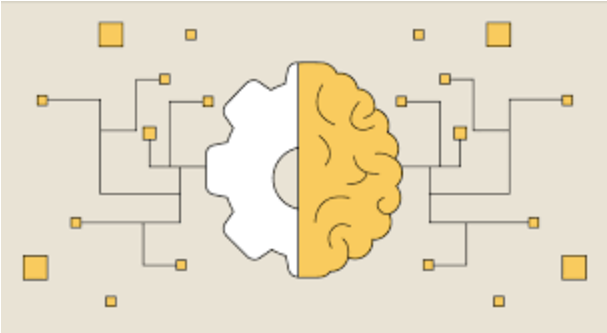
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Apprentissage supervisé

Compte rendu des séances de TP d'apprentissage supervisé

Hassouna Mohamed Amine – Laforge Mateo



Institut National des Sciences Appliquées de Toulouse

16 décembre 2025

Table des matières

Chapitre 1

Analyse et préparation du jeu de données ACSIncome

1.1 Description du jeu de données

Chaque observation du jeu de données *ACSIncome* correspond à un individu adulte résidant en Californie en 2018. La variable cible est le revenu annuel `PINCP`, transformé en une variable binaire indiquant si le revenu est supérieur à 50 000$.

1.2 Répartition des classes (déséquilibre).

Le jeu de données n'est pas parfaitement équilibré :

$$\text{False } (0) = 98112, \quad \text{True } (1) = 68203, \quad \text{Total} = 166315.$$

La classe majoritaire est donc **0 (revenu** $\leq 50K$**)**. Ce déséquilibre peut biaiser l'apprentissage (un modèle peut privilégier la classe majoritaire), et il justifie l'utilisation de **poids de classes** (*class weights*) ou de **poids d'exemples** (*sample weights*) lors de l'entraînement.

1.3 Analyse des attributs

Les attributs peuvent être regroupés en plusieurs catégories selon leur nature.

1.3.1 Attributs numériques

- **AGEP (âge)** : valeurs comprises entre 16 et 99 ans. La distribution est concentrée entre 25 et 65 ans, correspondant à la population active. Une corrélation positive modérée avec le revenu est observée jusqu'à un certain âge.
- **WKHP (heures travaillées par semaine)** : variable numérique discrète. La distribution présente un pic autour de 40 heures hebdomadaires, caractéristique du travail à temps plein. Cette variable est fortement corrélée positivement au revenu.

1.3.2 Attribut ordinal

- **SCHL (niveau d'éducation)** : variable ordinale allant de l'absence de scolarité jusqu'au doctorat. La majorité des individus possède un niveau lycée ou *bachelor*. Cette variable est l'une des plus explicatives du revenu : un niveau d'éducation élevé augmente significativement la probabilité d'un revenu supérieur à 50 000$.

1.3.3 Attributs catégoriels

- **COW (classe de travailleur)** : type d'emploi (privé, public, indépendant, etc.). Certaines catégories sont associées à des revenus plus élevés.

— **MAR (état matrimonial)** : la majorité des individus est mariée ou jamais mariée. Le statut de marié est positivement corrélé au revenu.
— **OCCP (occupation)** : variable à très forte cardinalité décrivant la profession. La distribution est fortement déséquilibrée, avec de nombreuses catégories rares.
— **POBP (lieu de naissance)** : pays ou État de naissance. Cette variable présente une grande diversité et une corrélation indirecte avec le revenu.
— **RELP (relation familiale)** : relation au sein du foyer. Les personnes de référence ou conjoints présentent en moyenne des revenus plus élevés.
— **SEX** et **RAC1P (race)** : ces variables présentent des corrélations avec le revenu, reflétant des inégalités socio-économiques structurelles. Elles sont considérées comme des attributs sensibles.

1.4 Préparation des données pour l'apprentissage automatique

L'objectif est de produire des données compatibles avec des modèles de la famille des arbres de décision (*Random Forest*, *AdaBoost*, *XGBoost*).

1.4.1 Encodage des variables

Les variables catégorielles (`COW`, `MAR`, `OCCP`, `POBP`, `RELP`, `RAC1P`) sont transformées par encodage binaire (*one-hot encoding*). Ce choix permet de représenter chaque modalité sans introduire d'ordre artificiel.

Les variables numériques (`AGEP`, `WKHP`) sont conservées sous forme continue. La variable `SCHL`, bien qu'ordinale, est également conservée sous forme numérique afin de préserver son ordre sémantique.

1.4.2 Sélection et traitement des attributs

Les variables à forte cardinalité, notamment `OCCP` et `POBP`, augmentent la dimension des données mais sont conservées afin de ne pas perdre d'information pertinente. La variable `PINCP` est utilisée uniquement pour la construction du label et n'est pas incluse parmi les variables explicatives.

1.4.3 Standardisation

Une normalisation des variables `AGEP` et `WKHP` est envisagée. Toutefois, pour les modèles à base d'arbres, cette étape n'est pas indispensable car ces modèles ne sont pas sensibles à l'échelle des variables. La standardisation est donc réalisée à titre expérimental et pour assurer une compatibilité avec d'autres familles de modèles.

1.5 Partition du jeu de données

Le jeu de données est mélangé puis séparé en deux ensembles :
— Ensemble d'entraînement : 75 %
— Ensemble de test : 25 %

La partition est effectuée aléatoirement avec une graine fixée (`random_state = 42`) afin de garantir la reproductibilité des résultats.

Chapitre 2

Expérimentation 1 : Comparaison de modèles par défaut

Partition des données

— Ensemble d'entraînement : **124 736 observations** × **10 variables**
— Ensemble de test : **41 579 observations** × **10 variables**
— Partition : 75% entraînement / 25% test (données mélangées)

Résultats – Hyperparamètres par défaut

Évaluation sur l'ensemble d'entraînement Les performances mesurées sur l'ensemble d'entraînement sont légèrement supérieures à celles observées sur l'ensemble de test, ce qui indique un sur-apprentissage limité pour les trois modèles.

TABLE 2.1 – Performances sur l'ensemble d'entraînement

Métrique	**Random Forest**	**AdaBoost**	**XGBoost**
Accuracy	≈ 0.83	≈ 0.81	≈ 0.84
Temps de calcul (s)	14.47	2.37	0.92

TABLE 2.2 – Performances sur l'ensemble de test

Métrique	**Random Forest**	**AdaBoost**	**XGBoost**
Accuracy	0.8155	0.8034	**0.8261**
Temps de calcul (s)	14.47	2.37	**0.92**

Évaluation sur l'ensemble de test

Matrices de confusion

Les matrices de confusion présentées ci-dessous permettent d'analyser la répartition des erreurs de classification pour chaque modèle.

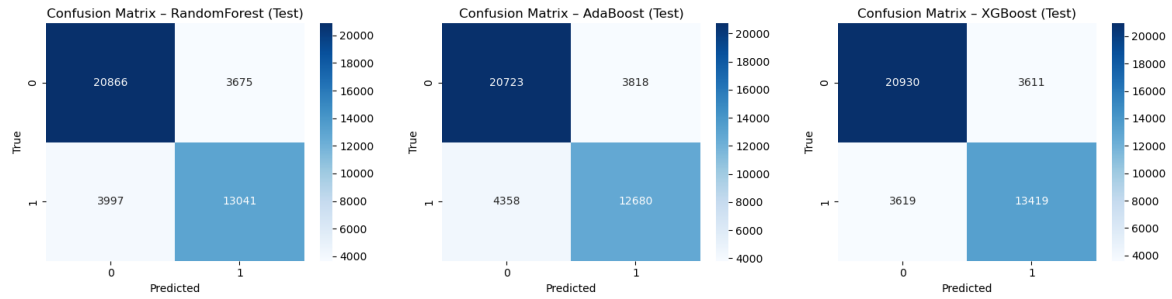


FIGURE 2.1 – Matrices de confusion sur l'ensemble de test (Random Forest, AdaBoost, XGBoost)

Comparaison globale des performances

La figure suivante synthétise les performances des trois modèles en termes d'accuracy sur l'ensemble de test.

