PEUT-ON PRÉDIRE SI LES REVENUS D'UN INDIVIDU EXCÈDENT 50K DOLLARS PAR AN ÉTANT DONNÉ SON PROFIL?

EL RAFEI Jana - HASSAN Yousra - MA Haiyang
Rapport - Printemps 2024

RÉSUMÉ

L'objectif du projet de cette UV AI28 - Machine Learning, enseignée à l'Université de Technologie de Compiègne (UTC), est de mettre en pratique les méthodes d'apprentissage supervisé abordées en cours, en les appliquant à un jeu de données réel. Le jeu de données sélectionné, intitulé Adult est un ensemble de données tiré par Barry Becker en 1994 de la base de données Census. L'étude vise à déterminer les indicateurs permettant de prédire les revenus d'un individu en fonction de son profil.

1 INTRODUCTION

Ce projet de prédiction est destiné à analyser et interpréter le profil d'une multitude d'individus. L'objectif est de mettre en évidence les facteurs permettant de prédire si les revenus d'un individu sont supérieurs à 50K/an. Pour ce faire, nous entreprendrons dans un premier temps une analyse exploratoire des données (EDA) pour pouvoir les comprendre et avoir une idée globale de l'échantillon étudié. Nous utiliserons notamment différentes solutions de visualisations. Nous continuerons en présentant les différentes méthodes de traitement de données réalisées qui nous donnent accès à l'information utile en se débarrassant du bruit. Finalement, en tirant parti des algorithmes d'apprentissage automatique (ML), nous construirons des modèles prédictifs de classification binaire permettant d'estimer les revenus d'un individu en fonction de son profil.

2 STRUCTURE DU PROJET

Notre projet est structuré en plusieurs dossiers qui revêtent des responsabilités différentes:

- **data**: contient les modules permettant les actions d'extraction des données depuis le dépôt
- **images**: contient des sauvegardes des graphiques en format .png
- **lib**: contient tous les modules à importer (fonctions, librairies python, ..)
- **notebooks**: contient le détail des exécutions et les résultats de nos algorithmes d'entraînement ainsi que le module data_exploration.ipyng de visualisation de données
- **src**: contient les fonctions qui permettent d'automatiser les actions dans les algorithmes

Pour lancer les exécutions, il faut exécuter le fichier **run_project.py** qui permet l'installation des librairies et l'importation des données. Ensuite, en fonction de la catégorie du modèle que nous souhaitons entraîner, nous pouvons exécuter le notebook correspondant. Chaque modèle est répertorié dans ce rapport et indique le notebook associé.

Dans la suite, nous détaillerons la démarche suivie pour aboutir à un algorithme performant capable de répondre à notre problème de classification.

3 EXPLORATION DES DONNÉES

Avant de s'atteler à la prédiction des revenus pour un individu X, il nous a semblé fondamental de mieux comprendre le jeu de données. L'analyse exploratoire est disponible dans le fichier **note-**

books/data_exploration.ipynb. Celle-ci est très complète et seuls les éléments essentiels à retenir pour la suite du rapport sont expliqués ci-dessous.

Ce dataset est une matrice (48 842 * 15). Les colonnes, exceptées la dernière, représentent les variables et composent le profil d'un individu; elles seront appelées par leur dénomination anglaise, 'features'. La dernière est la variable cible, appelée 'target'. Les noms des colonnes et leur signification sont consignées ci-dessous :

Variable	Type	Description
age	int64	Age
workclass	object	Catégorie socio-professionnelle
fnlgwt	int64	Poids d'instance, caractéristiques démographiques similaires - correspond à l'appartenance à un groupe ethnique précis
education	object	Type d'études réalisées
education-num	int64	Nombre d'années d'études réalisées
marital-status	object	Statut marital
occupation	object	Emploi actuel
relationship	object	Relations
race	object	Groupe ethnique (différent de fnlgwt qui entre dans le détail de la région. Ici, on ne souhaite savoir que si l'individu est un 'blanc', 'noir, 'amérindien', .. de manière très générale).
sex	object	Genre
capital-gain	int64	Gains en capital
capital-loss	int64	Pertes en capital
hours-per-week	int64	Nombre d'heures travaillées par semaine
native-country	object	Pays d'origine
income	object	Revenus. 'Income' est la variable cible, elle est catégorielle et est composée de deux valeurs '>50K' ou '<=50K'.

3.1 Présentation des données

3.1.1 Target

Nous commençons notre analyse par l'encodage de la variable cible. En effet, prédire qu'un individu perçoit plus ou moins que 50K par an revient à un problème de classification binaire. On choisit alors d'encoder cette colonne qui prendra une valeur booléenne. Ainsi, si un individu perçoit plus de 50K, la valeur codée est 1. La colonne, initialement de type object et appelée 'income', devient alors de type int64 et intitulée '>50K'.
Nous observons immédiatement que les classes sont déséquilibrées: seules 25% des observations concernent les revenus '>50K'.

