Rapport de projet - Datamining

Adult Census Income

Gabrielle DIDELOT - Pauline MARTIN

I. Problème et enjeux

Pour ce projet du cours de DAC, nous avons choisi de travailler sur un dataset relativement classique issu de la base de données du bureau de recensement des États-Unis de 1994. Ce dataset recense les informations socio-démographiques d'un peu plus de 32 000 personnes aux Etats-Unis, et notamment leur salaire annuel. L'objectif de notre travail est de développer un modèle de classification permettant de prédire si un individu gagne plus ou moins de 50k$ par an. Un tel modèle s'inscrit dans un contexte de développement de plus en plus poussé de techniques de détection de fraude, et notamment de fraude à l'impôt, et représente donc un enjeu majeur.

II. Description du dataset

Le dataset, composé de 32 561 lignes, chacune représentant un habitant, et de 15 colonnes, représentant leurs caractéristiques socio-démographiques. Ces différentes caractéristiques, appelées variables ou features tout au long de ce rapport, peuvent être divisées en deux sous-ensembles : les variables quantitatives et les variables qualitatives

Variables qualitatives :

- **workclass :** le type d'entreprise dans lequel travaille l'individu
- **education :** le niveau d'études de l'individu
- **marital.status :** le statut marital de l'individu
- **occupation :** le type de métier exercé par l'individu
- **relationship :** la relation que l'individu entretient avec le responsable du foyer dans lequel il habite
- **race :** la race ethnique de l'individu
- **sex :** le sexe de l'individu
- **native.country :** le pays d'origine de l'individu
- **income :** le niveau de salaire de l'individu : plus ou moins de 50k$ par an. C'est cette variable que nous cherchons à prédire.

Variables quantitatives :

- **age :** l'âge de l'individu, représenté par un entier naturel.
- **fnlwgt :** le poids de l'individu dans l'échantillon afin que celui-ci soit représentatif de la population totale.
- **education.num :** le niveau d'études de l'individu, représenté cette fois par un entier naturel entre 1 et 16.

- **capital.gain :** les revenus issus de différents investissements, autres que le salaire, représentés par un entier naturel.
- **capital.loss :** les pertes issues de différents investissements, représentées par un entier naturel.

Parmi l'ensemble des personnes représentées, un peu plus de 75% d'entre eux gagnent moins de 50k$ par an. Le dataset est donc globalement déséquilibré.

III. Preprocessing des données

A. Sélection de variables par filtrage

Afin de créer un modèle pertinent, les données en entrée doivent être propres et pertinentes. En effet, des variables redondantes ou n'apportant aucune information utile favorisent l'over-fitting, autrement dit induisent une variance très importante dans le modèle, qui empêche alors sa généralisation à de nouvelles données.

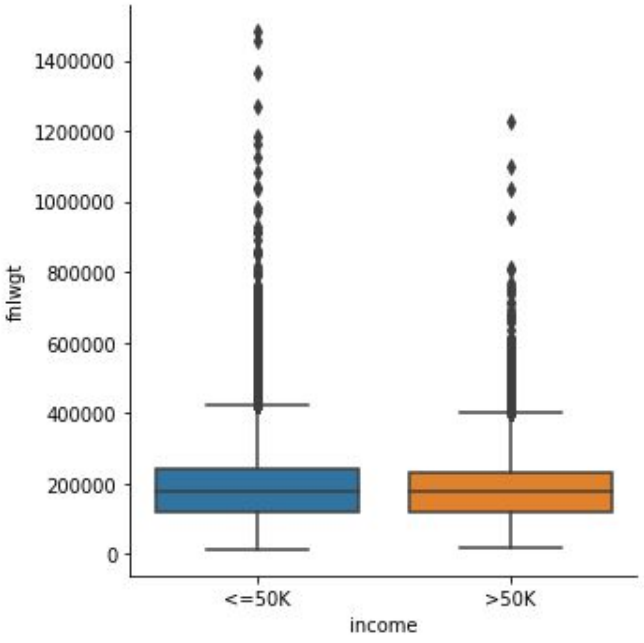
- **education et education.num**

On constate toute d'abord que l'éducation de l'individu, à savoir son niveau d'études, est représenté par deux variables différentes : une variable qualitative 'education' et une variable quantitative 'education.num'. Or, il s'avère que chacune des valeurs de la variable 'education.num' correspond à une unique valeur de 'education' et inversement. 'education.num' peut donc être considérée comme une traduction numérique du niveau d'études de l'individu. De plus, on peut vérifier qu'elle traduit également une relation d'ordre : plus la valeur de 'education.num' est élevée, plus l'individu a fait de longues études.

