AI28 - MACHINE LEARNING
RAPPORT DE PROJET

Anonymous authors
Paper under double-blind review

ABSTRACT

Le dataset utilisé dans le cadre de ce projet, récupéré du site web `https://www.kaggle.com/`, a été collecté par le Service National d'Assurance Maladie en Corée. Toutes les informations personnelles et les données sensibles en ont été exclues. L'objectif de ce jeu de données est d'analyser les signaux corporels et d'en déduire une classification des fumeurs et/ou buveurs.

1 ANALYSE EXPLORATOIRE DE DONNÉES

1.1 VARIABLES ET ANALYSE UNIVARIÉE

Le dataset initial comporte 991 320 entrées. Deux variables cibles peuvent être identifiées :

- **Variable cible 1: DRK_YN**
  - *Type:* Catégorielle (binaire)
  - *Description:* indique si l'individu boit ou non
- **Variable cible 2: SMK_stat_type_cd**
  - *Type:* Catégorielle (à trois classes)
  - *Description:* 1 (non-fumeur), 2 (ancien fumeur), 3 (fumeur)

Pour nos modèles, nous avons choisi de nous concentrer sur la variable cible à trois classes **SMK_stat_type_cd**. Notons que les classes de cette dernière sont déséquilibrées, ce qui a par la suite été pris en compte.

Le dataset est composé de 22 variables explicatives, listées en détail en annexe (voir Annexe A). La plupart des données sont brutes et ne semblent pas avoir été prétraitées (très peu de distributions dont la moyenne est 0 et l'écart-type 1). Nous avons cependant validé à l'aide de tests d'hypothèses que certaines d'entre elles peuvent être associées à une loi normale (une illustration est donnée en Annexe B).

1.2 ANALYSE BIVARIÉE

L'analyse bivariée nous a permis de formuler des hypothèses quant aux relations entre les variables explicatives avec la variable cible et entre elles.

Tout d'abord, nous avons pu remarquer graphiquement quelques relations linéaires entre les variables, soutenues par des scores de corrélation assez proches de $|1|$. De manière assez évidente, *weight* et *height* partagent une corrélation positive (avec un score de 0,67), ainsi que *weight* et *waistline* (0,64). C'est également le cas du couple *SBP*/*DBP* (0,74). Quelques recherches supplémentaires nous ont confirmé que les pressions artérielles systoliques et diastoliques ont en effet tendance à partager une corrélation positive.

Nous avons également identifié des relations entre les variables explicatives et la variable cible *SMK_stat_type_cd*.

Les variables qualitatives ayant des classes fortement déséquilibrées, sauf pour une d'entre elles (*sex*), il nous a été possible d'identifier un pattern uniquement pour cette dernière. Nous pouvons

constater assez clairement sur la figure 1 que très peu de femmes sont fumeuses/anciennes fumeuses et qu'un fumeur a donc plus de probabilité d'être un homme qu'une femme. Cela est confirmé par les données de l'OCDE, qui indiquent que la Corée du Sud enregistre un des taux de tabagisme les plus bas pour les femmes.

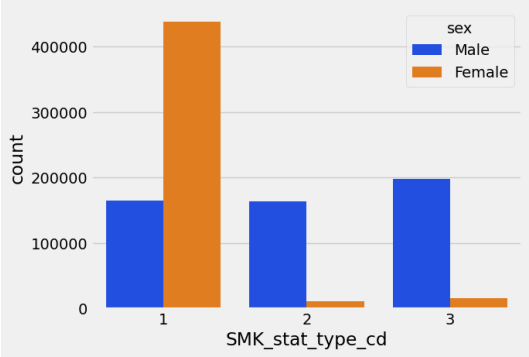

Figure 1: Relation entre le sexe et les habitudes de consommation de tabac (1 : non-fumeur, 2 : ancien fumeur, 3 : fumeur)

Parmi les variables quantitatives, les distrbutions conditionnelles de certaines variables par rapport à la variable cible ont supposé une corrélation entre elles. Un exemple est donné en figure 2 pour la variable *hemoglobin*, dont la distribution diffère entre les non-fumeurs et les autres individus. Aucune différence suffisamment notable ne peut cependant être remarquée entre les fumeurs et les anciens fumeurs.

