**PROJET REALISE PAR :**

Khattabi Idriss

Boufarhi Ayman

Sabahhi Mohamed Amine

**ENCADRE PAR :**

M'hamed AIT KBIR

**Sommaire :**

**1- Description des données**

**2- Synthèse des Travaux de Recherche**

**3- Pré-traitement des données**

**4- Implémentation des techniques d’apprentissage automatique**

**4.1- Implémentation de XGBoost**

**4.2- Implémentation de Gaussian Naive Bayes**

**5- Présentation des résultats les différentes métriques**

**6- Comparer les résultats trouvés avec ceux issus des implémentations fournies par certains modules des librairies Python**

**7- Conclusion**

**8- Bibliographie**

# **Description des données**

Le jeu de données Arcene, constitue une ressource cruciale dans le domaine de l'apprentissage automatique appliqué à la classification binaire du cancer de l'ovaire ou de la prostate, ce jeu de données offre un aperçu détaillé des caractéristiques moléculaires capturées à travers la spectrométrie de masse. Les données incluent des mesures d'intensités de spectres de masse, qui sont essentielles pour la distinction entre les échantillons sains et ceux associés au cancer. Cette granularité moléculaire permet de construire des modèles de classification sophistiqués et précis

Le jeu de données Arcene, résulte de la fusion de trois ensembles de données de spectrométrie de masse, fournissant une abondance de données d'entraînement et de test. Il se compose de caractéristiques originales détaillant l'abondance des protéines dans le sérum humain en fonction de la masse, visant à différencier les patients atteints de cancer des patients en bonne santé. Des caractéristiques distrayantes, appelées "sondes," ont été ajoutées sans pouvoir prédictif, et l'ordre des caractéristiques a été aléatoirement mélangé.

Réparties en ensembles d'entraînement, de validation et de test, les données comprennent environ 10 000 variables, dont 7 000 sont réelles et 3 000 sont des sondes. Le jeu de données est un défi de sélection de caractéristiques NIPS 2003.

Le tableau ci-dessous illustre la distribution des exemples positifs et négatifs dans chaque ensemble :

| **ARCENE** | **Positifs** | **Négatifs** | **Total** |
| --- | --- | --- | --- |
| Entraînement | 44 | 56 | 100 |
| Validation | 44 | 56 | 100 |
| Test | 310 | 390 | 700 |
| Total | 398 | 502 | 900 |

Le format des données est structuré en plusieurs fichiers, notamment des fichiers de paramètres, d'identités de caractéristiques, d'ensembles d'entraînement, de validation et de test, ainsi que des fichiers d'étiquettes indiquant les vérités de classes :

• dataname.param : Ce fichier contient les paramètres et les statistiques détaillées sur les données. Il offre un aperçu global des propriétés du jeu de données.

• dataname.feat : Ce fichier identifie les caractéristiques, cependant, ces informations sont délibérément retirées pour éviter tout biais potentiel dans le processus de sélection des caractéristiques. Cela garantit une évaluation impartiale des modèles.

• dataname\_train.data : Il s'agit du fichier décrivant l'ensemble d'entraînement. Les données sont présentées sous forme d'une matrice régulière délimitée par des virgules, avec les motifs disposés en lignes et les caractéristiques en colonnes. Ces données constituent la base sur laquelle les algorithmes d'apprentissage automatique seront formés.

• dataname\_valid.data : Similaire au fichier d'entraînement, ce fichier contient l'ensemble de validation. Il est utilisé pour évaluer les performances des modèles pendant la phase de développement.

• dataname\_test.data : Ce fichier décrit l'ensemble de test, une partie cruciale du processus d'évaluation. Les modèles formés sur l'ensemble d'entraînement sont testés sur ces données pour évaluer leur généralisation aux nouvelles observations.

• dataname\_train.labels : Les étiquettes associées aux exemples de l'ensemble d'entraînement. Elles représentent les valeurs de vérité des classes auxquelles appartiennent les exemples.

• dataname\_valid.labels : Les étiquettes de l'ensemble de validation. Ces informations étaient retirées pendant le défi, mais elles sont maintenant fournies pour une analyse approfondie.

• dataname\_test.labels : Les étiquettes de l'ensemble de test. Bien que retirées, cela permet l'utilisation des données comme référence pour les performances des modèles.