Ainsi, on peut supprimer la variable 'education' qui est totalement redondante par rapport à 'education.num'.

- **fnlwgt**

Comme évoqué précédemment, la variable 'fnlwgt' représente l'importance donnée à cette observation afin que le dataset dans son ensemble soit représentatif de la population totale. Ce n'est donc pas un attribut propre à l'individu mais plutôt de l'ordre du collectif.

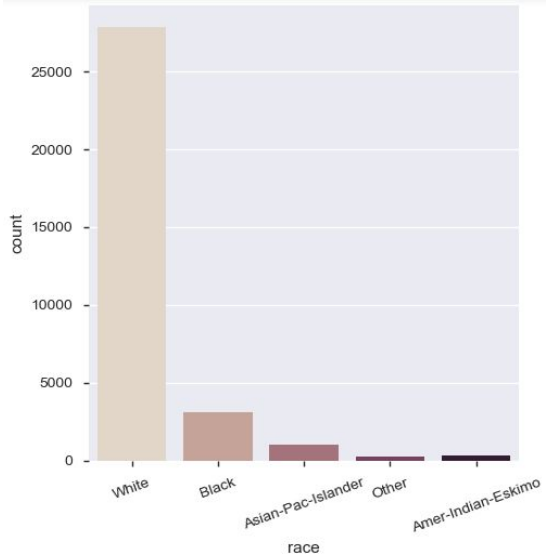


De plus, d'après le graphique ci-contre, on constate que les valeurs de 'fnlwgt' sont globalement distribuées de la même manière pour les individus gagnant plus de 50K$ que pour ceux gagnant moins.

La variable 'fnlwgt' n'est donc pas pertinente et est supprimée du modèle.

B. Traitement des variables qualitatives

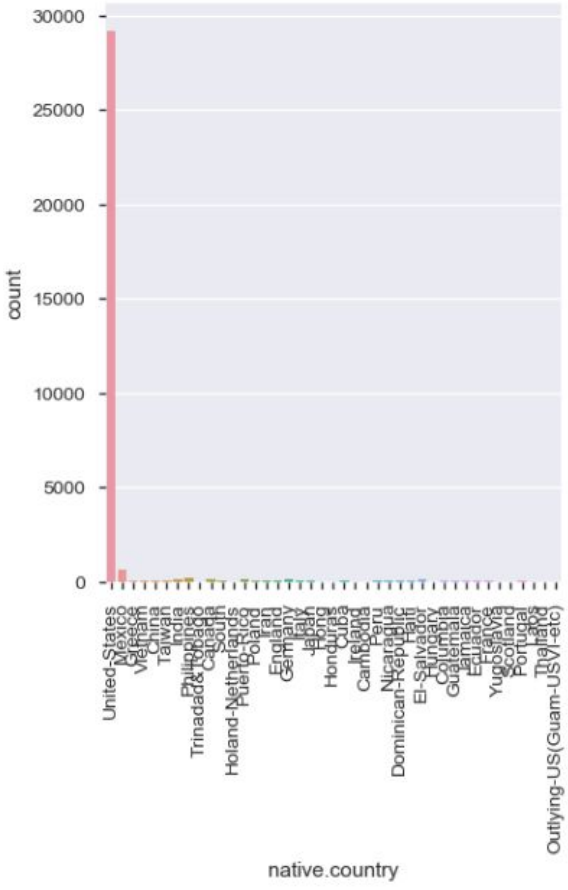
- **race**



En traçant le nombre de représentants de chaque ethnie, on remarque très clairement que les individus de “race” blanche sont bien plus nombreux parmi l’ensemble des observations. Les autres catégories sont beaucoup moins représentées.

On décide donc de regrouper toutes les catégories différentes de “White” sous le libellé “Other”. On se retrouve avec environ 15% d’individus classés comme “Other” et le reste, soit 85%, comme “White”.