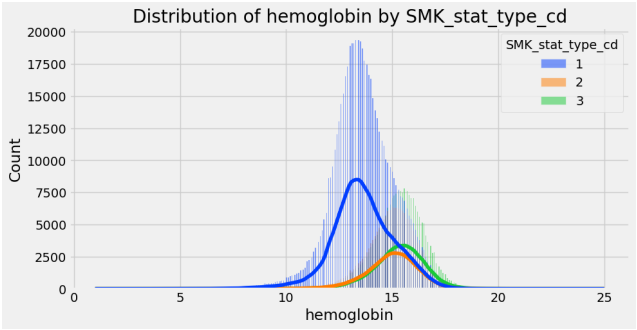

Figure 2: Distribution conditionnelle de *hemoglobin* sachant la variable cible *SMK_stat_type_cd* (1 : non-fumeur, 2 : ancien fumeur, 3 : fumeur)

2 PRÉ-TRAITEMENT DES DONNÉES

2.1 ECHANTILLONNAGE

Notre dataset étant assez conséquent et par souci de performance et temps de calcul (notamment pour l'optimisation des modèles), nous avons choisi de réduire sa taille. Nous en avons profité pour rééquilibrer les données étudiées en construisant un échantillon avec autant de samples pour chaque classe que la classe la moins représentée, car la variable cible étudiée *SMK_stat_type_cd* avait des classes déséquilibrées (22% de fumeurs et 18% d'anciens fumeurs contre 60% de non-fumeurs). En effet, il ne nous semblait pas pertinent de générer des observations avec SMOTE ou d'en éliminer avec NearMiss étant donnée la richesse des données du dataset initial.

2.2 NETTOYAGE ET ENCODAGE

Nous avons identifié assez peu d'outliers pour décider de ne pas les écarter, le nombre de données nous permettant une certaine sécurité quant au biais qu'ils auraient pu causer. Nous n'avons non

plus pas eu besoin de remplacer les valeurs manquantes (il n'y en avait pas). Les duplicatas ont été conservés car nous avons considéré qu'il pouvait s'agir de réels cas de doublons (rappelons que les variables *age*, *weight*, *height* ont été tronquées à 5 unités près). Leur nombre était par ailleurs négligeable par rapport à la taille du dataset.

Quant à l'encodage, nous avons utilisé la standardisation pour les variables quantitatives (validant notre remarque quant aux valeurs aberrantes, nous avons remarqué que l'utilisation du *RobustScaler()* plutôt que le *StandardScaler()* n'impactait en rien les performances des modèles) et un encodage ordinal pour les variables qualitatives (celles-ci étant binaires ou ordinales).

2.3 SÉLECTION DE VARIABLES

Pour certains de nos modèles, nous avons effectué une sélection préliminaire des k features les plus importantes. Nous avons utilisé pour cela la fonction *SelectKBest* en choisissant une fonction de score adaptée à notre contexte de classification multi-classe (F-value de l'analyse de la variance ANOVA). En faisant une validation croisée, nous avons choisi k=12, valeur à partir de laquelle le score semblait se stabiliser (cf figure 3)

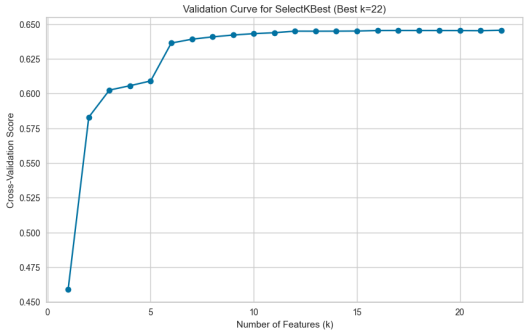

Figure 3: Scores de validation croisée pour différentes valeurs de k dans le cadre de la sélection préliminaire de variables

Une autre approche utilisée pour le modèle des K-plus-proches-voisins (KNN) a été de réduire la dimension du dataset. La motivation derrière cela était double : réduire le temps de calcul du modèle en diminuant la dimension de l'espace de recherche, et améliorer la performance en éliminant les caractéristiques redondantes ou non informatives. Pour ce faire, nous avons appliqué l'Analyse en Composantes Principales (PCA) pour sélectionner les composantes qui expliquent 95% de la variance totale des données. Cela a permis de réduire le nombre de caractéristiques de 22 à 15 composantes principales. Cette réduction est cohérente avec les corrélations identifiées lors de l'analyse exploratoire des données (AED).