# **2- Synthèse des Travaux de Recherche**

La dataset "arcene" occupe une place significative dans la recherche en bioinformatique et en apprentissage automatique, en particulier en raison de son utilisation dans le cadre du défi KDD Cup 2003. Ce jeu de données, provenant d'une étude sur la classification du cancer basée sur des données de spectrométrie de masse, a été le centre de nombreux travaux de recherche visant à développer des modèles d’apprentissage automatique capables de distinguer entre tissus sains et tissus cancéreux.

Les chercheurs ont exploré diverses approches d'apprentissage automatique, notamment des algorithmes de classification tels que les machines à vecteurs de support (SVM), les réseaux de neurones, et les méthodes d'ensemble comme les forêts aléatoires. L'objectif commun était d'optimiser la précision de la classification afin d'améliorer la détection précoce du cancer.

Les travaux de recherche ont également porté sur l'extraction et la sélection de caractéristiques pertinentes à partir des données de spectrométrie de masse, cherchant à identifier les signatures moléculaires distinctives associées à différents types de tissus. Ces efforts ont contribué à une compréhension plus approfondie des patterns biologiques impliqués dans le développement du cancer.

Par ailleurs, la dataset arcene a été utilisée comme point de comparaison pour évaluer la performance de nouvelles méthodes et techniques émergentes dans le domaine de l'apprentissage automatique. Les résultats de ces travaux ont souvent été publiés dans des revues scientifiques spécialisées, contribuant ainsi à l'avancement des connaissances dans la compréhension et la classification des données liées au cancer à partir de la spectrométrie de masse.

En résumé, la dataset arcene a joué un rôle crucial en fournissant un banc d'essai standardisé pour évaluer les algorithmes de classification dans le contexte de la détection du cancer. Les travaux de recherche associés ont contribué à l'amélioration des méthodes diagnostiques et ont ouvert la voie à de nouvelles avenues pour la recherche en bio-informatique et en oncologie.

# **3- Pré-traitement des données :**

Dans la phase de prétraitement de l'ensemble de données « Arcene », la première étape consiste à charger les données à l'aide de la bibliothèque « pandas ». Et aussi, les données d’Arcene sont déjà devisé en l’ensemble de formation et l’ensemble de validation ou de teste pour évaluer les performances du modèle, et chaque de ces l’ensembles sont déja séparées on les caractéristiques (X) et les étiquettes (y), tel que les fichiers ‘arcene\_train.data’ et ‘arcene\_train.labels’ représente correspondant les caractéristiques d’entrainement (X\_train) et les étiquettes d’entrainement (y\_train), la même chose pour l’ensemble de teste par les fichiers ‘arcene\_valid.data’ (X\_test) et ‘arcene\_valid.labels’ (y\_test). Donc l’étape de séparation des données et de la division en ensembles de formation et de teste sont déjà inclue dans la 1ére étape.

Et Comme nous le savons que les données de « Arcece » ont 10000 variables, dont 7 000 sont réelles et 3 000 sont des sondes. Donc le 2éme étape consiste à sélectionnée les features la plus relevant et éliminer les redondant. Cette étape peut être bénéfique pour améliorer les performances du modèle, réduire la complexité des calculs et garantir la qualité des résultats. Pour réaliser ce processus, on va utiliser les méthodes de filtrage dans la catégorie de sélection des features, par exemple : Corrélation, variance de seuil, chi-carré, Anova et gain d’information…, et tout ces fonctions sont existées dans la bibliothèque de « Sklearn.feature\_selection ». Dans cette étape on va utiliser notre propre méthode de la sélection des features, on supprime certaines colonnes inutiles, si 50 % ou plus des valeurs dans une colonne sont 0.

La dernière étape est la standardisation des fonctionnalités. Le StandardScaler est appliqué pour mettre à l'échelle les fonctionnalités, en garantissant qu'elles ont une moyenne de zéro et un écart type de 1.