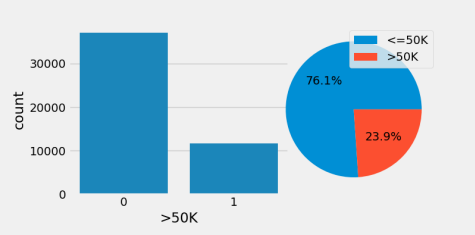

Figure 1: Distributions des classes dans le dataset

3.1.2 Features

Notre dataset est composé principalement de variables catégorielles, mais aussi de variables continues :

- Les variables catégorielles sont de type 'object', et composent 57% des variables (8/14).
- Les variables continues sont de type 'int64', et composent 43% des variables (6/14)

A cette étape, nous songions à faire un encodage one hot pour les variables catégorielles mais cela revient à multiplier le nombre de features qui passent de 14 à 108 (à cause de la multitude de catégories pour chaque variable).

Les graphiques ci-dessous représentent la distribution des variables catégorielles puis numériques, par classes pour avoir une idée plus concrète des données disponibles.

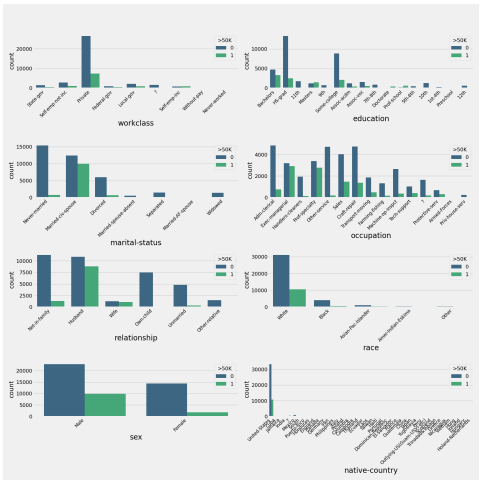

Figure 2: Distributions des variables catégorielles par classe

Nous observons que les données catégorielles présentent des pics significatifs qui témoignent de l'homogénéité du profil des individus. En effet, 90% des individus sont américains d'origine, 85% sont blancs, 67% sont des hommes, 70% d'une catégorie socio-professionnelle privée, .. Ces chiffres mettent en évidence le caractère biaisé des données issues d'une certaine catégorie de profil, ce qui pourra largement biaiser le modèle (erreurs de prédictions sur d'autres profils).

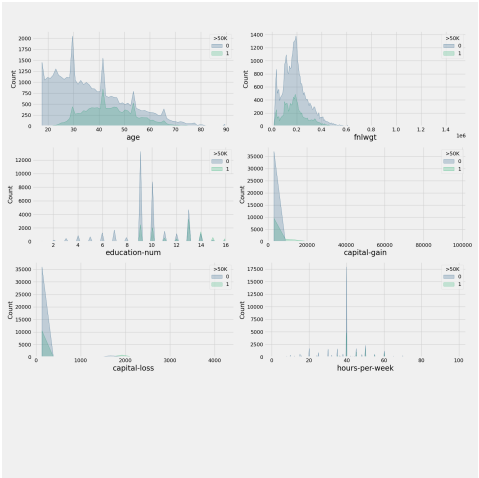

Figure 3: Distributions des variables catégorielles par classe

Les données numériques présentent également des données exploitables. Les données *Age* sont très irrégulières et s'étalent sur un spectre large même si la plupart des individus gagnant plus de 50K/an sont dans la tranche d'âge de 30 à 60 ans. Les données *education_num* fournissent des informations pertinentes sur le profil des individus et plusieurs pics sont mis en évidence à 9, 10 et 13 années d'études. Le reste des graphiques suit une loi plutôt normale. Enfin, la feature 'fnlwgt' semble avoir une très faible influence sur la cible, car les distributions des deux groupes sont très similaires.

3.2 Pré-traitement avant visualisation

A la réception du dataset, 48 lignes sont des duplicatas. On les supprimera lors du prétraitement mais également dans ce notebook pour mieux comprendre les données. De plus, trois variables (catégorielles) ont des valeurs manquantes avec des taux inférieurs à 2%. Nous décidons alors de les garder pour conserver l'intégrité des données.

3.3 Valeurs aberrantes

Les valeurs aberrantes des variables numériques sont représentées sur le graphique ci-dessous :

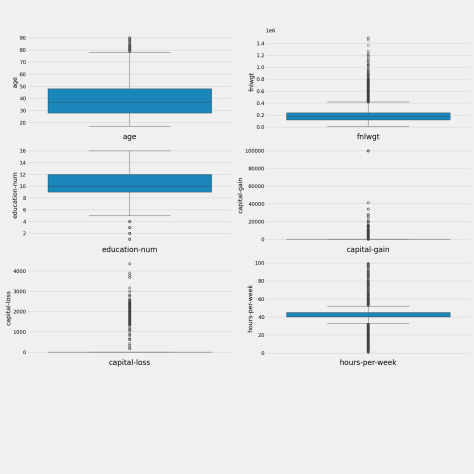

Figure 4: Distributions des variables catégorielles par classe

Nous avons choisi d'être prudents avec les valeurs aberrantes de ce dataset car tout profil différent du profil majoritaire évoqué plus tôt peut être considéré comme atypique.

Nous avons donc choisi de ne supprimer que les valeurs excédant 40000 pour la variable *capital-gain* et 4000 *capital_loss*.

Les outliers des variables catégorielles seront évoquées dans la partie prétraitement-regroupement des variables.