3 MODÉLISATION ET OPTIMISATION

3.1 RÉGRESSION LOGISTIQUE

3.1.1 MODÈLE CLASSIQUE

Nous avons tout d'abord entraîné un modèle simple de régression logistique multi-classe sur les données, sans pénalisation. Les résultats sans optimisation sont donnés par la matrice de confusion 1. Pour l'évaluation des modèles, nous avons utilisé le score F1 (adapté selon le contexte : classes équilibrées), afin de donner un poids égal au rappel et à la précision. En effet, nous avons décidé que dans notre contexte, toutes les erreurs de classification avaient une importance égale. Pour ce premier modèle de régression, nous avons **F1 = 0.65**.

Table 1: Matrice de confusion - régression logistique sans pénalité ni optimisation

Predicted labels	**1**	**2**	**3**
True labels			
1	25 373	4912	4705
2	2181	21 059	11 750
3	2638	11 685	20 668

3.1.2 AJOUT D'UNE PÉNALISATION

Afin de vérifier si le modèle pouvait être optimisé, nous avons testé l'ajout des différentes pénalisations Ridge, Lasso et ElasticNet. Le meilleur résultat était donné par la régression Ridge, pour un score **F1 = 0.65** et une matrice de confusion très similaire à celle sans pénalisation (1).

3.1.3 OPTIMISATION DES MODÈLES

Nous avons effectué l'hyper-optimisation des paramètres avec *GridSearchCV*, mais également le framework *Optuna*. Celui-ci utilise plusieurs algorithmes d'optimisation, dont celui par défaut *Tree-structured Parzen Estimator (TPE)*. Il s'agit d'une méthode de recherche bayésienne qui construit des estimations de densité de probabilité pour les bonnes et les mauvaises configurations d'hyperparamètres et utilise ces estimations pour proposer de nouvelles configurations (tout en balaçant exploration et exploitation dans l'espace de recherche).

Dans notre cas, l'espace de recherche était composé du type de pénalisation, de sa valeur C et du $l1\_ratio$. Le modèle le plus performant ayant été retourné par l'algorithme est une régression avec pénalisation Ridge et C = 0.0106. Cependant, aucune des méthodes utilisées n'a permis d'améliorer de manière considérable les résultats, comme nous pouvons le constater par exemple sur la courbe de validation en figure 4. Cependant, cette dernière montre que nos résultats sont satisfaisants, dans le sens où les résultats obtenus sur les train et test sets sont très proches (pas de situation d'under ou d'over-fitting).

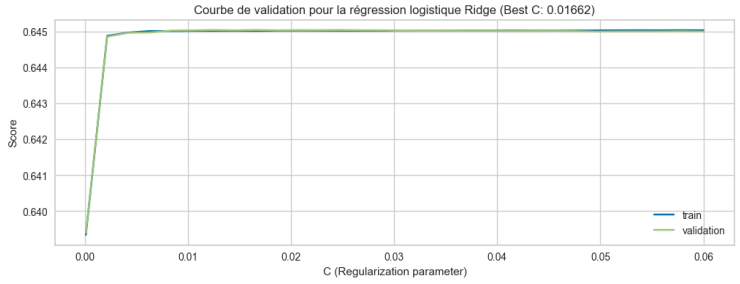

Figure 4: Courbe de validation pour la régression Ridge

3.1.4 AMÉLIORATION DES RÉSULTATS

Un élément ayant attiré notre attention dans la matrice de confusion est le fait que les résultats erronés se trouvaient principalement entre les classes fumeurs et ancien-fumeurs. En lien avec l'observation faite pendant l'AED (cf figure 2), nous avons formulé l'hypothèse qu'il n'était pas évident pour un algorithme de classification de distinguer un fumeur d'un ancien fumeur. Dans ce sens, nous avons fusionné les deux classes correspondantes et entraîné de nouveaux nos modèles. Après optimisation, nous avons obtenu des résultats bien supérieurs (**F1 = 0.83**), confirmant notre intuition.