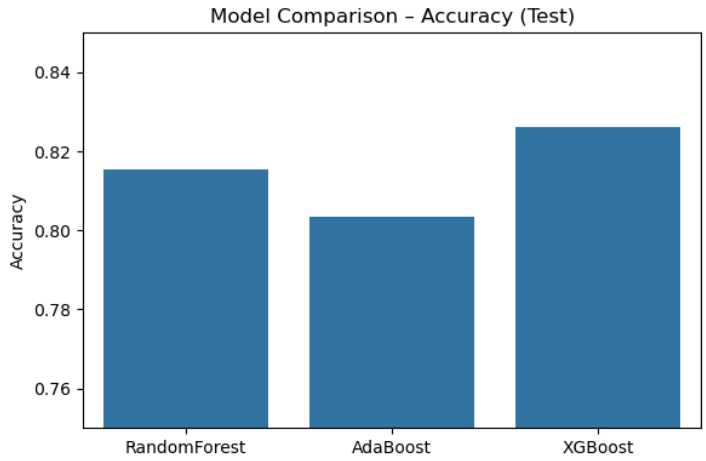


FIGURE 2.2 – Comparaison des modèles selon l'accuracy sur l'ensemble de test

Analyse et commentaires

Les résultats montrent que le modèle **XGBoost** obtient la meilleure accuracy sur l'ensemble de test (82.61%), tout en présentant le temps de calcul le plus faible. Le modèle **Random Forest** offre des performances comparables, mais avec un coût computationnel plus élevé. **AdaBoost** présente des performances légèrement inférieures, tout en restant compétitif grâce à un temps d'entraînement modéré.

Chapitre 3

Expérimentation 2 : Comparaison des modèles après optimisation

Jeu de données

Le jeu de données utilisé est **ACSIncome** (État de Californie, 2018). La tâche consiste à prédire si le revenu annuel d'un individu est supérieur à 50 000$ à partir de variables socio-démographiques et professionnelles.

Partition des données

- Ensemble d'entraînement : **124 736 lignes** × **10 colonnes**
- Ensemble de test : **41 579 lignes** × **10 colonnes**
- Partition : 75% entraînement / 25% test (données mélangées)

Hyperparamètres testés

- `n_estimators` $\in \{100, 400\}$ (2 valeurs)
- `max_depth` $\in \{\text{None}, 10, 20\}$ (3 valeurs)
- `min_samples_leaf` $\in \{1, 3, 5, 10\}$ (4 valeurs)

Validation croisée

- Nombre de plis : 3
- Nombre de combinaisons testées : $2 \times 4 \times 4 = 24$
- Nombre total d'entraînements : $24 \times 3 = 72$

Performances

- Accuracy entraînement : 0.9581
- Accuracy test : 0.8210
- Temps de calcul total : 1101.96 s

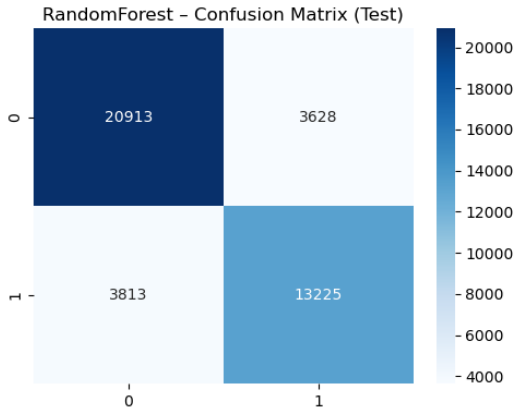


FIGURE 3.1 – Matrice de confusion Random Forest (test)

Analyse L'optimisation améliore légèrement les performances par rapport au modèle par défaut (0.8155 $\rightarrow$ 0.8210), mais au prix d'un coût computationnel élevé et d'un sur-apprentissage notable.

3.1 AdaBoost

Processus d'entraînement Le modèle AdaBoost est optimisé avec `GridSearchCV` afin d'ajuster les hyperparamètres ayant le plus d'impact sur le compromis biais/variance : le **nombre d'itérations** et le **taux d'apprentissage**.

Hyperparamètres testés et rôle

- `n_estimators` $\in \{50, 200\}$: nombre de weak learners (itérations). *Plus il est grand, plus le modèle peut capturer des relations complexes, mais le risque de sur-apprentissage et le temps de calcul augmentent.*
- `learning_rate` $\in \{0.5, 1.0, 2.0\}$: poids appliqué à chaque itération (vitesse d'apprentissage). *Une valeur faible rend l'apprentissage plus progressif (souvent plus stable), une valeur élevée peut améliorer vite mais rendre le modèle plus sensible au bruit.*

Hyperparamètres les plus utiles (à retenir) Dans AdaBoost, `n_estimators` et `learning_rate` sont les deux paramètres déterminants : ils contrôlent directement la **capacité du modèle** et la **stabilité de l'optimisation**. Les autres réglages ont un impact secondaire par rapport à ces deux leviers.

Validation croisée

- Nombre de plis : 3
- Nombre de candidats : $2 \times 3 = 6$
- Nombre total d'entraînements : $6 \times 3 = 18$

Meilleurs hyperparamètres

```
n_estimators = 200
learning_rate = 1.0
```

Performances

- Accuracy entraînement : 0.8100
- Accuracy test : 0.8092
- Temps de calcul total : 173.58 s

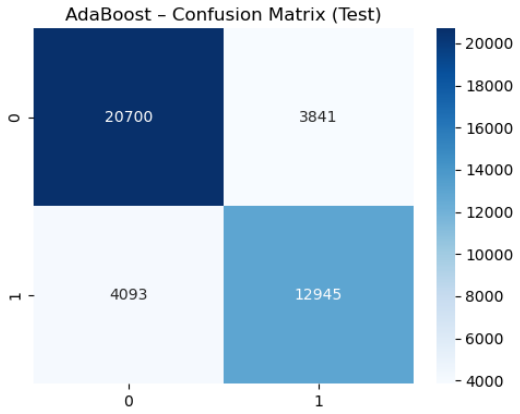


FIGURE 3.2 – Matrice de confusion AdaBoost (test)

Analyse Le tuning améliore légèrement les performances par rapport au modèle par défaut (0.8034 $\rightarrow$ 0.8092). Le couple (`n_estimators`=200, `learning_rate`=1.0) indique qu'un nombre d'itérations plus élevé apporte un gain, sans déstabiliser l'apprentissage.

3.2 XGBoost

Processus d'entraînement Le modèle XGBoost est optimisé via `GridSearchCV` en ajustant les hyperparamètres qui contrôlent principalement : la **complexité des arbres** et la **vitesse d'apprentissage**.

Hyperparamètres testés et rôle

- `max_depth` $\in \{3, 6\}$: profondeur maximale des arbres. *Plus elle est grande, plus le modèle est expressif, mais plus il risque de sur-apprendre.*
- `n_estimators` $\in \{100, 300\}$: nombre d'arbres boosting. *Plus il est grand, plus la performance peut augmenter, au prix d'un temps de calcul supérieur.*
- `learning_rate` $\in \{0.05, 0.1\}$: taux d'apprentissage (shrinkage). *Valeur faible = apprentissage plus progressif (souvent plus robuste), valeur plus élevée = convergence plus rapide.*

Hyperparamètres les plus utiles (à retenir) Dans XGBoost, `max_depth` et `learning_rate` sont généralement les paramètres les plus influents car ils contrôlent directement la **généralisation** (sur/sous-apprentissage). `n_estimators` ajuste ensuite la capacité globale du boosting.

Validation croisée

- Nombre de plis : 3
- Nombre de candidats : $2 \times 2 \times 2 = 8$
- Nombre total d'entraînements : $8 \times 3 = 24$

Meilleurs hyperparamètres

```
n_estimators = 300
max_depth = 6
learning_rate = 0.1
```

Performances

- Accuracy entraînement : 0.8440
- Accuracy test : 0.8261
- Temps de calcul total : 27.18 s

3.3 Comparaison globale :

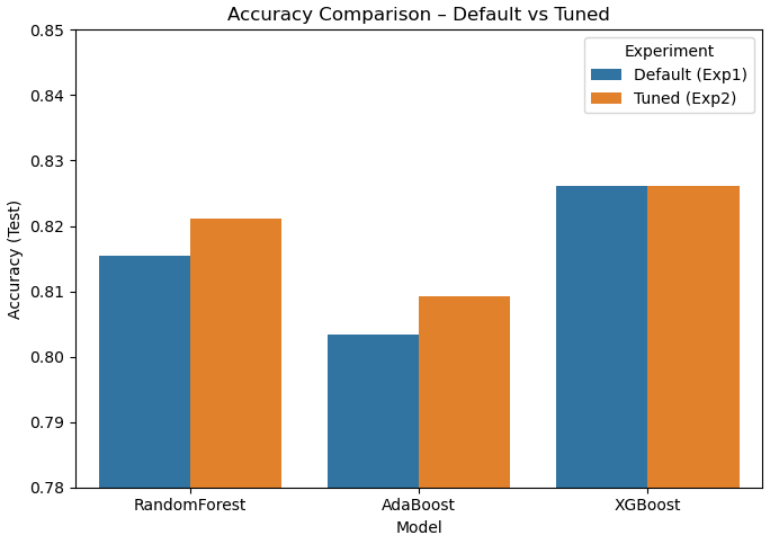


FIGURE 3.3 – Comparaison des accuracies sur l'ensemble de test entre les modèles par défaut (Expérimentation 1) et optimisés (Expérimentation 2)

Conclusion

La recherche d'hyperparamètres permet une amélioration modérée des performances pour Random Forest et AdaBoost, au prix d'un coût computationnel plus élevé. XGBoost conserve les meilleures performances globales et présente une grande robustesse, ce qui en fait le modèle le plus adapté à cette tâche de classification.