3.4 Corrélations

Les données sont très peu corrélées entre elles, comme en témoignent les deux graphiques:

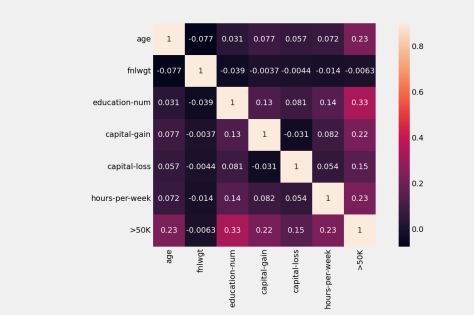

Figure 5: Matrice de corrélation entre les variables catégorielles

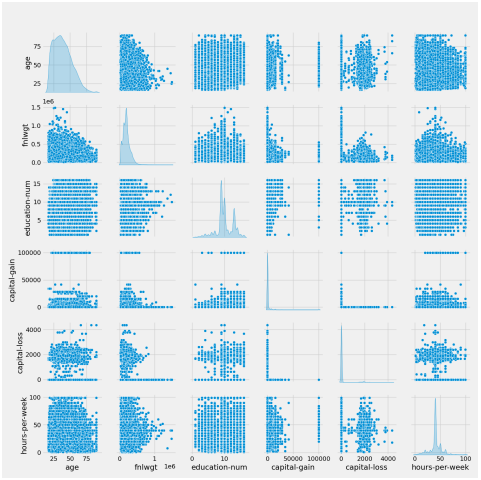

Figure 6: Distributions de la corrélation entre les variables numériques

Finalement, la visualisation des différents aspects de ces données nous a permis d'établir une stratégie de pré-traitement développée dans la partie suivante.

4 Traitement des données

Après avoir visualisé les données, nous connaissons à présent l'information utile de ce dataset. Nous procédons donc séquentiellement pour éliminer le bruit avant de passer à la prédiction. Tous les modèles développés possèdent cette étape dans leur notebook. Une fonction *preprocess* issue du fichier **src/data_preprocessing.py** est appelée, et permet le nettoyage des données. Le préprocessing continue ensuite avec l'imputation des valeurs manquantes et l'encodage des variables.

4.1 Nettoyage des données

La première phase de prétraitement est composée de plusieurs étapes distinctes :

- **Encodage de la variable cible:** Premièrement, on s'attelle à encoder la target pour qu'elles puissent prendre des valeurs booléennes. Cela simplifiera grandement la prédiction et l'analyse des résultats par la suite. En effet, la variable était initialement codée avec des chaines de caractères et était inscrite dans le dataset sous plusieurs formes pour une même classe. Il était donc impératif de normaliser la colonne afin qu'elle prenne ds valeurs binaires.
- **Suppression de variables inutiles:** Nous avons fait le choix de supprimer la colonne *education* car elle était équivalente à la colonne *education-num*. Ainsi, *education-num* représentait la colonne supprimée encodée en ordinal. Elle n'était donc pas nécessaire. Nous avons également souhaité supprimer la variable *fnlwgt* car elle n'apportait pas d'informations pertinentes. Cependant, après sa suppression, nous nous sommes rendus compte qu'elle engendrait +6000 duplicatas. De plus, nous avons testé de la supprimer et d'entraîner les modèles de régression logistique et d'arbres de décision (voir : `sans_avec_fnwlgt.ipynb` Cette suppression nous a valu un modèle parfait, avec une précision, un rappel et une score f1 de 1. Nous en avons conclu que cette variable apportait une information importante qui permet de distinguer des profils entre eux. Nous allons donc la garder même si l'information encodée est un nombre qui n'est pas exploitable (en effet, il représente l'encodage d'un groupe ethnique, sans qu'on puisse savoir lequel).
- **Suppression des valeurs aberrantes:** Comme nous l'avons vu dans les diagrammes en boîtes des variables numériques, les variables *capital gain* et *capital-loss* ont des seuils (respectivement 40000 et 4000) à partir desquels les valeurs sont considérées comme aberrantes. Ces observations ont été supprimées.
- **Regroupement des catégories:** La majorité des variables sont catégorielles et comportent de nombreuses catégories. Cependant, comme indiqué dans l'analyse exploratoire des données (AED), bien que ces catégories existent, peu sont réellement utilisées dans le dataset. Il a donc été nécessaire de regrouper certaines catégories pour mieux comprendre le jeu de données. Par exemple, la variable *relationship* comprend des catégories telles que 'Husband' et 'Wife', qui peuvent être regroupées sous une nouvelle catégorie appelée 'Married'. Plusieurs regroupements similaires ont été effectués. De même, nous avons consolidé sous la catégorie 'Other' tous les pays d'origine apparaissant dans moins de 200 observations. En effet, l'AED a révélé que 89,7% des individus sont originaires des États-Unis, mais que le dataset répertorie un total de 42 pays. Il est donc nécessaire de réduire le nombre de valeurs catégorielles inutiles ou peu représentatives pour simplifier l'encodage ultérieur. Sans ce regroupement, l'encodage one-hot aurait fait passer le nombre de features de 14 à 108, alors qu'avec le regroupement, on obtient 47 features. Cette consolidation a permis d'améliorer significativement les performances. Enfin, les valeurs '?' présentes dans plusieurs colonnes du dataset ont été renommées en 'Not referenced'.
- **Suppression des duplicatas:** L'étape d'après a été d'éliminer les duplications, qu'elles soient des erreurs de saisies où les conséquence des regroupements. Environ 300 duplicatas ont été supprimés.

Cette phase de nettoyage de données s'est avérée nécessaire pour pouvoir avoir des résultats exploitables. Les nouvelles données catégorielles sont présentées dans l'Appendix.