Les coefficients des modèles de régression (cf figure 5) ont également validé l'hypothèse formulée durant l'AED de l'importance du sexe dans la prédiction des habitudes de tabagisme.

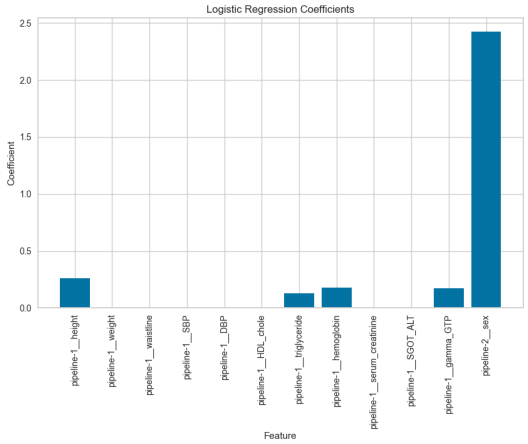

Figure 5: Coefficients d'une régression logistique simple pour la classification binaire : "A été / est fumeur" VS "N'a jamais été fumeur"

3.2 Machines à vecteurs de support (SVM)

3.2.1 Modèle classique

Nous avons tout d'abord entraîné 3 modèles de SVM multi-classe selon les trois types de noyaux : **linéaire**, **gaussien** et **polynomial**. Cet entraînement a été effectué uniquement sur 100k lignes pour cause de temps computationnel important. Dans un premier temps, nous avons fait le choix de ne pas prendre en compte le déséquilibre des classes. Voici un exemple de résultat pour le noyau gaussien (2).

Table 2: Matrice de confusion - SVC linéaire sans class_weight

Predicted labels	**1**	**2**	**3**
True labels			
1	9 500	1 100	1 400
2	500	1 450	1 500
3	800	700	2 800

Nous avons utilisé le score F1 pour évaluer nos modèles, afin de donner un poids égal au rappel et à la précision. Pour ces modèles de SVC, notre score F1 est de **0.69** environ sauf pour le modèle à noyau gaussien RBF avec un score F1 légèrement supérieur à **0.70**. On remarque que la classe majoritaire est prédite avec une bien meilleure précision que les autres, ce qui nous amène à nous demander si les résultats seront significativement différents avec une prise en compte du déséquilibre des classes.

3.2.2 Prise en compte du déséquilibre des classes

Afin de vérifier si le modèle pouvait être optimisé en prenant en compte le déséquilibre des classes, nous avons entraîné ces trois mêmes modèles SVC avec l'hyperparamètre *class_weight='balanced'*. Les résultats sont présentés dans la matrice de confusion pour le modèle à noyau gaussien RBF (3).

Pour ce modèle ajusté, notre score F1 est très légèrement amélioré à **0.706**. Nous remarquons également que la classe majoritaire est encore mieux prédite qu'auparavant, mais que ce n'est pas le cas de la classe 2 et 3.

Table 3: Matrice de confusion - SVM avec class_weight

Predicted labels	**1**	**2**	**3**
True labels			
1	8 800	1 560	1 700
2	200	2 100	1 200
3	300	1 300	2 700

3.2.3 OPTIMISATION DES MODÈLES

Nous avons effectué l'hyper-optimisation des paramètres avec *GridSearchCV* et le framework *Optuna*. Les paramètres optimisés incluaient le paramètre de régularisation C, le paramètre du noyau γ pour le SVM RBF et $degree$ pour le SVM polynomial.

Le temps de calcul important ne nous a pas permis d'aller au bout de toutes les exécutions, mais les scores F1 obtenus étaient légèrement inférieurs aux modèles non optimisés, dû au choix de plage des hyperparamètres. Afin de régler ce problème, tracer les courbes de validation en fonction des hyperparamètres cités précédemment nous aurait permis de choisir des plages de valeurs plus pertinentes.