- **native.country**

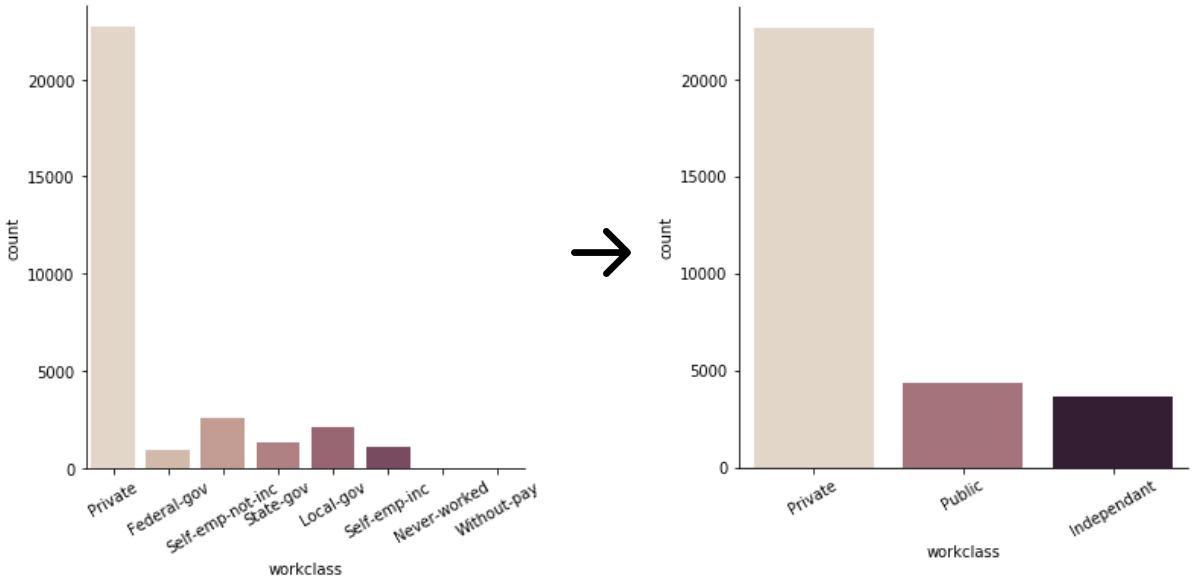


Le graphe ci-contre, bien que peu lisible dû au nombre de catégories différentes, montre clairement que le pays d’origine prédominant est ‘United-States’. Les autres pays apparaissent très rarement.

Tout comme pour la feature ‘race’, on décide donc de regrouper tous les pays autres que ‘United-States’ sous le libellé ‘Other’. On se retrouve ainsi avec un peu plus de 2 800 observations ‘Other’ et plus de 29 000 observations ‘United-States’.

- **workclass**

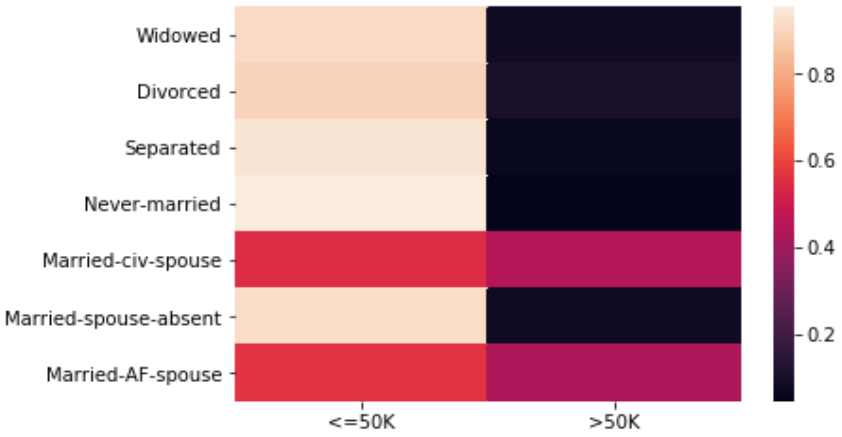
Le graphe ci-dessous présente la répartition des observations selon la variable ‘workclass’. Après analyse, il semble pertinent de regrouper les observations sous trois catégories différentes : les travailleurs du privé (‘Private’), ceux du public (‘Public’), et enfin les travailleurs indépendants (‘Independant’).
La catégorie ‘Private’ reste de loin la plus fréquente, avec près de 70% des observations.