4.2 Imputation des valeurs manquantes

La suite du pré-traitement consiste en l'imputation des valeurs manquantes. Seules deux variables catégorielles voient certaines de leur valeurs manquer: *workclass* et *occupation*. Nous remplaçons alors les valeurs manquantes par la valeur la plus fréquente dans le dataset.

4.3 Normalisation des données - encodage des variables

La dernière phase de ce pré-traitement est l'encodage des variables. Les features sont regroupées par variables numériques et catégorielles. Les variables numériques sont encodées avec un *StandardScaler* tandis que les variables catégorielles sont encodées avec *One Hot Encoder*. Ce choix a été réalisé pour lever le problème de la priorisation de certaines variables lors d'un encodage ordinal pour certains modèles.

5 MODÈLES DE PRÉDICTION - APPRENTISSAGE SUPERVISÉ

Dans cette partie, nous allons explorer différents modèles que nous avons choisi d'implémenter dans notre projet.

Nous nous intéressons principalement à la métrique 'f1_score'. Ce choix repose sur le fait que nous ne privilégions pas une classe par rapport à une autre. En effet, prédire faussement qu'une classe est négative a le même impact que prédire qu'une classe est positive. L'objectif étant de maximiser la prédiction et le rappel, le problème revient donc à maximiser le score F1.

5.1 RÉGRESSION LOGISTIQUE

Le notebook `models_RegLog.ipynb` traite les différents types de régression logistique. Nous nous intéressons dans un premier temps à la régression logistique parce qu'elle est particulièrement adaptée à ce type de problème de classification binaire. Elle permet de modéliser la relation entre les variables indépendantes (telles que l'âge, le niveau d'éducation, etc.) et la variable dépendante (niveau de revenu) en estimant les coefficients qui maximisent la vraisemblance des observations. De plus, elle fournit une interprétation claire des coefficients, facilitant la compréhension de l'impact de chaque variable sur la probabilité de l'événement étudié.

Nous testons dans un premier temps la régression logistique simple afin d'identifier la performance du modèle et sa capacité à se généraliser.

Nous obtenons les résultats suivants :

	precision	recall	f1-score	support
0	0.88	0.93	0.90	7422
1	0.72	0.57	0.64	2287

Nous remarquons que le f1-score de la classe 1 est relativement bas par rapport à celui de la classe 0. Cela peut être dû à un jeu de données déséquilibré. Nous effectuons alors un sur-échantillonnage de la classe minoritaire (classe 1) en utilisant la technique SMOTE.

Nous obtenons les résultats suivants :

	precision	recall	f1-score	support
0	0.93	0.80	0.86	7422
1	0.56	0.82	0.66	2287

Nous remarquons une amélioration relative de f1-score pour la classe 1 mais une baisse pour la classe 0.

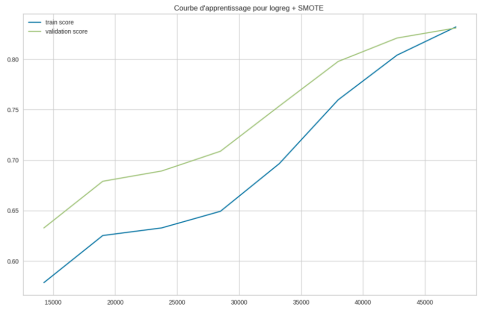

Figure 7: Courbe d'apprentissage du modèle de régression logistique

A partir de la courbe d'apprentissage (figure 7), nous pouvons voir que l'écart entre les deux courbes diminue avec l'augmentation de la taille de l'échantillon jusqu'à leur chevauchement. Cela nous permet d'espérer un ajustement du modèle si nous avions plus de données.
Nous continuons en ajoutant une régularisation du modèle en utilisant les pénalisations Lasso, Ridge et ElasticNet.

5.1.1 RÉGULARISATION : RIDGE (L2)

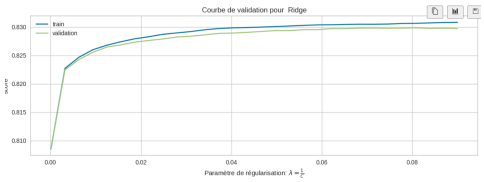

Figure 8: Courbe d'apprentissage du modèle de régression logistique Ridge

Les deux courbes montrent une tendance générale à la hausse au fur et à mesure que λ diminue (ou C augmente), ce qui indique qu'une régularisation plus faible (ou une absence de régularisation) améliore les scores jusqu'à un certain seuil.

Les courbes d'apprentissage (train) et de validation (validation) sont très proches l'une de l'autre, ce qui suggère que le modèle généralise bien. Cela signifie que le modèle ne souffre ni de sur-apprentissage ni de sous-apprentissage.

Le meilleur paramètre trouvé par la technique Grid Search est : C = 1.0 pour 100 itérations maximales.

5.1.2 RÉGULARISATION : LASSO (L1)

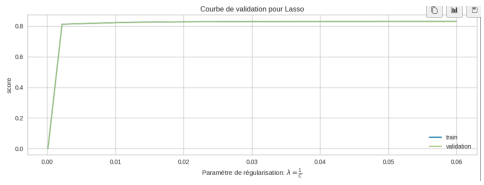

Figure 9: Courbe de validation du modèle de régression logistique Lasso

D'après la courbe de validation pour la régression Lasso, le score augmente très légèrement pour les deux datasets à partir d'environ 0.0025 de λ. Le modèle Lasso est ajusté (pas de sur-apprentissage ni de sous-apprentissage) à partir de cette valeur.