3.2.4 ÉVALUATION

Les résultats finaux montrent une amélioration marginale des performances avec l'utilisation de *class_weight* et l'optimisation des hyperparamètres. Cependant, l'augmentation significative du temps de calcul doit être prise en compte. Les temps de calcul sont passés de 3 minutes à 7 heures, pour un gain de performance de seulement 0.6% sur le score F1.

Table 4: Comparaison des scores F1 moyen sur les 3 modèles

Modèle	**Score F1 moyen**
SVM sans class_weight	0.69
SVM avec class_weight	0.70
SVM sans class_weight optimisé	0.685

La courbe d'apprentissage du modèle au noyau linéaire (6) montre que les performances du modèle sur les jeux d'entraînement et de validation sont proches, ce qui indique un bon comportement de généralisation. Les courbes révèlent que les scores de validation augmentent régulièrement avec la taille de l'échantillon d'entraînement, mais le score de validation reste légèrement inférieur au score d'entraînement, suggérant un léger surapprentissage pour le noyau linéaire.

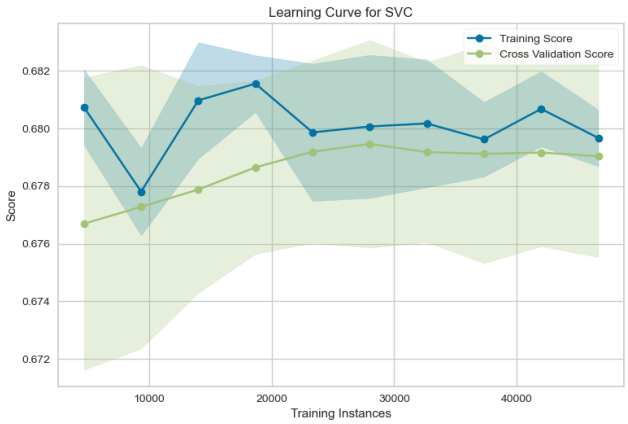

Figure 6: Courbe d'apprentissage pour le SVM linéaire

Pour la prise en compte des poids pour le modèle SVM linéaire, la courbe d'apprentissage en figure 7 montre que les performances du modèle sur les jeux d'entraînement et de validation se stabilisent à mesure que la taille de l'échantillon d'entraînement augmente. Les courbes se croisent plusieurs fois, indiquant une certaine instabilité ou des variations dues à la sélection des ensembles de validation. Vers la fin, les courbes convergent, indiquant une meilleure généralisation du modèle.

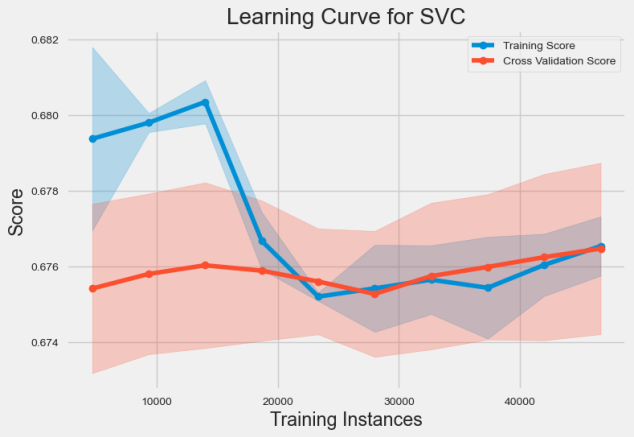

Figure 7: Courbe d'apprentissage pour le SVM linéaire avec class_weight

En conclusion, bien qu'une meilleure optimisation des hyperparamètres et la prise en compte du déséquilibre des classes améliorent les performances du modèle SVM, ces améliorations doivent être mises en balance avec le coût computationnel accru.

3.3 Apprentissage par ensemble - Bagging et Boosting

3.3.1 Modèles

Pour cette section, nous avons utilisé deux modèles d'apprentissage par ensemble : RandomForestClassifier et GradientBoostingClassifier. Les deux modèles ont été évalués sur les mêmes jeux de données et les performances ont été comparées.