Chapitre 4

Expérimentation 3 : Comparaison des meilleurs modèles

Jeu de données

Le jeu de données utilisé est **ACSIncome** (État de Californie, année 2018). L'objectif est de prédire si le revenu annuel d'un individu dépasse 50 000$ à partir de variables socio-démographiques et professionnelles.

Partition des données

— Ensemble d'entraînement : **124 736 lignes** × **10 colonnes**
— Ensemble de test : **41 579 lignes** × **10 colonnes**
— Partition : 75% entraînement / 25% test (mélange aléatoire)

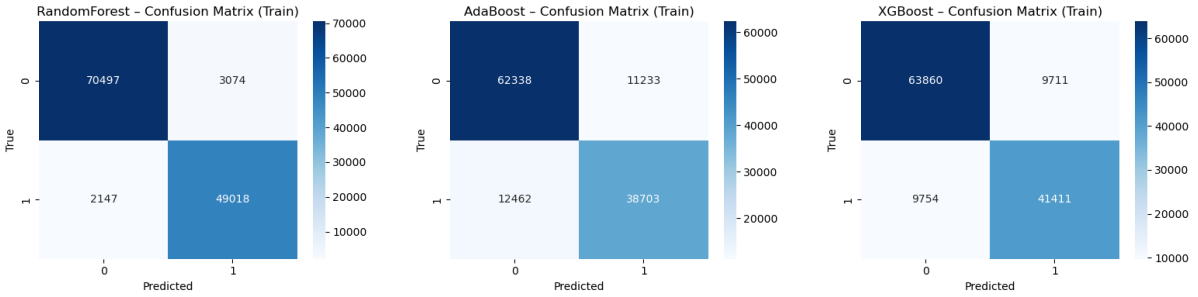
Objectif de l'expérimentation

Cette expérimentation vise à comparer les **meilleurs modèles obtenus à l'issue de l'Expérimentation 2** (Random Forest, AdaBoost et XGBoost), en évaluant leurs performances finales sur les ensembles d'entraînement et de test à l'aide de plusieurs métriques.

4.1 Résultats sur l'ensemble d'entraînement

TABLE 4.1 – Performances des meilleurs modèles sur l'ensemble d'entraînement

Métrique	**Random Forest**	**AdaBoost**	**XGBoost**
Accuracy	0.9581	0.8100	0.8440
Temps de calcul (s)	113.50	28.76	12.04



(a) Random Forest (b) AdaBoost (c) XGBoost

FIGURE 4.1 – Matrices de confusion sur l'ensemble d'entraînement

Analyse Random Forest présente une accuracy d'entraînement très élevée, traduisant un sur-apprentissage marqué. AdaBoost et XGBoost montrent des performances plus modérées et mieux régularisées.

4.2 Résultats sur l'ensemble de test

TABLE 4.2 – Performances des meilleurs modèles sur l'ensemble de test

Métrique	Random Forest	AdaBoost	XGBoost
Accuracy	0.8210	0.8092	**0.8261**
Précision	0.7847	0.7712	**0.7877**
Rappel	0.7762	0.7598	**0.7879**
F1-score	0.7804	0.7654	**0.7878**
Temps de calcul (s)	113.50	28.76	**12.04**

4.3 Comparaison globale des métriques

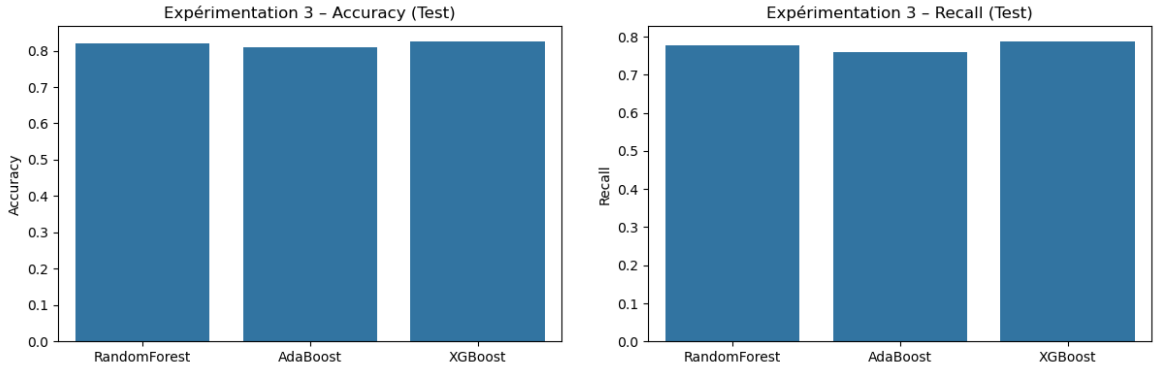


FIGURE 4.2 – Comparaison des métriques (accuracy , rappel) sur l'ensemble de test

Commentaires et analyse

Les résultats montrent que **XGBoost** offre le meilleur compromis global entre performance et coût computationnel. Il obtient les meilleures valeurs pour l'accuracy, la précision, le rappel sur l'ensemble de test, tout en étant le modèle le plus rapide à entraîner. Random Forest présente un sur-apprentissage important, tandis qu'AdaBoost, bien que plus stable, reste moins performant. Ces observations confirment que XGBoost est le modèle final le plus adapté pour cette tâche de classification.

Chapitre 5

Expérimentation 4 : inférence sur un autre jeu de données (optionnel)

Objectif

L'objectif de cette expérimentation est d'évaluer la **capacité de généralisation** des meilleurs modèles (optimisés sur la Californie lors de l'Expérimentation 2) lorsqu'ils sont appliqués **sans ré-entraînement** sur des jeux de données provenant d'autres États. Deux États sont considérés : **Colorado (CO)** et **Nevada (NE)**.

Méthodologie

Les modèles optimisés (*Random Forest*, *AdaBoost* et *XGBoost*) sont appliqués directement aux nouveaux jeux de données. On évalue les performances à l'aide des métriques suivantes :

— **Accuracy** : proportion de prédictions correctes.
— **Précision (Precision)** : proportion de vrais positifs parmi les positifs prédits.
— **Rappel (Recall)** : proportion de vrais positifs détectés.

En plus des matrices de confusion, on utilise :

— **Courbes Precision–Recall (PR)** : précision en fonction du rappel selon le seuil (utile si classes déséquilibrées).

5.1 Résultats sur le Colorado (CO)

Performances globales

TABLE 5.1 – Résultats en inférence sur l'État CO

Modèle	Accuracy	Précision	Rappel
Random Forest	0.7805	0.6926	0.8455
AdaBoost	0.7766	0.6895	0.8384
XGBoost	**0.7881**	**0.6973**	**0.8632**

Matrices de confusion

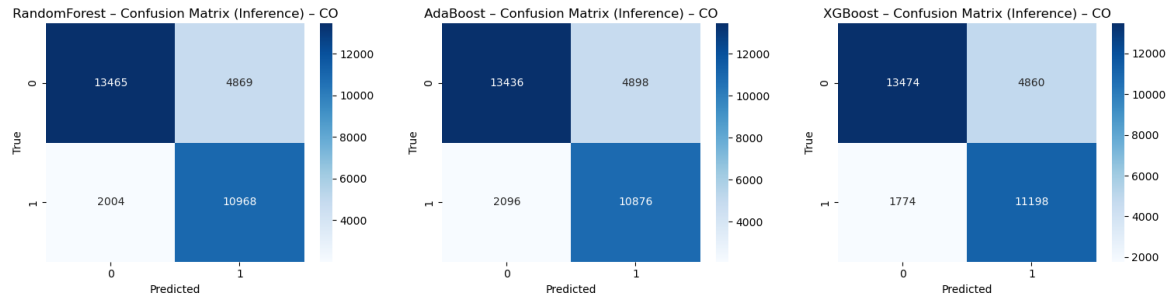


FIGURE 5.1 – Matrices de confusion (CO) en inférence

Courbes Precision–Recall

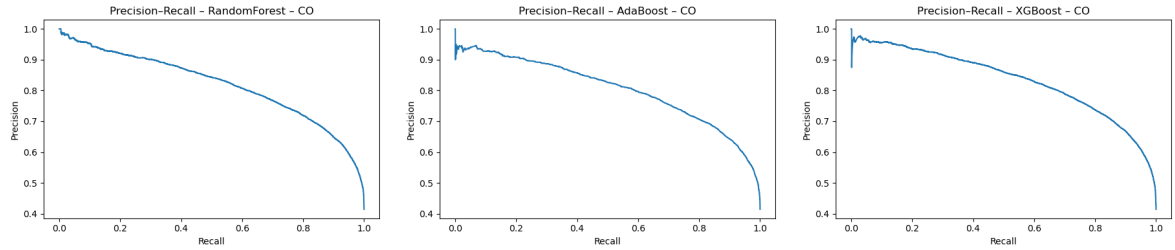


FIGURE 5.2 – Courbes Precision–Recall (CO). Elles montrent le compromis précision/rappel selon le seuil.