Le meilleur paramètre trouvé par la technique Grid Search est : C = 1.0 pour 100 itérations maximale.

5.1.3 RÉGULARISATION : ELASTICNET

Figure 10: Courbe de validation du modèle de régression logistique ElasticNet

La courbe de validation du ElasticNet est identique à celle de Lasso. Cela pourrait indiquer que le paramètre de mélange α dans ElasticNet est réglé de manière à donner un poids très élevé à la régularisation L1, ou que la composante L2 n'a que peu ou pas d'impact. En d'autres termes, l'ElasticNet fonctionne essentiellement comme un modèle Lasso.

Le meilleur paramètre trouvé par la technique Grid Search est : C = 1.0 pour 100 itérations maximales.

Nous obtenons les résultats suivants pour les trois modèles de régressions régularisées :

	precision	recall	f1-score	support
0	0.88	0.93	0.90	7422
1	0.72	0.57	0.64	2287

Les courbes d'apprentissage et de validation de ces trois modèles régularisée sont identiques à ceux de la régression logistique simple. Aucune amélioration de performance n'est obtenue malgré la régularisation du modèle. Nous essayerons alors par la suite d'autres modèles de prédiction.

5.2 ALGORITHME DES K-PLUS PROCHES VOISINS - KNN

Le notebook `models_KNN.ipynb` traite les algorithmes des K plus proches voisins.

Le modèle KNN est une autre approche couramment utilisée en apprentissage supervisé pour la classification binaire. Il se base sur la similarité des exemples d'apprentissage pour prédire la classe d'un nouvel individu.

Pour évaluer la pertinence du modèle KNN pour notre tâche de prédiction du revenu, nous avons commencé par une configuration de base avec $k = 5$ voisins. Voici les résultats obtenus :

	precision	recall	f1-score	support
0	0.82	0.89	0.86	7422
1	0.56	0.43	0.48	2287

Les performances initiales montrent une précision respectable pour la classe majoritaire mais une précision plus faible pour la classe minoritaire, ce qui peut indiquer une sensibilité à la distribution déséquilibrée des classes.

En explorant différentes valeurs de k et en appliquant des techniques comme le sur-échantillonnage de la classe minoritaire, nous avons cherché à améliorer les performances du modèle. Voici les résultats après ajustement :

	precision	recall	f1-score	support
0	0.85	0.82	0.83	7422
1	0.51	0.57	0.54	2287

Bien que les performances aient légèrement augmenté pour la classe minoritaire, le modèle KNN montre une tendance à sous-performer par rapport aux arbres de décision, en particulier en termes de précision et de recall pour la classe 1.

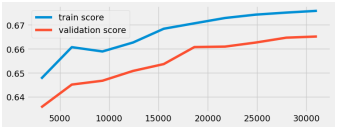

Figure 11: Courbe d'apprentissage du modèle KNN

La courbe d'apprentissage illustre le fait que le modèle a une performance faible, qui augmente peu à mesure de la taille de l'échantillon, indiquant un risque potentiel de sous-ajustement avec les données actuelles.

Pour optimiser les performances du modèle KNN, nous avons utilisé GridSearchCV pour ajuster les hyper-paramètres, en ciblant principalement k et la méthode de calcul des distances. Voici les paramètres optimisés :

```
param_grid = {
    'n_neighbors': list(range(1, 11, 2)) + list(range(11, 81, 10)),
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan', 'minkowski']
}
```

Les résultats avec les paramètres optimisés montrent une amélioration modeste mais significative :

	precision	recall	f1-score	support
0	0.88	0.93	0.90	7422
1	0.71	0.59	0.64	2287

Malgré les efforts d'optimisation, le modèle KNN ne parvient pas à atteindre une bonne performances sur f1-score de la classe 1.

5.3 Arbres de décision et forêts aléatoires

Le notebook `models_random_forests.ipynb` traite les algorithmes d'arbres de décision et des forêts aléatoires.

5.3.1 Arbres de décision

Les arbres de décision offrent un bon équilibre entre simplicité, interprétabilité et performance, ce qui est en fait un choix judicieux pour une tâche de classification binaire comme prédire si un individu gagne plus ou moins de 50K. Ils sont particulièrement utiles lorsque l'objectif est de comprendre les facteurs influençant les décisions et d'obtenir des résultats rapidement sans un prétraitement lourd des données. Ayant émis des réserves concernant les valeurs atypiques dans le dataset, il nous a semblé important d'entraîner un modèle peu sujet à l'impact du prétraitement.

Pour entraîner ce modèle, nous avons commencé par entraîner un modèle simple d'arbre de décision. Les résultats sont consignés ci-dessous :

	precision	recall	f1-score	support
0	0.88	0.88	0.88	7422
1	0.60	0.60	0.60	2287

Comme pour les modèles précédents, nous tentons alors d'augmenter les performances en sur-échantillonant la classe minoritaire.

Les résultats sont les suivants :

	precision	recall	f1-score	support
0	0.88	0.86	0.87	7422
1	0.57	0.63	0.60	2287

A première vue, les résultats semblent très légèrement se dégrader mais la courbe d'apprentissage nous montre que le modèle sur-échantilloné offre de meilleurs perspectives car il permet de diminuer la variance (malgré un biais plus élevé).