3.3.2 RandomForestClassifier

Nous avons tout d'abord entraîné un modèle de RandomForestClassifier sur les données. Le modèle a été évalué en utilisant une matrice de confusion (5) et le score F1 obtenu est de **0.70** en prenant en compte les classes déséquilibrées mprise en compte nous obtenons un score F1 est de **0.68**.

Table 5: Matrice de confusion - RandomForestClassifier

Predicted labels	**1**	**2**	**3**
True labels			
1	8 800	1 620	1 680
2	210	2 080	1 260
3	350	1 300	2 700

Au niveau de la courbe d'apprentissage, le score de validation croisée reste stable autour de 0.675, sans amélioration significative malgré l'augmentation de la taille de l'échantillon. Cette tendance suggère que le modèle de RandomForestClassifier est en situation de surapprentissage : il apprend très bien les données d'entraînement, mais ne parvient pas à généraliser efficacement aux nouvelles données.

Pour remédier à ce problème, nous pouvons envisager de réduire la complexité du modèle en ajustant des hyperparamètres tels que *max_depth*, *min_samples_split* et *min_samples_leaf*, ou en utilisant une méthode de régularisation pour améliorer la généralisation.

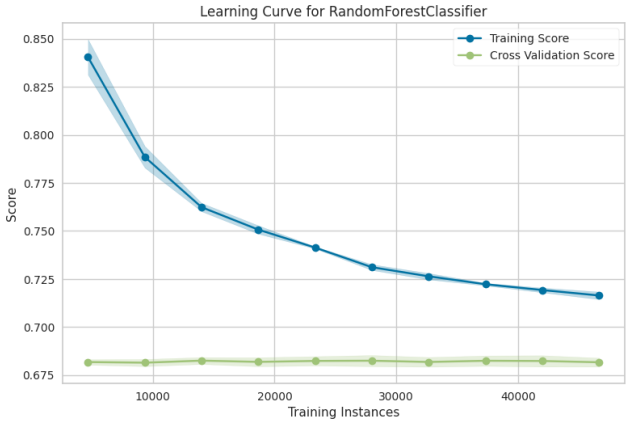

Figure 8: Courbe d'apprentissage pour RandomForestClassifier

La courbe de validation 9 sur le paramètre n_estimators confirme le sur-apprentissage et nous indique également que ce sont les autres hyperparamètres des arbres qui nécessitent une optimisation.

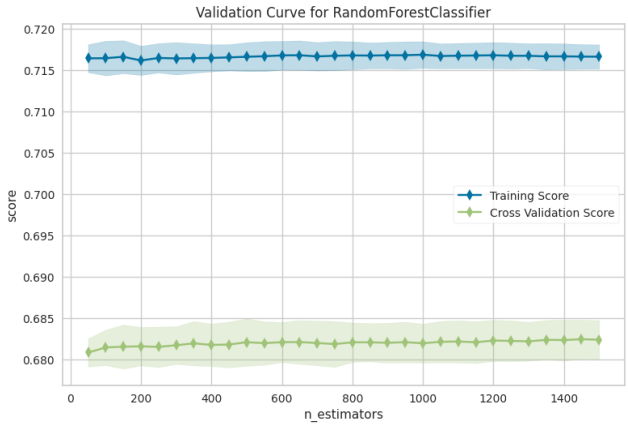

Figure 9: Courbe de validation pour RandomForestClassifier

Pour remédier à ce problème, nous pouvons envisager de réduire la complexité du modèle en ajustant des hyperparamètres tels que *max_depth*, *min_samples_split* et *min_samples_leaf*, ou en utilisant une méthode de régularisation pour améliorer la généralisation.

3.3.3 GradientBoostingClassifier

Par la suite, nous avons entraîné un modèle de GradientBoostingClassifier. Le modèle a été évalué en utilisant la matrice de confusion 6 et le score F1 obtenu est également de **0.70**. C'est pour le moment le meilleur modèle.