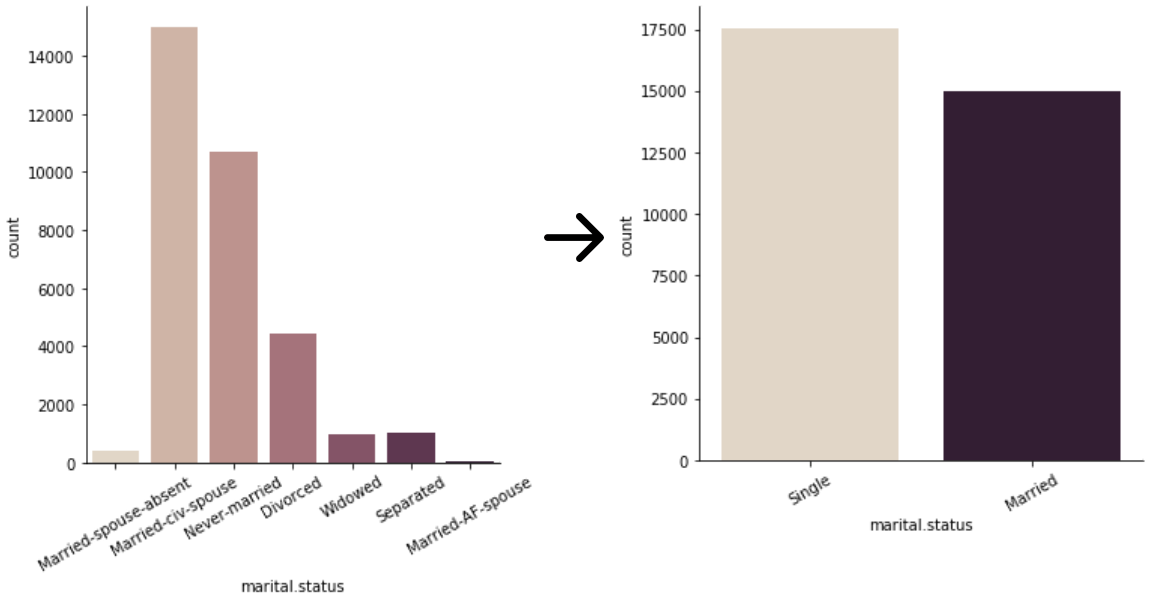
- **marital.status**

La variable ‘marital.status’ présente plusieurs catégories que l’on se propose de regrouper.

L’étude de corrélation entre ‘marital.status’ et ‘income’ (ci-contre) distingue deux groupes pertinents :
- les personnes seules qui gagnent pour la plupart moins de $50k
- les personnes mariées vivant en couple, dont la répartition des revenus est plus équilibrée



Les différentes catégories de ‘marital.status’ sont donc regroupées selon ces 2 groupes.

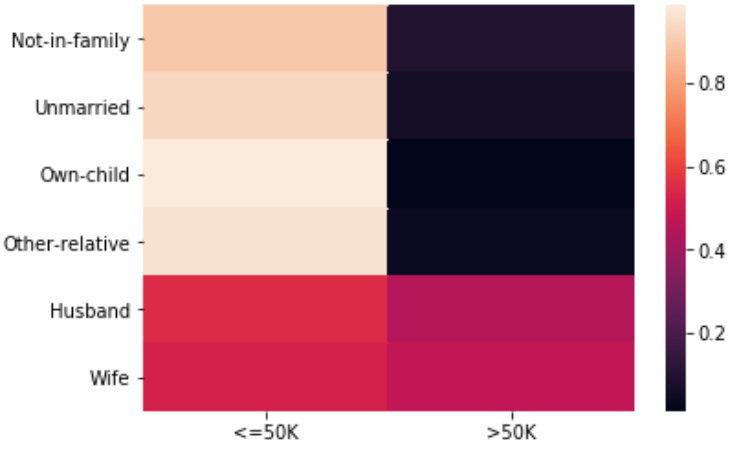


- **relationship**

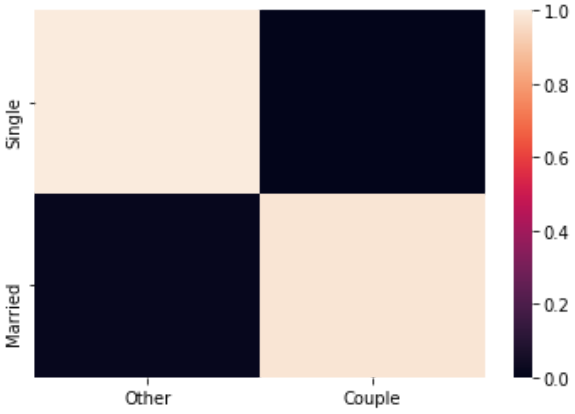
La variable ‘relationship’ correspond au lien entre une personne est le responsable du foyer où elle habite. Cette variable présente plusieurs catégories à regrouper.

L’étude de corrélation entre ‘relationship’ et ‘income’ (ci-contre) distingue deux groupes pertinents :
- les personnes vivant avec leur époux/se, dont la répartition des revenus est équilibrée
- les personnes vivant avec d’autres personnes, qui gagnent pour la plupart moins de $50k

Les différentes catégories de ‘relationship’ sont donc regroupées selon ces 2 groupes: ‘Couple’ et ‘Other’.



De plus, la corrélation ci-contre entre les groupes ‘Married’ et ‘Single’ de ‘marital.status’, et les groupes ‘Couple’ et ‘Other’ de ‘relationship’ montre que les deux informations sont redondantes.