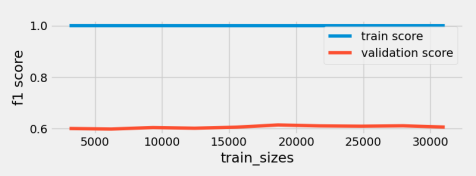

Figure 12: Courbe d'apprentissage de l'arbre de décision entraîné

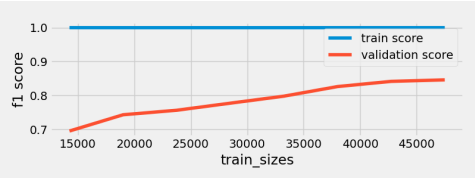

Figure 13: Courbe d'apprentissage de l'arbre de décision entraîné après suréchantillonage de la classe minoritaire

Nous avons décidé d'entraîner un modèle optimisé sur les données suréchantillonnées. Pour optimiser les hyperparamètres, nous avons utilisé GridSearchCV. Le processus de définition de l'intervalle des hyperparamètres s'est déroulé de manière itérative. Nous avons d'abord initialisé l'intervalle avec un large éventail de valeurs, puis l'avons affiné en fonction des résultats obtenus.

Ainsi par exemple, nous avons commencé avec param_grid = 'max_depth': [None, 10, 20, 30, 40], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'max_features': [None, 'sqrt', 'log2'] . Les meilleurs paramètres déterminés ont été : param_grid = 'max_depth': [16], 'min_samples_split': [2], 'min_samples_leaf': [6], 'max_features': [None] .

Ces paramètres ont fourni les résultats suivants, qui sont bien meilleurs que ceux d'avant mais toujours pas exceptionnels:

	precision	recall	f1-score	support
0	0.91	0.84	0.88	7422
1	0.59	0.74	0.66	2287

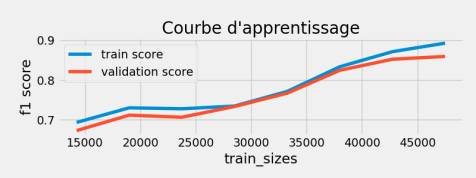

Figure 14: Courbe d'apprentissage de l'arbre de décision optimisé

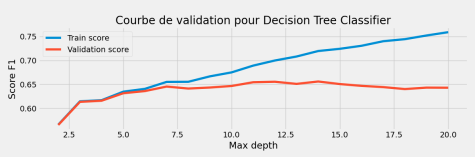

Figure 15: Courbe de validation de l'arbre de décision optimisé

Les courbes d'apprentissage de l'ensemble de test et de validation se confondent et se stabilisent autour de 37 000 observations, ce qui confirme notre hypothèse initiale selon laquelle les données sont très homogènes. En augmentant le nombre d'observations, l'écart entre les courbes se creuse, ce qui est indicatif d'un sur-apprentissage (overfitting).

De la même manière, la courbe de validation stagne aux alentours de 0.6 comme les autres modèles. Nous allons donc tester un autre modèle, les forêts aléatoires, pour tenter d'améliorer les performances.

5.3.2 Forêts aléatoires

Les arbres de décision ont tendance à sur-ajuster les données d'entraînement, surtout s'ils sont profonds (dans notre cas, max_depth = 16). Cela signifie qu'ils peuvent capturer le bruit dans les données et ne pas bien généraliser sur les données de test. En combinant plusieurs arbres de décision (souvent des centaines), chacun formé sur un sous-échantillon des données avec des caractéristiques sélectionnées aléatoirement, les forêts aléatoires réduisent le risque de sur-apprentissage. Les résultats sont moyennés sur tous les arbres, ce qui tend à lisser les prédictions et à améliorer la généralisation.

A l'instar de la méthodologie utilisée pour les arbres de décisions, nous avons entraîné un modèle sur les données sur-échantillonées puis nous avons optimisé les hyper-paramètres. Les résultats sont consignés ci-dessous:

Random Forest -données suréchantillonnées :

	precision	recall	f1-score	support
0	0.89	0.89	0.89	7422
1	0.65	0.66	0.66	2287

Random Forest -données suréchantillonnées et hyperparamètres optimisés:

	precision	recall	f1-score	support
0	0.90	0.89	0.89	7422
1	0.65	0.67	0.66	2287

La précision et le rappel ont augmenté mais légèrement malgré un entraînement sur plusieurs heures (plus de 5 heures). Nous continuons donc à chercher un modèle performant pouvant entraîner de meilleurs résultats.

5.4 Algorithmes d'ensemble

Dans cette section, nous examinons les algorithmes d'ensemble. Notre objectif est de combiner les prédictions de plusieurs modèles de base pour obtenir une performance supérieure à celle d'un seul modèle. En combinant plusieurs modèles, nous pouvons souvent améliorer la précision des

prédictions, car les erreurs de certains modèles peuvent être compensées par les prédictions correctes d'autres, ce qui conduit à une meilleure performance globale.

Dans le domaine du Machine Learning, plusieurs algorithmes d'ensemble sont couramment utilisés, notamment Adaboost, XGBClassifier et Stacking. Pour déterminer laquelle de ces trois techniques est la mieux adaptée à notre problématique, nous avons entrepris une comparaison de leurs performances.