Table 6: Matrice de confusion - GradientBoostingClassifier

Predicted labels	**1**	**2**	**3**
True labels			
1	10 000	980	1 140
2	870	1 490	1 190
3	980	840	2 520

Les courbes d'apprentissage (10) montrent que les performances du modèle sur les jeux d'entraînement et de validation sont proches, indiquant une bonne généralisation. Les courbes révèlent que les scores de validation augmentent régulièrement avec la taille de l'échantillon d'entraînement, suggérant que le modèle continue à apprendre efficacement à partir de données supplémentaires. Il serait intéressant d'essayer sur un plus grand jeu de données, ce modèle s'exécutant rapidement.

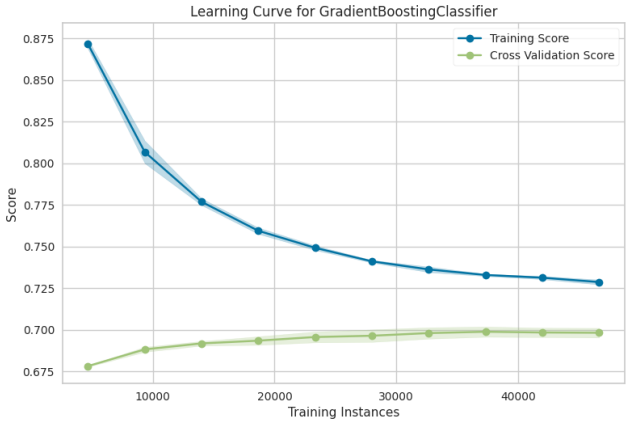

Figure 10: Courbe d'apprentissage pour GradientBoostingClassifier

3.3.4 Optimisation des modèles

Pour les deux modèles, nous avons effectué l'hyper-optimisation des paramètres avec le framework *Optuna* sur l'ensemble des paramètres des arbres.

3.3.5 Évaluation

Les résultats finaux montrent que les deux modèles, RandomForestClassifier et GradientBoosting-Classifier, obtiennent des scores F1 d'environ **0.70**. Sans surprise, sans optimisation, la méthode de bagging a tendance à se diriger vers de l'overfiting, à l'inverse du boosting.

3.4 K-Plus-Proches-Voisins (KNN)

3.4.1 Modélisation et résultats

Nous avons également entraîné sur un sous-ensemble équilibré du dataset un modèle des plus proches voisins. Les premiers résultats avec le modèle par défaut k=5 ont donné les moins bons résultats parmi tous les modèles : **F1 = 0,57** sur les données de test. L'écart entre les performances train et test sont par ailleurs donnés par la courbe d'apprentissage en figure (11)).

Nous avons de nouveau utilisé Optuna pour optimiser la valeur du paramètre k. Notons que plus de 8 heures ont été nécessaires pour effectuer seulement 10 tours d'optimisation. L'optimisation a en effet permis de réduire l'écart entre les performances train et test avec **F1_train = 0.65** et **F1_test = 0,62** (cf matrice de confusion 7). Nous notons qu'elle n'est cependant pas suffisante, et qu'il est possible d'améliorer le modèle, ce que n'avons pas pu achever cela par manque de ressources.

Table 7: Matrice de confusion - KNN (k=20)

Predicted labels	**1**	**2**	**3**
True labels			
1	25 245	5565	4180
2	3423	20 941	10 626
3	3677	13 198	18 116

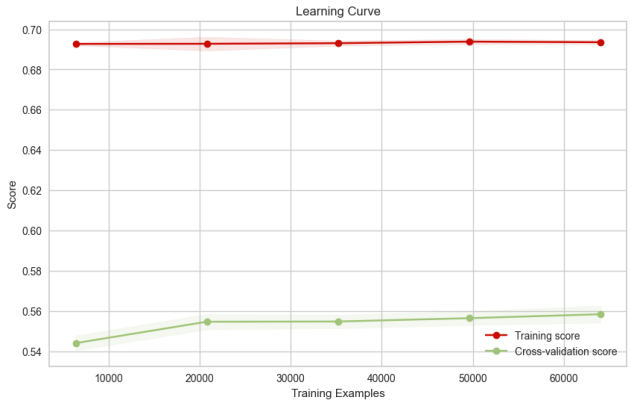

Figure 11: Courbe d'apprentissage pour le modèle KNN (k=5)

De manière assez évidente, l'algorithme KNN est celui qui nous a posé le plus problème en terme de temps de calcul et d'optimisation, conséquence directe de la taille de notre dataset (qui a pourtant été réduit au préalable). Il est important de souligner le fait que, même après avoir optimisé ce modèle et dépendamment de l'hyper-paramètre k sélectionné, il est extrêmement coûteux en temps de générer des prédictions sur un sous-ensemble des données.