Analyse CO Sur le Colorado, **XGBoost** obtient les meilleures performances globales en termes d'accuracy. Les trois modèles présentent un **rappel élevé** (≈ 0.84–0.86), indiquant une bonne détection des individus ayant un revenu $> 50k$. Cependant, la précision reste modérée (≈ 0.69), ce qui met en évidence l'existence d'un nombre non négligeable de **faux positifs**. Les courbes de rappel et de précision en fonction du seuil montrent qu'une augmentation du seuil améliore généralement la précision au détriment du rappel.

5.2 Résultats sur le Nevada (NE)

Performances globales

TABLE 5.2 – Résultats en inférence sur l'État NE

Modèle	Accuracy	Précision	Rappel
Random Forest	0.7371	0.5509	**0.8560**
AdaBoost	0.7317	0.5464	0.8281
XGBoost	**0.7446**	**0.5596**	0.8557

Matrices de confusion

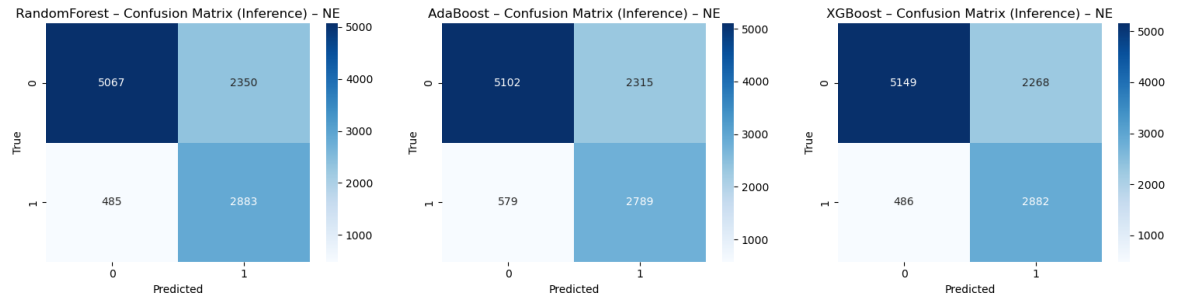


FIGURE 5.3 – Matrices de confusion (NE) en inférence

Courbes Precision–Recall

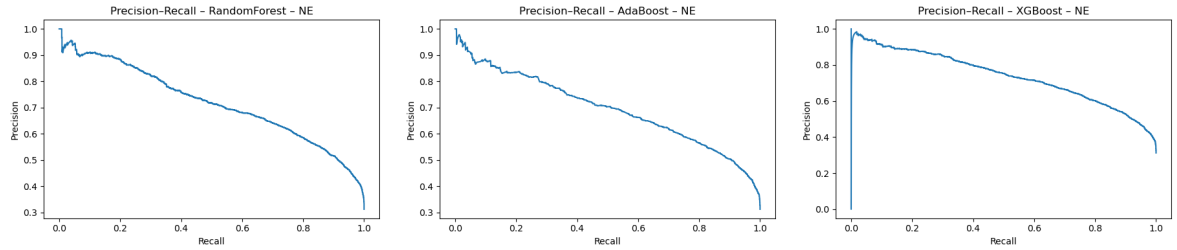


FIGURE 5.4 – Courbes Precision–Recall (NE). Elles montrent une précision plus faible pour un rappel élevé.

Analyse NE Sur le Nevada, les performances chutent davantage (accuracy ≈ 0.73–0.74). La **précision** est nettement plus faible (≈ 0.55) alors que le **rappel** reste très élevé (≈ 0.83–0.86). Cela signifie que les modèles détectent bien les individus appartenant à la classe positive, mais produisent un nombre important de **faux positifs**. Ce comportement est cohérent avec un changement de distribution entre la Californie et le Nevada (différences économiques, structure des métiers, proportion de revenus élevés, etc.). Un ajustement du seuil de décision pourrait permettre d'améliorer la précision au détriment du rappel.

Commentaires et conclusion

Les résultats confirment que l'application directe d'un modèle optimisé sur un État (Californie) vers d'autres États induit une baisse de performance, ce qui est attendu en présence d'un **décalage de distribution**. **XGBoost** se montre le plus robuste et obtient les meilleures performances globales sur CO et NE en termes d'accuracy. Les courbes de précision et de rappel montrent que la capacité de généralisation reste satisfaisante, mais que la calibration du seuil de décision influence fortement le compromis précision/rappel, en particulier sur le Nevada où la précision est plus faible.

Chapitre 6

Expérimentation 5 : impact de la taille du jeu de données

Objectif

Cette expérimentation vise à analyser l'impact de la **taille du jeu d'entraînement** sur la qualité des prédictions. On utilise uniquement les **meilleurs hyperparamètres** obtenus lors de l'Expérimentation 2 (GridSearchCV). L'évaluation est réalisée sur un **ensemble de test fixe** (identique pour toutes les tailles), afin d'observer comment la performance évolue lorsque l'on augmente progressivement la quantité de données disponibles pour l'apprentissage.

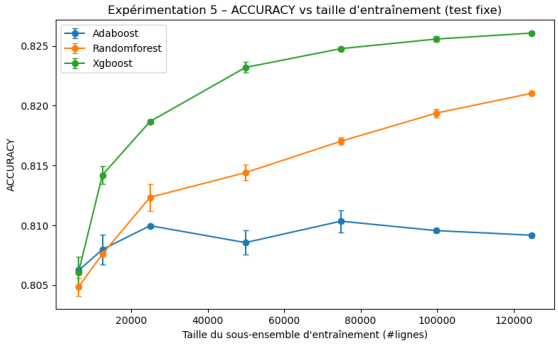
Protocole expérimental

Soit D_{train} l'ensemble d'entraînement complet. On construit des sous-ensembles **stratifiés** (afin de préserver la proportion des classes) de taille croissante (par exemple 5%, 10%, 20%, 40%, 60%, 80%, 100% de D_{train}). Pour chaque taille :

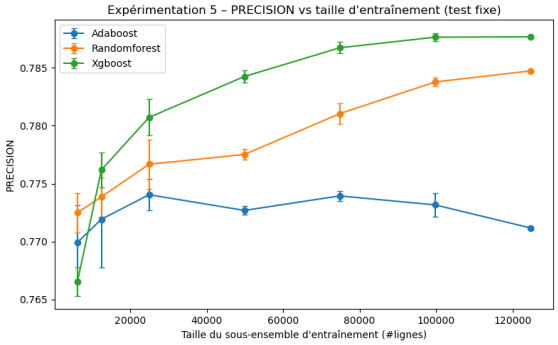
- On entraîne chaque modèle (*RandomForest*, *AdaBoost*, *XGBoost*) avec ses hyperparamètres optimaux.
- On évalue sur le même jeu de test fixe.
- On mesure : **accuracy**, **precision**, **recall**, **F1-score** et le **temps d'entraînement**.

Remarque (barres d'erreur). Les traits verticaux présents sur les courbes correspondent aux **barres d'erreur** : elles représentent l'**écart-type** des scores obtenus sur plusieurs répétitions (plusieurs sous-échantillonnages). Des barres d'erreur grandes indiquent une performance plus **instable** (variance élevée), ce qui arrive souvent lorsque la taille d'entraînement est faible.

Courbes de performance en fonction de la taille



(a) Accuracy



(b) Précision

FIGURE 6.1 – Accuracy et précision en fonction de la taille du jeu d'entraînement (test fixe).

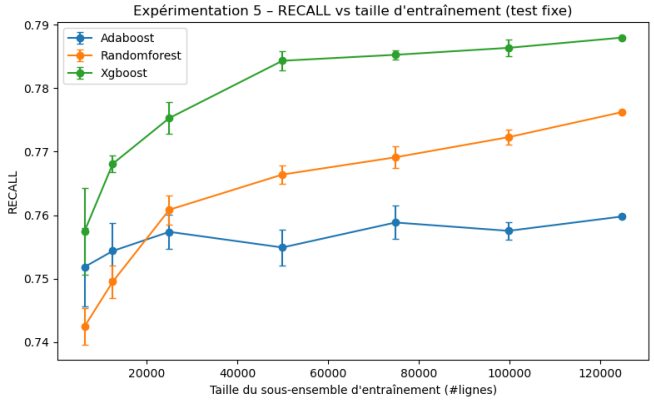


FIGURE 6.2 – Rappel en fonction de la taille du jeu d'entraînement (test fixe).