Dans le notebook `models_comparison_ensembles.ipynb`, nous avons comparé les modèles mentionnés.

Nous avons initialisé les modèles de manière à garantir des conditions égales, en fixant *n_estimators* à 100 pour Adaboost et XGBClassifier, et en utilisant une validation croisée à 5 plis pour le modèle Stacking.

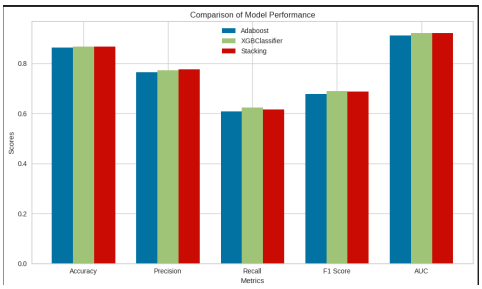

Figure 16: Courbe d'apprentissage du modèle de régression logistique

Les résultats de notre comparaison, illustrés par l'histogramme ci-dessus, montrent les performances des trois techniques selon différentes métriques : Accuracy, Precision, Recall, F1 Score et AUC. Nous observons que les trois modèles présentent des performances comparables en termes de précision et de rappel, avec des scores légèrement supérieurs pour XGBClassifier et Stacking en AUC, ce qui indique une meilleure capacité à distinguer entre les classes.

Après une analyse comparative rigoureuse des performances des modèles Adaboost, XGBClassifier et Stacking, notre choix s'est porté sur XGBClassifier pour la recherche d'hyperparamètres. Cette décision est motivée par plusieurs facteurs clés :

- **Performances supérieures :** XGBClassifier a démontré des scores élevés sur toutes les métriques de performance, en particulier en termes de Recall.
- **Efficacité et flexibilité :** XGBClassifier est reconnu pour sa capacité à gérer efficacement les données déséquilibrées et à s'adapter à différents types de problèmes grâce à ses nombreux paramètres réglables.

Cependant, il est important de noter que cette démarche n'est pas exhaustive. Nous pourrions également tirer parti des modèles Adaboost et Stacking pour obtenir un modèle plus performant en optimisant leurs hyperparamètres. Cette approche permettrait de choisir le modèle le plus adapté en fonction des meilleures performances obtenues.

XGBClassifier :
Dans le notebook `models_comparaison_ensembles.ipynb`, nous exploitons le modèle XGBClassifier.

Nous avons employé la technique d'optimisation hyperparamétrique Optuna pour déterminer les meilleurs paramètres du modèle. Notre fonction objectif visait à maximiser la moyenne des valeurs f1_micro obtenues par la validation croisée. Pour ce faire, nous avons exploré une plage étendue d'hyperparamètres au cours de 50 essais.

Il est important de noter que, dans cette partie du projet, nous avons utilisé un GPU Nvidia (via `nvidia-smi`) pour accélérer l'exécution.

XGBClassifier - hyperparamètres optimisés:

```
              precision    recall  f1-score   support
```

0	0.89	0.95	0.92	7422
1	0.78	0.61	0.69	2287

Nous remarquons ainsi une amélioration significative dans les f1-score des deux classes par rapport aux résultats précédentes.

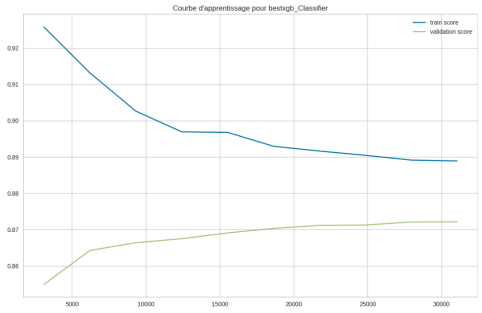

Figure 17: Courbe d'apprentissage du modèle de XGBClassifier avec les hyperparametres

La courbe d'entraînement commence à un score élevé (environ 0.92) mais diminue progressivement pour se stabiliser autour de 0.89. Cela indique que le modèle a initialement un faible biais puisqu'il s'ajuste bien aux données d'entraînement.
Cependant, la légère diminution du score d'entraînement indique que le modèle commence à capturer plus de complexité des données, mais reste globalement avec un biais faible.

La courbe de validation commence plus bas (autour de 0.86) et montre une légère augmentation pour se stabiliser autour de 0.87. La différence entre les scores d'entraînement et de validation indique une certaine variance. La différence n'est pas excessive, ce qui suggère que le modèle a une variance modérée. Le modèle ne sur-ajuste pas excessivement les données d'entraînement, mais il y a encore une petite différence à combler. Le taux d'erreur est l'inverse du score. Ainsi, pour la courbe de validation, un score de 0.87 signifie un taux d'erreur de 0.13 (ou 13 %). Le taux d'erreur est relativement stable et bas, ce qui indique une bonne performance générale du modèle, bien qu'il y ait encore une marge d'amélioration.

La courbe d'apprentissage montre que le modèle `bestxgb_Classifier` a un biais faible et une variance modérée. Le modèle s'ajuste bien aux données d'entraînement et généralise raisonnablement bien aux données de validation, avec un taux d'erreur qui se stabilise autour de 13 %. Des améliorations pourraient se concentrer sur la réduction de la variance pour encore mieux généraliser aux données de validation.