# **4- Implémentation des techniques d’apprentissage automatique :**

Dans la phase de l’implémentation des algorithmes d’apprentissage automatique, on va implémenter l’algorithme « **XGBoost »** et l’algorithme de **« Naive Bayse ».**

## 4.1- Implémentation de XGBoost :

L’algorithme de « XGBoost », qui signifie « eXtreme Gradient Boosting », est un algorithme d'apprentissage automatique populaire et puissant utilisé à la fois pour les tâches de régression et de classification. XGBoost appartient à la famille des méthodes d'apprentissage d'ensemble. L'apprentissage d'ensemble implique la formation de plusieurs modèles et la combinaison de leurs prédictions pour améliorer les performances globales. Il s'agit d'une implémentation du « Gradient Boosting », conçu pour la vitesse et les performances. XGBoost est largement utilisé dans diverses compétitions d'apprentissage automatique et est devenu un algorithme incontournable pour les données structurées/tabulaires.

Comme nous avons dit que XGBoost appartient à la famille des méthodes d'apprentissage par ensemble, donc il construit une collection d'apprenants faibles (les arbres de décision) et les combine pour créer un apprenant fort.

Avec XGBoost est basé sur le « gradient boosting », qui construit des modèles de manière séquentielle, chacun corrigeant les erreurs commises par les précédents. Il minimise une fonction de perte (loss function) prédéfinie en ajoutant des apprenants faibles, chaque nouvel apprenant se concentrant sur les erreurs de l'ensemble existant.

* Voici un bref aperçu du fonctionnement de XGBoost :

Après d’initialisation du nombre des arbres de décision pour créé (n\_estimators), et le taux d’apprentissage (learning rate) et le nombre maximal de la profondeur de l’arbre (max\_depth), on peut crée un modèle de XGBoost.

1. **Conception d'un Arbre de Décision (Apprenant Faible) :**

* **Un arbre de décision est construit pour ajuster les résidus (différences entre les prédictions actuelles et les vraies valeurs).**
* **La construction de l'arbre implique le choix des points de division en fonction des critères tels que mse (mean squire error), Entropy, gain d’information ou Gini index...**
* **Le seuil de la colonne est la moyenne de la colonne.**

1. **Calcul de la Perte Régularisée (fonction objective) :**

* La perte régularisée est calculée pour l'arbre nouvellement créé.
* Elle est composée de la perte (erreur) et des termes de régularisation qui pénalisent la complexité de l'arbre et du modèle global.

**Objective**

• **n** : est le nombre d'exemples de formation.

• **:** est la fonction de perte qui mesure la différence entre le véritable label yi et le label prédit .

• K : est le nombre d'apprenants faibles (arbres) dans l'ensemble.

• Ω () : est le terme de régularisation pour le k-ième arbre.

1. **Mise à jour du modèle :**

* L'arbre nouvellement créé est ajouté à l'ensemble existant de modèles avec un poids déterminé par la descente de gradient.
* La prédiction du modèle est mise à jour en ajoutant la prédiction de l'arbre.
* Mise à jour **les résidus (différences entre les prédictions actuelles et les vraies valeurs).**

1. **Répétition :**

* Les étapes 1 à 3 sont répétées jusqu'à ce qu'un nombre d'arbres prédéfini soit construit (n\_estimators).

1. **Prédiction :**

* La prédiction finale pour une nouvelle entrée xx est la somme des prédictions de tous les arbres de l'ensemble :

**(x)=**

• η est le taux d'apprentissage, contrôlant la taille du pas pendant l'optimisation.

• (x) est la prédiction du k-ième arbre pour l'entrée x.

* l'addition des prédictions de tous les arbres de l'ensemble, chacune étant mise à l'échelle en fonction du taux d'apprentissage.

Dans l'ensemble, la combinaison de XGBoost d'apprentissage d'ensemble, d'augmentation de gradient, de régularisation et d'autres techniques d'optimisation en fait un algorithme robuste et efficace pour un large éventail de tâches d'apprentissage automatique. Sa popularité peut être attribuée à ses performances, sa flexibilité et sa capacité à gérer divers ensembles de données.

## 4.2- Implémentation de Gaussian Naive Bayes :

# **5-Présentation des résultats les différentes métriques :**

# **Comparer les résultats trouvés avec ceux issus des implémentations fournies par certains modules des librairies Python :**

# **Conclusion :**

# **Bibliographie :**

[1] Nand Sharma; Prathamesh Verlekar; Rehab Ashary; Sui Zhiquan, Regularization and feature selection for large dimensional data.

[2] <https://www.geeksforgeeks.org/xgboost/>

[3] <https://simonwenkel.com/2019/03/09/revisiting-ML-datasets-arcene.html>

[4]