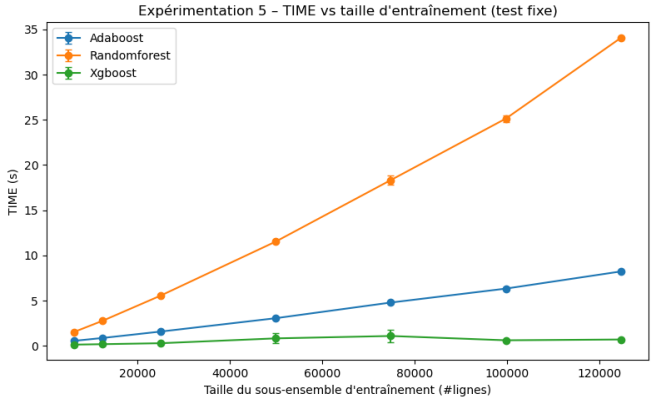


FIGURE 6.3 – Temps d'entraînement en fonction de la taille du jeu d'entraînement.

Comparaison finale sur la taille maximale

Sur la taille maximale ($n_{train} \approx 124736$), les performances finales observées sont :
— **Accuracy :** XGBoost (0.8261) > RandomForest (0.8210) > AdaBoost (0.8092)
— **Précision :** XGBoost (0.7877) > RandomForest (0.7847) > AdaBoost (0.7712)

— **Rappel :** XGBoost (0.7879) > RandomForest (0.7762) > AdaBoost (0.7598)

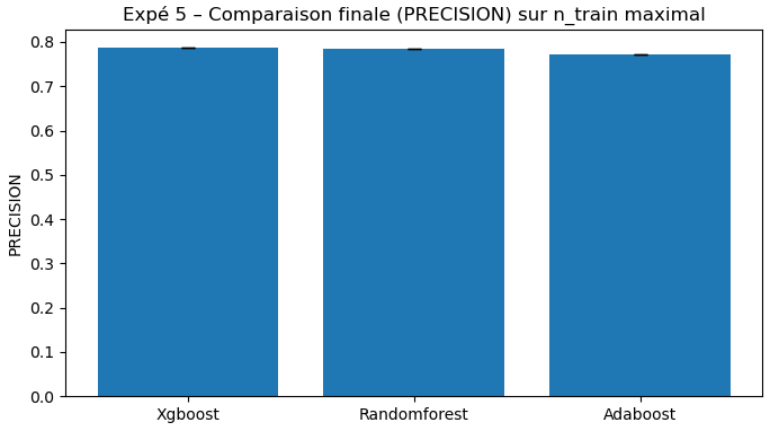


FIGURE 6.4 – Comparaison finale de la précision sur la taille maximale ($n_{train} \approx 124736$).

Résultats / Commentaires / Analyses

Effet de la taille sur la qualité de prédiction Les courbes montrent que l'augmentation du nombre d'exemples d'entraînement améliore globalement les performances sur le jeu de test, en particulier pour **l'accuracy**. Avec peu de données, les modèles sont plus sensibles au sous-échantillonnage, ce qui se traduit par une variance plus élevée et des performances moins stables. Lorsque la taille d'entraînement augmente, les performances se stabilisent progressivement.

Compromis précision / rappel L'analyse conjointe des courbes de **précision** et de **rappel** met en évidence un compromis classique : une amélioration de la précision peut s'accompagner d'une légère variation du rappel. **XGBoost** présente globalement le meilleur équilibre entre ces deux métriques sur la majorité des tailles d'entraînement.

Phénomène de saturation Un ralentissement de la progression des performances est observé lorsque la taille d'entraînement se rapproche de son maximum : les courbes tendent vers un plateau. Cela suggère que, passé un certain volume de données, les gains deviennent marginaux et sont principalement limités par la capacité du modèle et l'information contenue dans les attributs.

Comparaison des modèles Sur l'ensemble des tailles considérées, **XGBoost** obtient les meilleures performances globales en test, suivi de **RandomForest**. **AdaBoost** progresse plus lentement avec l'augmentation de la taille d'entraînement, indiquant une capacité plus limitée dans ce contexte.

Coût computationnel Le temps d'entraînement augmente naturellement avec la taille du jeu d'entraînement. Sur la taille maximale ($n_{train} \approx 124736$), on observe :
- RandomForest : ≈ 34.10 s (coût le plus élevé)
- AdaBoost : ≈ 8.23 s
- XGBoost : ≈ 0.70 s (le plus rapide ici)

Ainsi, **XGBoost** présente le meilleur compromis performance/temps dans ce cadre expérimental.

Conclusion

L'augmentation de la taille du jeu d'entraînement améliore la qualité des prédictions, avec un effet de **saturation** lorsque l'on approche de la taille maximale. Dans ce cadre, **XGBoost optimisé** est le meilleur choix : il obtient les meilleures performances et reste le plus efficace en temps de calcul.

Chapitre 7

3 Explicabilité des prédictions

7.1 3.1 Classement des attributs dans la prédiction (Permutation Feature Importance)

Modèles étudiés et protocole

Dans cette partie, nous analysons globalement les prédictions des trois modèles optimisés (Expérimentation 2) : **Random Forest**, **AdaBoost** et **XGBoost**. Les importances sont estimées sur l'ensemble de test à l'aide d'une méthode de **Permutation Feature Importance** (importance par permutation), en utilisant le **F1-score** comme métrique (plus robuste que l'accuracy en cas de déséquilibre de classes). Pour améliorer la lisibilité (présence de variables encodées), les importances ont été **agrégées par attribut d'origine** (`perm_importance_grouped_top15`).

Principe de la méthode

L'idée est de mesurer la contribution d'un attribut X_j au pouvoir prédictif du modèle :

1. On calcule un score de référence S du modèle sur un ensemble d'évaluation (ici, le F1-score).
2. Pour un attribut X_j, on **permute aléatoirement** ses valeurs dans les données d'évaluation, en conservant toutes les autres colonnes inchangées. Cette permutation détruit le lien statistique entre X_j et la cible.
3. On recalcule le score S_j^{perm}. La **baisse** $\Delta_j = S - S_j^{perm}$ quantifie l'importance de X_j : si Δ_j est grand, le modèle dépend fortement de cet attribut.
4. L'opération est répétée plusieurs fois (répétitions), et l'on reporte la moyenne et l'écart-type de Δ_j.

Cette méthode est applicable à tout classifieur et fournit une explication **globale** des variables les plus influentes.

XGBoost – Permutation Feature Importance (groupée)

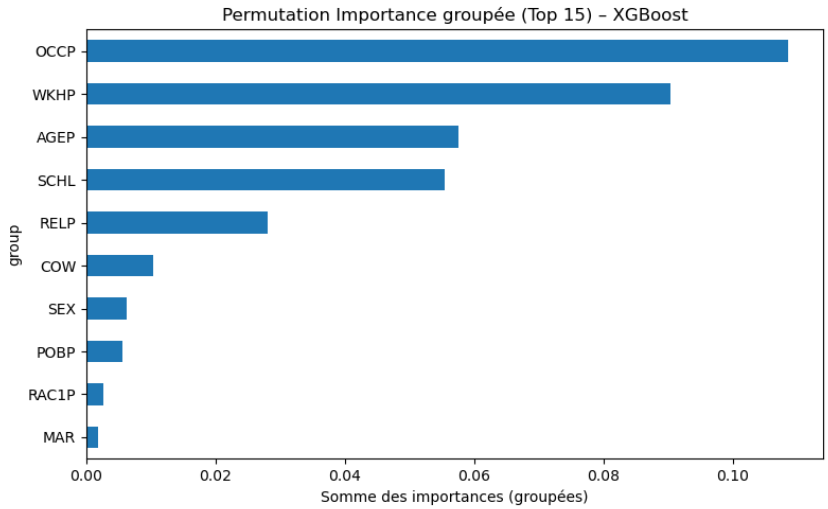


FIGURE 7.1 – Permutation feature importance groupée (XGBoost) – Top attributs (baisse du F1)

Interprétation et synthèse. L'analyse par permutation met en évidence un noyau restreint de variables dominantes dans les décisions du modèle XGBoost. L'attribut **OCCP** (profession) apparaît comme le facteur le plus influent : la permutation de cette variable entraîne la plus forte baisse du score F1, ce qui indique que XGBoost exploite fortement l'information portée par la profession. Son écart-type très faible suggère une importance **stable et robuste** à travers les permutations.

Les variables **WKHP** (heures travaillées), **AGEP** (âge) et **SCHL** (niveau d'éducation) suivent immédiatement. Ces résultats confirment que la prédiction d'un revenu supérieur à 50k repose principalement sur des dimensions liées au **capital humain** (éducation et expérience), au **temps de travail** et au **secteur professionnel**. Les attributs **RELP** et **COW** jouent un rôle secondaire, tandis que les variables sensibles telles que **SEX** ou **RAC1P** ont une contribution relativement faible comparée aux facteurs socio-professionnels.

