten. "Generating accurate rule sets without global optimization", Proceedings of the fifteenth International Conference on Machine Learning, pages 144-151, 1998.
[170] C. Stanfill and D. Waltz. "Toward Memory-based Reasoning", *Communications of the ACM*, vol. 29, no. 12, pp. 1213-1228, 1986.
[171] C. Stanfill. "Memory-based reasoning applied to English pronunciation", Proceedings of the Sixth National Conference on Artificial Intelligence, pp. 577-581, 1987.

[172] J. Demšar. “Statistical comparisons of classifiers over multiple data sets”, *Machine Learning Research*, vol. 7, pp. 1-30, 2006.
[173] F. Wilcoxon. “Individual comparisons by ranking methods”, *Biometrics Bulletin*, vol. 1, no. 6, pp. 80-83, 1945.
[174] J. H. Zar. “Biostatistical Analysis”, 1999.
[175] M. Friedman. “The use of ranks to avoid the assumption of normality implicit in the analysis of variance”, *American Statistical Association*, vol. 32, no. 200, pp. 675-701, 1937.
[176] M. Friedman. “A comparison of alternative tests of significance for the problem of m rankings”, *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86-92, 1940.