6 RÉSULTATS ET DISCUSSIONS

6.1 RÉSULTATS

Modèle	Precision	Recall	F1-score
Régression Logistique			
Classe 0	0.93	0.80	0.86
Classe 1	0.56	0.82	0.66
Régression Logistique avec pénalisation			
Classe 0	0.88	0.93	0.90
Classe 1	0.72	0.57	0.64
SVM			
Classe 0	0.87	0.94	0.91
Classe 1	0.75	0.55	0.64
KNN			
Classe 0	0.88	0.93	0.90
Classe 1	0.71	0.59	0.64
Arbres de décision			
Classe 0	0.88	0.88	0.88
Classe 1	0.60	0.60	0.60
Forêts aléatoires			
Classe 0	0.88	0.86	0.87
Classe 1	0.57	0.63	0.60
XGBClassifier			
Classe 0	0.89	0.95	0.92
Classe 1	0.78	0.62	0.69

6.2 DISCUSSIONS

Parmi les modèles étudiés, le XGBClassifier a montré les meilleures performances globales avec un F1-score de 0.69 pour la classe 1. Le modèle a démontré une capacité supérieure à distinguer les classes par rapport aux autres. Bien que nos résultats soient prometteurs, plusieurs avenues peuvent être explorées pour améliorer encore les performances :

1. Enrichissement des données :
   - Collecter des données supplémentaires ou intégrer des sources de données externes pour augmenter la diversité et la représentativité de l'échantillon, surtout de la classe minoritaire.
2. Traitement des données :
   - Explorer des techniques avancées de pré-traitement, telles que l'ingénierie des caractéristiques, la sélection automatique des variables et la réduction de dimensionnalité (par exemple, PCA).
3. Modèles avancés :
   - Tester des architectures de réseaux de neurones profonds (par exemple, les réseaux de neurones convolutifs et récurrents) pour capturer des relations complexes entre les variables.
   - Tester à optimiser les paramètres des modèles Adaboost et Stacking.
4. Approches d'ensemble :
   - Combiner plusieurs modèles de base à travers des méthodes d'ensemble plus robustes, comme le bagging et le stacking, pour améliorer la stabilité et la robustesse des prédictions.

7 CONCLUSION

Dans ce projet, nous avons exploré diverses méthodes d'apprentissage supervisé pour prédire si les revenus d'un individu excèdent 50K dollars par an en fonction de son profil. Nous avons testé

plusieurs modèles de classification, y compris la régression logistique, les machines à vecteurs de support (SVM, disponible dans le code), les K-plus proches voisins (KNN), les arbres de décision, les forêts aléatoires et les algorithmes d'ensemble tels que XGBClassifier.

En conclusion, ce projet a mis en évidence les défis et les opportunités dans la prédiction des revenus en utilisant des techniques d'apprentissage supervisé. Nos modèles offrent une base solide pour des améliorations futures et des applications potentielles dans des domaines similaires de la prédiction de données.

A APPENDIX

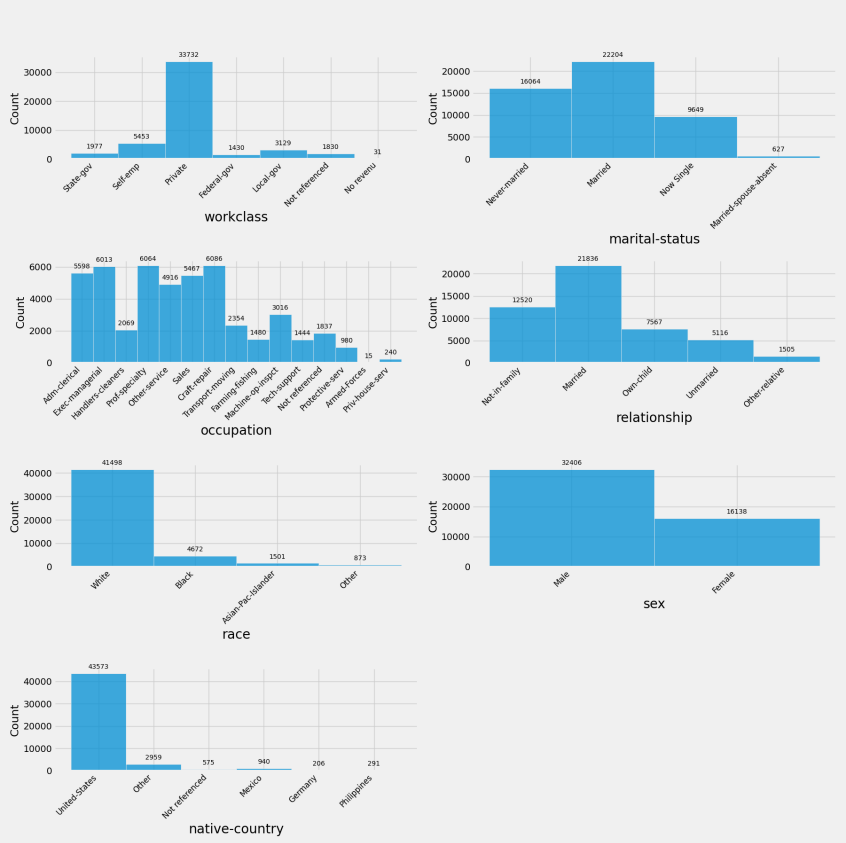

Figure 18: Distribution des variables catégorielles après nettoyage des données