Apport pour la compréhension des prédictions. La permutation importance permet d'identifier clairement **quels attributs pilotent réellement la décision du modèle**. Elle fournit une validation empirique de la cohérence des prédictions avec des connaissances socio-économiques établies, tout en mettant en évidence que les variables sensibles ne constituent pas les principaux moteurs de la classification.

Inférence qualitative « à la main ». Bien que le calcul exact d'une prédiction soit difficile en raison de la nature ensembliste du modèle, il est possible de raisonner qualitativement : un individu présentant un **WKHP** élevé, un **niveau d'éducation élevé** (**SCHL**) et une **profession qualifiée** (**OCCP**) aura une probabilité accrue d'être classé dans la catégorie *revenu* $> 50k$. À l'inverse, une combinaison de faible temps de travail, de niveau d'éducation plus bas et d'occupation moins qualifiée rend la classe $\leq 50k$ plus probable.

Chapitre 8

Explicabilité : avec LIME et SHAP

Dans ce chapitre, nous analysons le modèle (XGBoost optimisé) à l'aide d'outils d'explicabilité **locales** (explication d'un individu) et **globales** (tendances générales sur le jeu de test). Les figures proviennent des sorties enregistrées dans `tests/exp7/`.

8.1 Méthode LIME

Principe

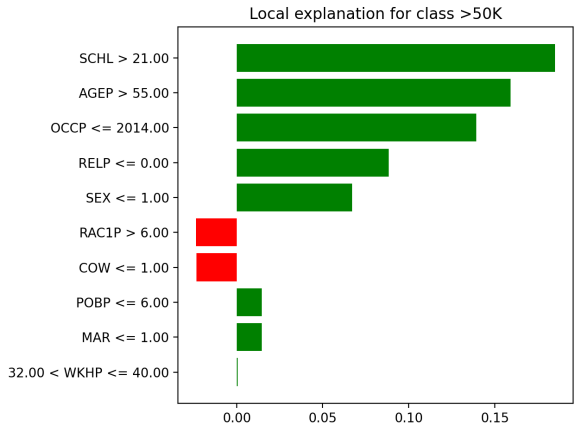
LIME (*Local Interpretable Model-agnostic Explanations*) explique une prédiction en construisant une approximation **locale** du modèle complexe autour d'un exemple x. L'idée est de : (i) générer des points proches de x (perturbations), (ii) obtenir les prédictions du modèle, (iii) ajuster un modèle interprétable (souvent linéaire) sur ces points, avec un poids de proximité plus fort près de x.

Comment lire un graphique LIME

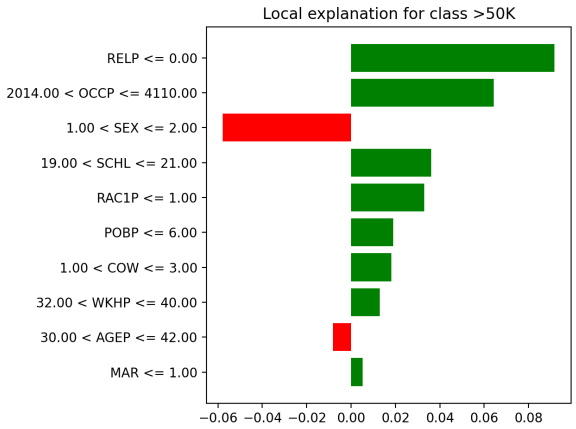
Dans les graphiques LIME pour la classe `>50K` :
— les **barres vertes** poussent la prédiction vers `>50K` ;
— les **barres rouges** poussent la prédiction vers $\leq$`50K` ;
— chaque ligne est une règle (ex. `WKHP > 40`) correspondant à une **discrétisation** locale (LIME transforme souvent les variables numériques en intervalles).
— la longueur de la barre est l'**importance locale** (contribution) dans l'approximation linéaire.

Exemples choisis

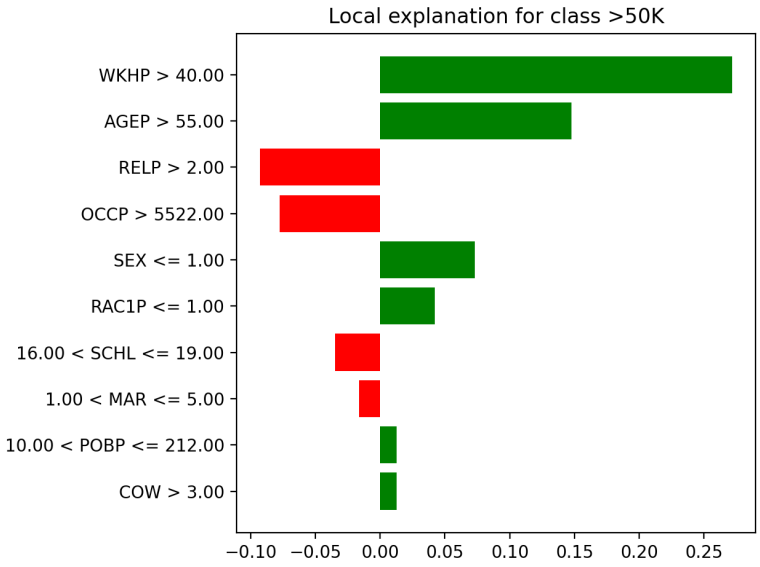
Nous illustrons trois explications locales LIME (trois individus du jeu de test).



(a) Exemple idx163305



(b) Exemple idx134592



(c) Exemple idx35822

FIGURE 8.1 – Explications locales LIME pour la classe `>50K`.

Résultats et commentaires

Sur la Figure 8.1, on observe que les attributs dominants localement sont souvent : **WKHP** (heures travaillées), **AGEP** (âge), **SCHL** (niveau d'étude), et **OCCP** (occupation).

- Pour l'exemple **idx163305**, `WKHP > 40` et `AGEP > 55` contribuent fortement **positivement** vers `>50K` (barres vertes). Certaines modalités de `RELP` ou `OCCP` tirent au contraire la décision vers $\leq$`50K` (barres rouges).
- Pour l'exemple **idx134592**, le niveau d'éducation (`SCHL > 21`) et l'âge poussent vers `>50K`, tandis que certaines modalités (ex. `RAC1P` ou `COW`) peuvent compenser négativement.
- Pour **idx35822**, l'effet de `SEX` apparaît négatif localement, alors que `OCCP` et `RELP` poussent positivement : cela illustre qu'une même variable peut avoir un **impact local** non négligeable, mais cet impact est **spécifique à l'individu** et à son voisinage local.

Limites (important). LIME dépend du voisinage généré (perturbations) et de la discrétisation : deux exécutions (ou un réglage différent) peuvent produire des poids légèrement différents. Les contributions sont **locales** et ne se généralisent pas directement.

8.2 Méthode SHAP

Principe

SHAP (*SHapley Additive exPlanations*) attribue à chaque attribut une contribution inspirée des valeurs de Shapley. SHAP fournit des explications additives :

$$f(x) \approx E[f(X)] + \sum_j \phi_j$$

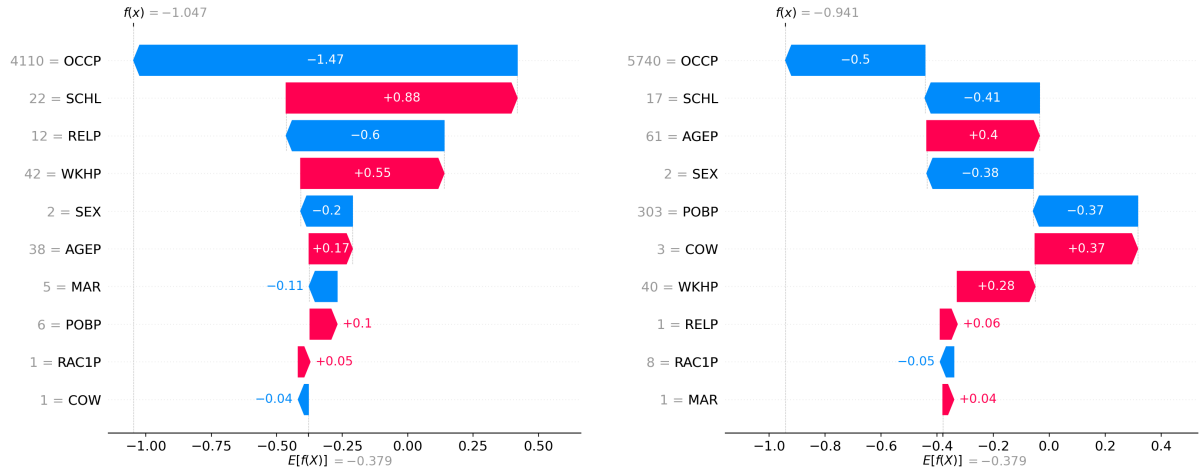
où $E[f(X)]$ est une valeur de base (moyenne du modèle) et ϕ_j la contribution de l'attribut j.