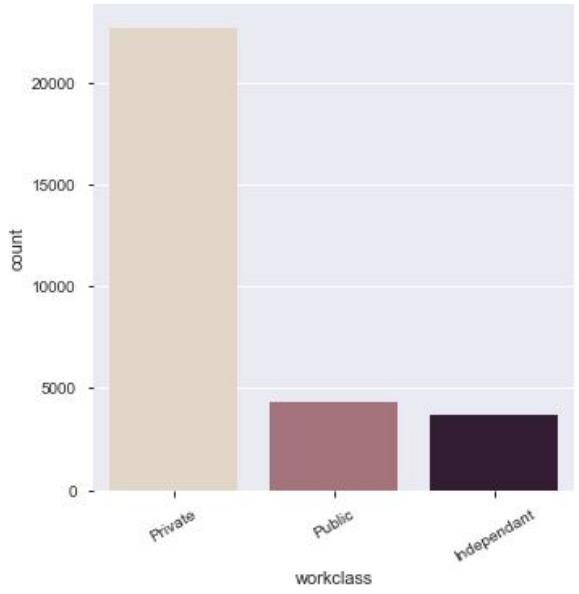
La variable ‘relationship’ est donc supprimée du modèle.

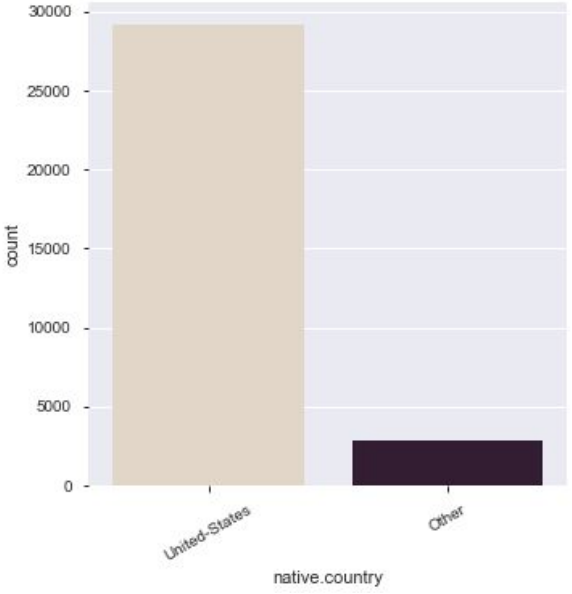
C. Valeurs manquantes

Les variables ‘workclass’, ‘native.country’ et ‘occupation’ ne sont pas renseignées pour tous les individus, et les caractéristiques de chacune de ces variables nous ont fait choisir différentes stratégies pour compléter les données manquantes.

- **workclass et native.country**

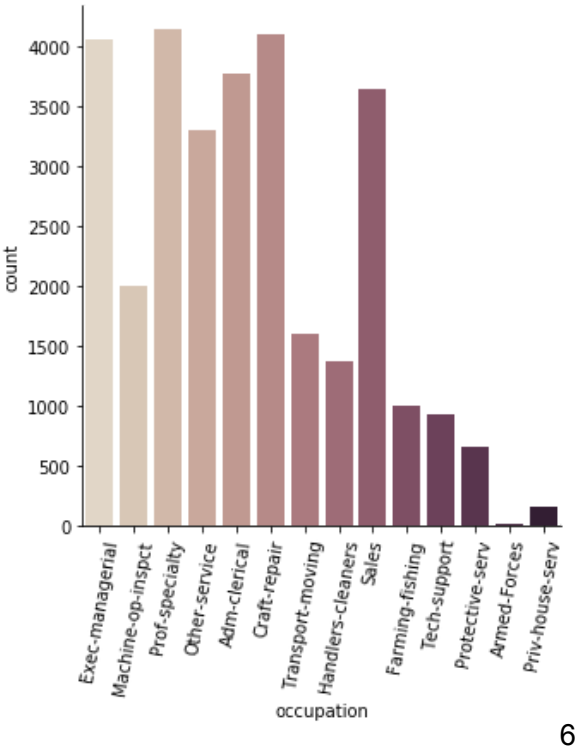
Les variables ‘workclass’ et ‘native.country’ sont très largement dominées par les valeurs ‘Private’ et ‘United-States’ respectivement (cf. graphes ci-dessous). Nous avons donc décidé de compléter toutes les valeurs manquantes par la valeur la plus fréquente dans chacun des cas. Nous utilisons pour cela le SimpleImputer du module impute de scikit-learn en choisissant la stratégie ‘most-frequent’.





- **occupation**

La variable ‘occupation’ est répartie de façon équilibrée entre ses différentes valeurs (cf. graphe ci-contre), il n’est donc pas judicieux de remplacer les valeurs manquantes par la plus fréquente. Cela induirait un déséquilibre non représentatif de la répartition réelle.