4 CONCLUSION

Les performances comparées des différents groupes de modèles sont donnés par la table suivante (8). Notons que cette comparaison directe peut être remise en question, les sous-ensembles de données n'étant pas identiques d'un modèle à l'autre.

Table 8: Comparaison des scores F1 par groupe de modèle

Groupe de modèle	**Meilleur score F1**
Régression logistique	0,65
SVM	0.70
Apprentissage par ensemble	0,70
KNN	0,62

Nous trouvons important de souligner les temps d'entraînement et d'optimisation très longs de certains modèles, notamment le KNN et les SVM. Une critique peut surtout être formulée envers les KNN et la pertinence de leur utilisation sur des grands jeux de données. Le meilleur compromis en termes de ratio coût computationnel / performance nous semble être l'apprentissage par ensemble, en particulier la méthode du Gradient Boosting, qui obtient également les meilleurs résultats.

Il nous semble par ailleurs intéressant de mentionner le fait que les modèles entraînés ont été fortement influencés par la variable *sex* dans la prédiction (propre aux habitudes de tabagisme en Corée), et que le modèle nous semble ainsi difficilement généralisable à des jeux de données d'autres pays.

A DESCRIPTION DES VARIABLES EXPLICATIVES

- **Variables quantitatives**
  - *age:* âge (arrondi à 5 ans près)
  - *height:* taille (arrondie à 5 cm près)
  - *weight:* poids (kg)
  - *waisline:* tour de taille (cm)
  - *sight_left:* vue oeil gauche (métrique inconnue)
  - *sight_right:* vue oeil droit (métrique inconnue)
  - *SBP:* pression artérielle systolique (mmHg)
  - *DBP:* pression artérielle diastolique (mmHg)
  - *BLDS:* glycémie à jeun (mg/dL)
  - *tot_chole:* cholesterol total (mg/dL)
  - *HDL_chole:* taux de cholesterol HDL (”bon” cholesterol) (mg/dL)
  - *LDL_chole:* taux de cholesterol LDL (”mauvais” cholesterol) (mg/dL)
  - *triglyceride:* taux de triglyceride (mg/dL)
  - *hemoglobin:* hemoglobine (g/dL)
  - *serum_creatinine:* creatinine sanguine (mg/dL)
  - *SGOT_AST:* SGOT (transaminase glutamique-oxaloacétique sérique) AST (Aspartate transaminase)(IU/L)
  - *SGOT_ALT:* ALT (alanine-aminotransférase)(IU/L)
  - *gamma_GTP:* y-glutamyl transpeptidase (IU/L)
- **Variables catégorielles**
  - *sex:* male, female
  - *hear_left:* ouïe gauche (1 : normale, 2 : anormale)
  - *hear_right:* ouïe droite (1 : normale, 2 : anormale)
  - *urine_protein:* protéines dans l’urine (de 1 à 6)

B SUIVI DE LOI NORMALE DE CERTAINES VARIABLES EXPLICATIVES DU DATASET

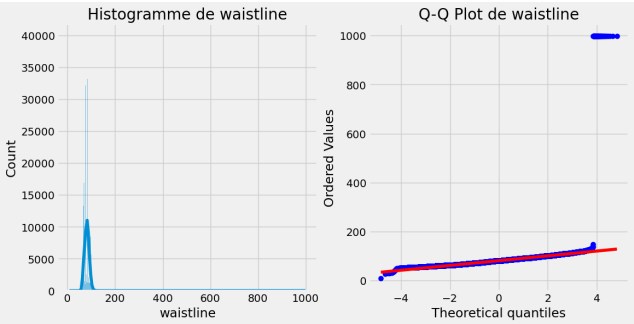

Figure 12: Suivi d’une loi normale de la variable *waistline* du dataset