Comment lire un Waterfall plot SHAP

Dans un **waterfall plot** :
— on part de la valeur de base $E[f(X)]$ (baseline),
— chaque barre ajoute (ou retranche) une contribution SHAP,
— les contributions **positives** poussent la prédiction vers la classe `>50K`,
— les contributions **négatives** la poussent vers $\leq$`50K`.

Remarque : pour XGBoost, SHAP affiche souvent $f(x)$ sur l'échelle interne du modèle (souvent proche des *log-odds*). Le signe et la taille restent interprétables : positif $\Rightarrow$ augmente la probabilité de `>50K`, négatif $\Rightarrow$ la diminue.

Exemples choisis (Waterfall)



(a) Waterfall idx107716

(b) Waterfall idx164515

FIGURE 8.2 – Explications locales SHAP (waterfall).

Résultats et commentaires

La Figure 8.2 met en évidence des contributions fortes de : **OCCP**, **SCHL**, **WKHP**, **AGEP** et parfois **RELP** ou **SEX**. On observe des cas où :
— certaines modalités de **OCCP** tirent fortement la prédiction vers $\leq$`50K` (grosses contributions négatives),
— un niveau d'étude élevé (**SCHL**) et un nombre d'heures élevé (**WKHP**) compensent partiellement (contributions positives),
— l'addition de plusieurs petites contributions peut inverser la décision ou la renforcer.

8.3 Comparaison LIME et SHAP

Points communs

Sur nos exemples, LIME et SHAP identifient des variables récurrentes (**WKHP**, **SCHL**, **AGEP**, **OCCP**, **RELP**) comme étant déterminantes, ce qui renforce la cohérence de l'explication.

Différences

— **LIME** : explication basée sur une approximation linéaire locale. Très lisible (barres vert/rouge), mais sensible au voisinage perturbé et à la discrétisation.
— **SHAP** : explication additive plus "théorique", souvent plus stable et comparable entre individus. Permet aussi une analyse globale (summary plot).

8.4 Analyse summary-plot de SHAP

Comment lire un summary plot (beeswarm)

Dans un **summary plot** SHAP :
— l'axe x = valeur SHAP (impact sur la sortie du modèle) : **à droite** ⇒ augmente la probabilité de `>50K`, **à gauche** ⇒ la diminue.
— chaque point = un individu du jeu de test.
— la couleur = valeur de la feature (bleu = faible, rouge = élevée).
— les features sont triées (haut ⇒ plus important) selon la moyenne de $|\phi_j|$.
Attention : si une variable est un **code catégoriel** (ex. `OCCP`), la notion "valeur élevée/faible" n'a pas forcément un sens ordinal ; la couleur indique seulement un code numérique plus grand/petit.

Résultats globaux

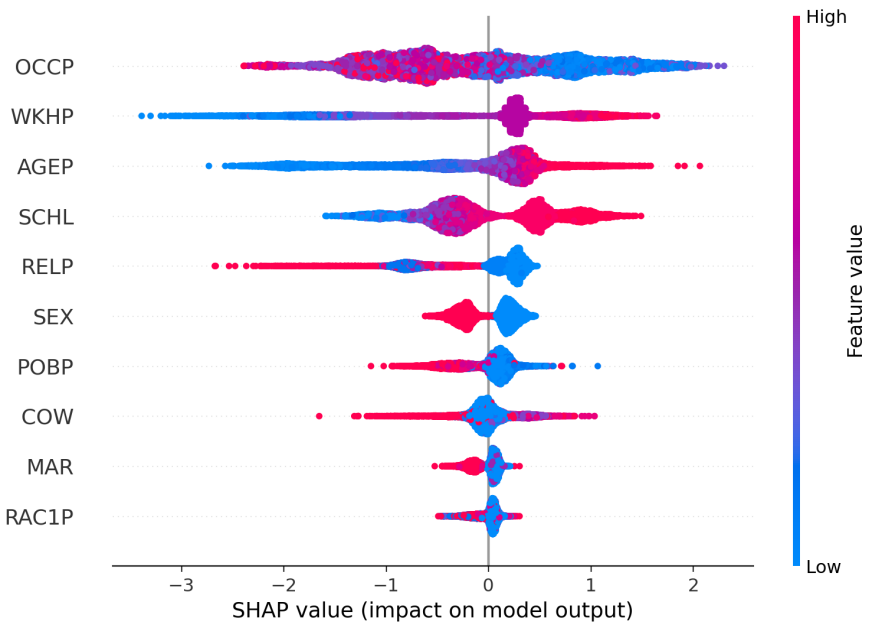
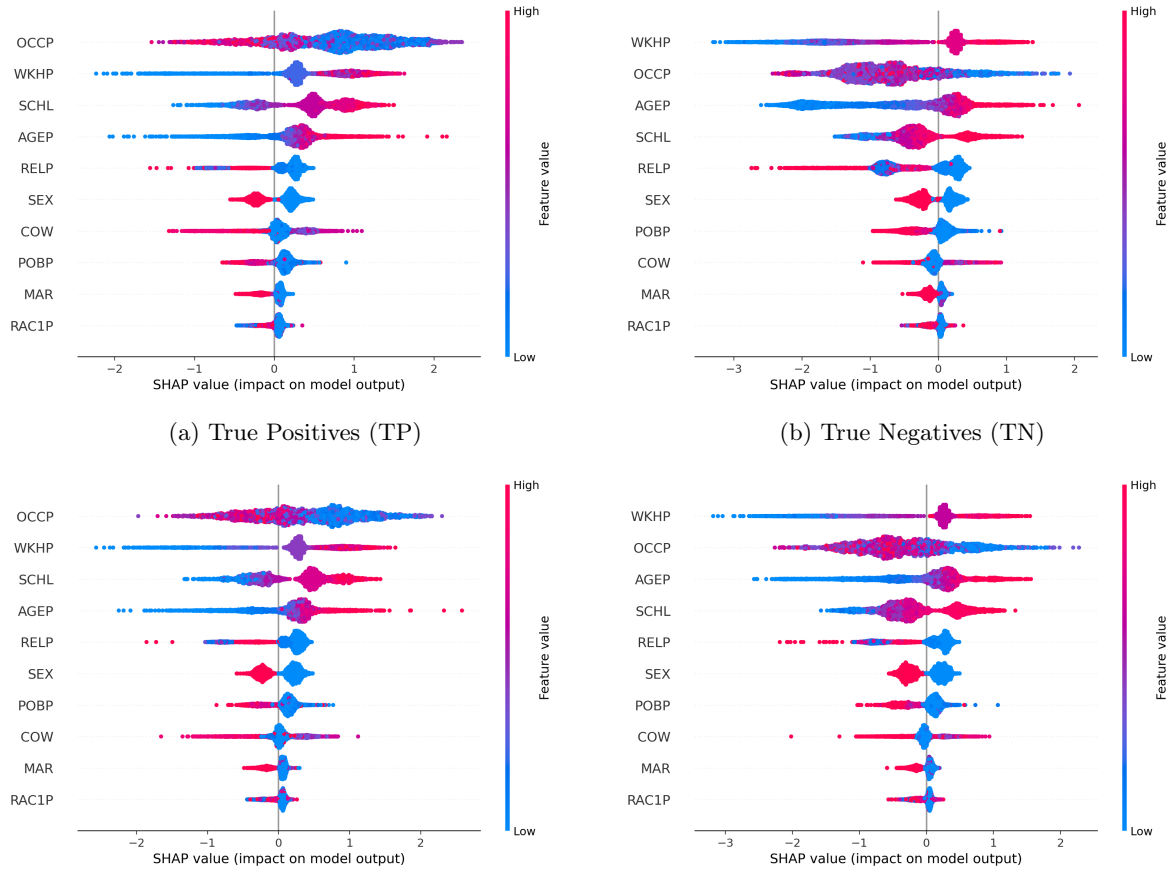


FIGURE 8.3 – SHAP summary plot global (top features).

Sur la Figure 8.3, les attributs les plus influents sont typiquement **OCCP**, **WKHP**, **AGEP**, **SCHL**, puis **RELP**. On observe notamment :
— **WKHP** : les valeurs élevées (points rouges) ont tendance à être associées à des SHAP positifs, ce qui est cohérent avec l'idée qu'un temps de travail important augmente la probabilité de revenus élevés.