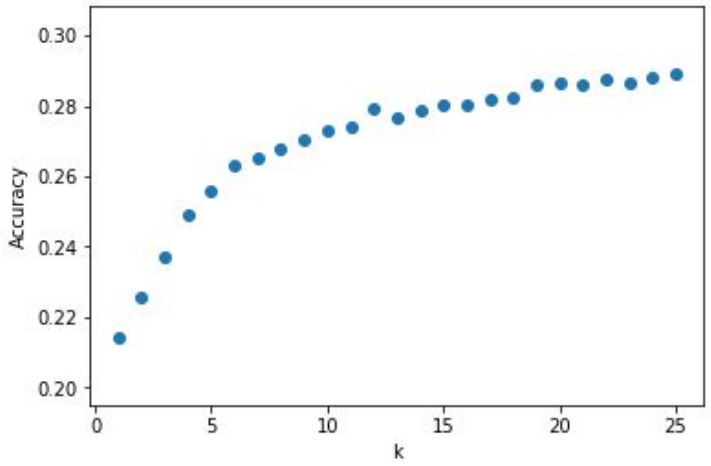


Nous prédisons donc les valeurs manquantes grâce à un modèle KNN, également appelé modèle des k plus proches voisins. Pour ce sous-problème de prédiction, le modèle est entraîné sur les lignes ne comportant pas de données manquantes, et est ensuite appliqué aux lignes non complètes pour prédire les données manquantes.

Pour le modèle KNN, nous appliquons le même preprocessing que le problème général.
De plus, la colonne ‘income’ est supprimée. En effet, les données de la variable ‘occupation’ que nous cherchons à prédire ici sont celles qui nous permettront ensuite de prédire la catégorie de revenus des individus (variable ‘income’). Il n’est donc pas rigoureux de conserver la variable ‘income’ pour la prédiction des données manquantes de la variable ‘occupation’.

Paramétrage du modèle

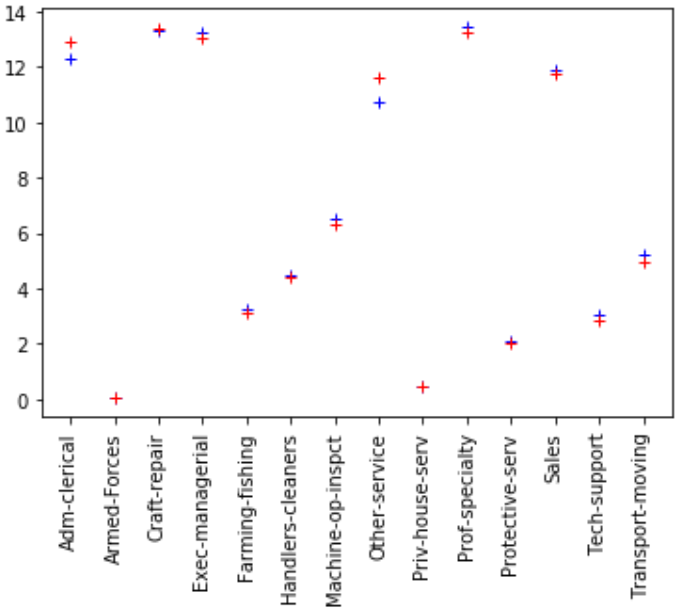
Le principal paramètre d’un modèle KNN est le nombre k de plus proches voisins auxquels se comparer. Pour choisir le k optimal, l’accuracy est mesurée pour différentes valeurs (cf. graphe ci-dessous). On choisit ainsi k=20, car au-delà l’accuracy est plafonnée par un palier.



Evaluation de la prédiction

L’accuracy ainsi obtenu est de 28%. A titre de comparaison, une répartition aléatoire des 14 différentes valeurs de ‘occupation’ donnerait un score de 7%.

Le modèle est appliqué aux lignes incomplètes pour prédire les données manquantes. Le graphe ci-contre compare le pourcentage de répartition des différentes valeurs de ‘occupation’ entre les données d’origine (en bleu) et les données complétées (en rouge).



Cette répartition statistique est quasi inchangée suite au remplissage des données, ce qui confirme la validité de la méthode utilisée.