— **AGEP** et **SCHL** : des valeurs plus élevées génèrent fréquemment des contributions positives, reflétant l'effet de l'expérience et du niveau d'études.
— **OCCP** : forte dispersion des contributions, ce qui est attendu car l'occupation discrimine fortement les revenus, mais l'interprétation "croissante" est limitée car `OCCP` est un code catégoriel.

Approfondissement : summary plot par sous-groupes TP/TN/FP/FN



(a) True Positives (TP)

(b) True Negatives (TN)

(c) False Positives (FP)

(d) False Negatives (FN)

FIGURE 8.4 – SHAP summary plots par sous-groupe (TP/TN/FP/FN).

La Figure 8.4 aide à comprendre **où le modèle se trompe** :
— Dans les **TP**, on voit souvent des contributions positives cohérentes (heures travaillées élevées, niveau d'études, âge), ce qui renforce correctement la classe `>50K`.
— Dans les **TN**, les contributions sont majoritairement négatives : certains codes d'occupation et faibles valeurs de `WKHP` tirent le modèle vers $\leq$`50K`.
— Dans les **FP**, certains individus reçoivent des contributions positives fortes (ex. `WKHP` ou `SCHL`) mais la vérité terrain est $\leq$`50K`. Cela suggère que le modèle peut **sur-interpréter** certains signaux comme "revenu élevé" alors que d'autres facteurs non observés (ou mal encodés, notamment via `OCCP`) devraient contredire.
— Dans les **FN**, l'inverse : malgré des signaux positifs, une ou plusieurs variables (souvent `OCCP`/`RELP`) tirent trop la prédiction vers $\leq$`50K`. On peut donc identifier des attributs susceptibles de **tromper le modèle** lorsqu'ils prennent certaines modalités.

Conclusion. LIME et SHAP confirment des déterminants majeurs (WKHP, SCHL, AGEP, OCCP) et fournissent une lecture complémentaire : LIME est très intuitif pour expliquer *un* individu, tandis que SHAP permet une vision locale **et** globale, et met en évidence les patterns d'erreurs via TP/TN/FP/FN.

Chapitre 9

Explication contrefactuelle

Principe et objectif L'objectif d'une explication contrefactuelle est d'étudier, pour un individu donné (un exemple du jeu de test), **quelles modifications minimales des attributs** pourraient **inverser la prédiction**. Ici, on réalise une analyse *one-feature-at-a-time* (un attribut à la fois) : toutes les variables sont figées à leur valeur initiale, et on fait varier un seul attribut sur une plage de valeurs. Pour chaque valeur testée, on calcule la probabilité prédite $P(y = \text{True})$ (ici $P(> 50K)$). On vérifie ensuite si la courbe franchit le seuil de décision 0,5 (ligne horizontale pointillée), ce qui indique qu'un changement de cet attribut **peut retourner la classe prédite**.

Sur chaque courbe :

— l'axe x représente la valeur testée de l'attribut (ex : AGEP, WKHP, OCCP, RELP),

— l'axe y représente $P(> 50K)$,

— le point marque la valeur réelle de l'individu,

— la ligne pointillée à 0,5 est le seuil de bascule entre les classes.

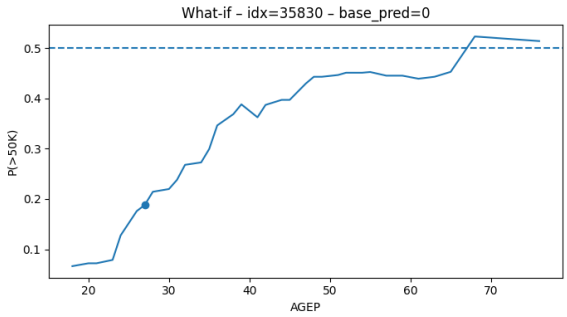
Exemples analysés et résultats (bascule possible) Quatre exemples ont été sélectionnés dans les quatre sous-groupes (TP, TN, FP, FN). Le Tableau 9.1 résume un attribut pour lequel une inversion est possible.

TABLE 9.1 – Exemples contrefactuels : un attribut pouvant inverser la prédiction.

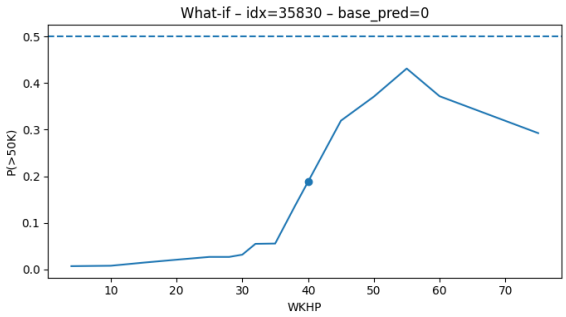
Groupe	idx	Attribut	Pred. base	P_0	Valeur modifiée	P_{new}
FN	35830	COW	0	0.188	7.0	0.526
TP	58567	SCHL	1	0.854	13.0	0.460
TN	98530	OCCP	0	0.075	3255.0	0.519
FP	150026	RELP	1	0.730	17.0	0.213

Interprétation rapide. On observe que des attributs comme **AGEP**, **WKHP**, **SCHL**, **RELP** (et parfois **OCCP**) peuvent fortement déplacer $P(> 50K)$ et donc inverser la classe prédite. Cela est cohérent avec les résultats d'importance globale (permutation importance) où ces attributs figurent parmi les plus influents.

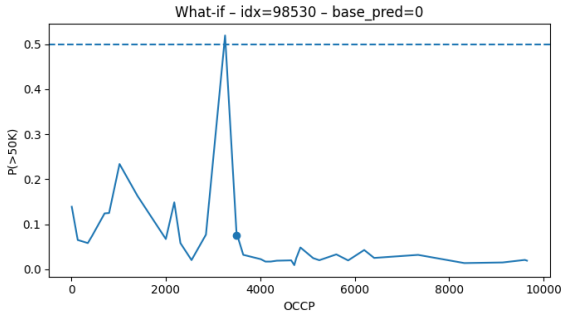
Courbes what-if (sensibilité + inversion) La Figure 9.1 illustre des courbes *what-if* typiques.



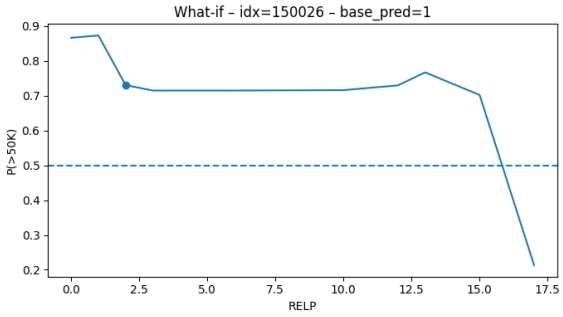
(a) FN (idx=35830) : variation de AGEP



(b) FN (idx=35830) : variation de WKHP



(c) TN (idx=98530) : variation de OCCP



(d) FP (idx=150026) : variation de RELP

FIGURE 9.1 – Courbes contrefactuelles (what-if) : $P(> 50K)$ en fonction d'un attribut, toutes choses égales par ailleurs.

Analyse et commentaires

1) Cas FN (idx=35830). La prédiction initiale est négative ($P_0 = 0.188$). En augmentant **AGEP** (ou en modifiant **COW** selon les tests), la probabilité franchit le seuil de 0.5, ce qui permet de **corriger** ce faux négatif. Cela suggère que, pour cet individu, le modèle est très sensible à des attributs socio-professionnels et démographiques.

2) Cas FP (idx=150026). La prédiction initiale est positive ($P_0 = 0.730$) alors que la vérité est négative. En faisant varier **RELP**, la probabilité peut chuter fortement ($P_{new} = 0.213$), ce qui **annule** la prédiction erronée. Cela indique que certaines modalités (ou valeurs) de RELP peuvent **tromper** le modèle sur cet exemple.

3) Cas TN (idx=98530) et attribut OCCP. On observe une bascule possible via **OCCP**, mais la variation nécessaire peut être difficile à interpréter car OCCP est un **code catégoriel** (encodé numériquement). Faire varier OCCP comme une variable continue est une approximation : le résultat doit être lu comme une **sensibilité aux changements de catégorie** plutôt qu'une relation monotone réelle.

Limites. Cette approche modifie un seul attribut à la fois et ne garantit pas que l'exemple modifié soit réaliste (cohérence entre variables, valeurs autorisées pour les catégories, etc.). Néanmoins, elle fournit une lecture claire de la **direction d'influence** et de la **capacité à inverser** la décision.

Conclusion. Les courbes contrefactuelles confirment que les variables dominantes (WKHP, SCHL, AGEP, RELP, OCCP) peuvent suffire à inverser certaines décisions. Elles complètent les explications globales (importance par permutation) en donnant une lecture **locale et actionnable** sur des cas TP/TN/FP/FN.