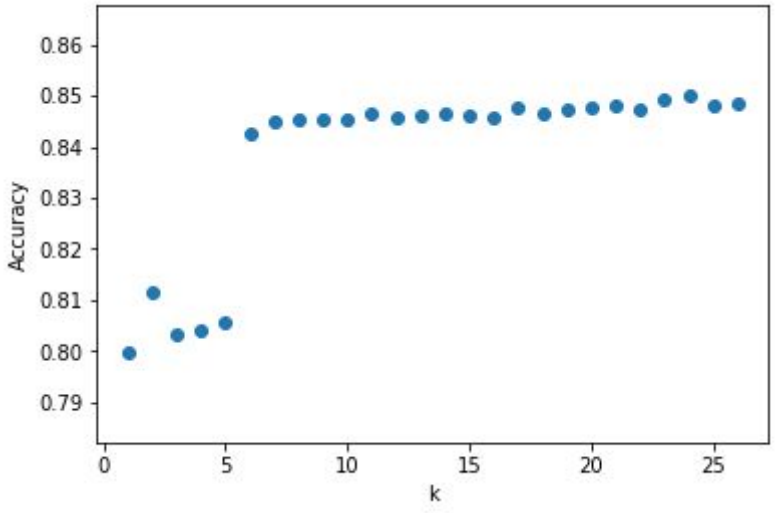
IV. Modèles de prédiction

Nous décidons d'évaluer l'ensemble de nos modèles en utilisant comme mesure de performance l'**accuracy**. En effet, c'est une mesure classique qui ne favorise aucune des deux classes : nous ne cherchons pas à détecter en priorité les individus gagnant plus ou moins de 50k$ par an.

A. Régression logistique

La régression logistique est un modèle classique de classification binaire. La variable à prédire ne peut donc prendre que deux valeurs : 0 ou 1. Basée sur la fonctione sigmoïde, la régression logistique permet de calculer la probabilité que la variable dépendante prenne la valeur 1, et ce pour chaque observation, en fonction des valeurs des features choisies. Par défaut, elle renvoie 1 si cette probabilité est supérieure à 0.5 et 0 dans le cas contraire.

Afin de s'assurer que nous entraînons bien notre modèle uniquement sur des features pertinentes, nous décidons d'appliquer une méthode de feature selection appelée SelectKBest : cette fonction permet de sélectionner les k features les plus corrélées à la variable à prédire (dans notre cas 'income'), et ce à l'aide du test du chi2.



On observe que le meilleur score est obtenu pour k = 26, soit le nombre total de features. Ainsi, il semble que toutes les features soient pertinentes
Ainsi, avec un modèle de **régression logistique,** on obtient une accuracy de **84.9%**.

Validation de la stratégie de remplissage des données manquantes
En testant le même modèle mais en ne prenant pas en compte les lignes où certaines données étaient manquantes, on obtient une accuracy maximale de **84.6%**. Celle-ci est inférieure à celle obtenue après imputation des données manquantes, ce qui valide donc la stratégie d'imputation.

B. Random Forest

L'algorithme Random Forest est une méthode de "bagging" qui permet de générer un ensemble aléatoire d'arbres de décisions de type CART. Cette méthode permet d'améliorer le niveau de prédiction qui serait obtenu avec un seul arbre. Pour obtenir des arbres tous différents, on sélectionne aléatoirement à chaque étape un nombre restreint de variables pour construire un noeud. Ainsi, la variable la plus discriminante retenue à chaque noeud est a priori différente pour chaque arbre.

Paramétrage du modèle

Les principaux paramètres à déterminer pour appliquer un algorithme RandomForest sont le nombre d'arbres, et le nombre maximum de features parmi lesquelles l'arbre a le choix à chaque noeud. Pour fixer ces paramètres, nous effectuons une recherche par GridSearch, un algorithme qui calcule le score du modèle pour différentes valeurs des paramètres, et choisit les plus pertinentes. En testant les paramètres suivants,

- nombre d'arbres : compris entre 100 et 340, testé avec un pas de 40
- nombre maximum de features : une fonction à appliquer au nombre total n de features parmi racine(n), log2(n), 0.1*n, et 0.5*n,

l'algorithme GridSearch renvoie 300 arbres et 0.5*n features au maximum au choix.
Avec ces paramètres, le modèle atteint une accuracy de 84.5%.

Sélection des variables les plus pertinentes par une approche 'wrappers'

Un trop grand nombre de variables en entrée du modèle créé des problèmes sur sur-apprentissage et dégrade le score.

Le modèle RandomForest permet de classer les variables selon le poids qu'elles ont dans la prédiction (cf. table ci-contre). Les 3 variables 'sex', 'race', et 'native.country' ont un poids particulièrement faible, nous étudions donc l'impact de chacune de ces variables sur le score.

category	feature_importance
age	0.209161
marital.status	0.200804
education.num	0.138725
capital.gain	0.126259
hours.per.week	0.109029
occupation	0.091041
capital.loss	0.041779
workclass	0.035396
sex	0.018242
race	0.016628
native.country	0.012938

Variables à supprimer	Accuracy	Impact
sex	84,3 %	⇓
race	84,5 %	=
native.country	84,6 %	⇑
race et native.country	84,4 %	⇓

Les variables ‘race’ et ‘native.country’ étant a priori corrélées, supprimer l’une des deux permet de réduire le sur-apprentissage et d’améliorer le score, mais supprimer les deux à la fois fait perdre de l’information. Nous supprimons donc seulement la variable ‘native.country’ pour améliorer le **modèle RandomForest**, et atteignons ainsi une accuracy de **84,9 %**.
La suppression de cette variable ne nécessite pas de modifier le modèle puisque l’algorithme GridSearch renvoie les mêmes paramètres optimaux suite à la suppression.

Validation de la stratégie de remplissage des données manquantes
Pour valider la pertinence de la stratégie d’imputation choisie lors du preprocessing, nous entraînons le même modèle sur le jeu de données sans prendre en compte les lignes comportant des données manquantes. Le score obtenu est alors 84,7 %, ce qui est moins performant et valide ainsi la pertinence de la stratégie d’imputation choisie.

C. XGBoost

XGBoost est un modèle de prédiction implémentant le gradient boosting. A ce jour, c’est un des algorithmes de machine learning les plus performants et est très utilisé dans de nombreux projets. Dans le cadre des arbres de décision, les techniques de boosting reposent sur la combinaison de nombreux arbres construits de manière séquentielle : chaque arbre est construit à partir des informations fournies par les autres arbres construits précédemment. Ainsi, le modèle final est constitué de nombreux ‘weak learners’ qui, considérés seuls, ont un niveau de performance très bas mais qui, combinés, permettent d’obtenir des résultats très concluants.

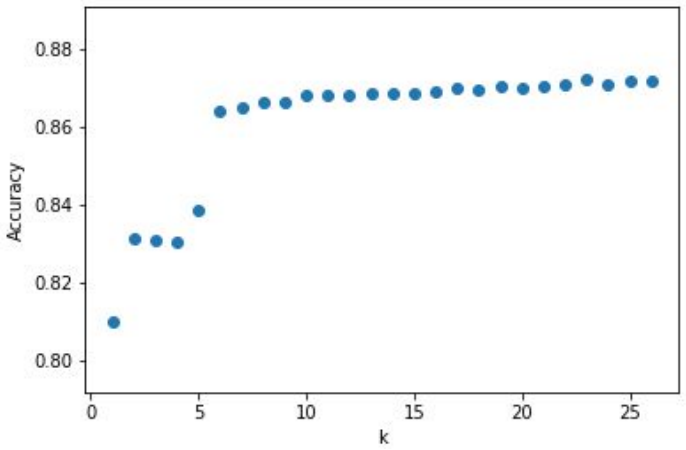
Paramétrage du modèle
Un modèle XGBoost repose sur trois paramètres importants :

- le nombre d’arbres t, qui permet de doser l’apprentissage et d’éviter l’over-fitting.
- le taux d’apprentissage λ, qui contrôle la vitesse à laquelle le modèle apprend. Il est fixé par défaut à 0.1 et nous décidons de ne pas le modifier.
- le nombre s de noeuds dans chaque arbre, qui contrôle la complexité de chaque arbre.

Afin de connaître les nombres d’arbres et de noeuds optimaux, nous effectuons un GridSearch qui va calculer le score pour différentes valeurs de ces deux paramètres et renvoyer le plus élevé. Ainsi, nous testons:

- un nombre d’arbres n entre 300 et 600, avec un pas de 50
- un nombre de noeuds s entre 1 et 3, avec un pas de 1

Le meilleur score est obtenu avec 500 arbres et 3 noeuds. Une fois ces paramètres fixés, on effectue à nouveau une analyse de la pertinence des différentes variables, en utilisant la fonction SelectKBest :



On observe que le meilleur score est obtenu en utilisant les k=23 features les plus corrélées à la variable 'income'. Les trois variables ignorées sont :

- native.country : il semble que le pays d'origine n'ait pas d'influence sur le salaire
- occupation_Armed-Forces : savoir qu'un individu travaille dans l'armée ne semble pas discriminant
- occupation_Sales : savoir qu'un individu travaille dans le secteur de la vente ne semble pas discriminant

Ainsi, avec un modèle de **XGBoost,** on obtient une accuracy de **87.2%**. C'est donc le modèle le plus efficace pour notre problème et par conséquent celui que nous sélectionnons.

Validation de la stratégie de remplissage des données manquantes

En testant le même modèle mais en ne prenant pas en compte les lignes où certaines données étaient manquantes, on obtient une accuracy maximale de **86.8%**. Celle-ci est inférieure à celle obtenue après imputation des données manquantes, ce qui valide donc la stratégie d'imputation.