INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Rapport TP Apprentisssage Supervisé

Imad Boukezatta

Hazem Mejri

5 SDBD B1

19 décembre 2024

Lien vers depot

Introduction:

Ce travail pratique se concentre sur l'application de modèles d'ensemble (comme Random Forest, AdaBoost, XGBoost et un modèle de stacking SVM) sur le jeu de données **ACSIncome** de l'État de Californie. L'objectif est d'optimiser ces modèles pour obtenir des prédictions précises et de les évaluer en termes de performance, d'interprétabilité et d'équité statistique.

Au terme de ce travail, un rapport détaillant les différentes étapes de l'analyse, les choix méthodologiques, ainsi que les résultats et leur interprétation, sera présenté. Ce rapport permettra d'évaluer non seulement la qualité des modèles obtenus, mais aussi leur pertinence dans un contexte éthique et équitable.

0. Description du dataset ACSIncome

Le dataset **ACSIncome** vise à prédire si le revenu annuel d'un adulte américain est supérieur à 50 000 $. La variable cible est le **PINCP** (revenu total de la personne), où une étiquette de 1 est attribuée si **PINCP > 50 000**, sinon l'étiquette est 0. Le dataset contient différentes variables représentant des informations démographiques, professionnelles et éducatives sur les individus. Parmi les principales caractéristiques, on trouve :

- **AGEP** (Âge) : Valeurs comprises entre **16** et 99 (entiers).
- **COW** (Classe de travailleur) : Valeurs décrivant différents types d'emploi, de l'employé d'une entreprise privée aux travailleurs sans emploi.
- **SCHL** (Niveau d'éducation) : Valeurs allant de "aucune scolarité" à un "doctorat", représentant le plus haut niveau d'éducation atteint.
- **MAR** (État matrimonial) : Valeurs indiquant si une personne est mariée, veuve, divorcée, séparée, ou jamais mariée.
- **OCCP** (Occupation) : Codes d'occupation détaillant le type de travail occupé par une personne.
- **POBP** (Lieu de naissance) : Valeurs représentant le pays ou l'État de naissance de la personne.
- **RELP** (Relation familiale) : Valeurs représentant le lien de parenté de l'individu avec la personne de référence (par exemple, mari, fils, frère).
- **WKHP** (Heures habituelles travaillées par semaine) : Nombre d'heures travaillées chaque semaine (**N/A** pour les personnes ne travaillant pas).
- **SEX** (Sexe) : Homme ou femme.
- **RAC1P** (Race détaillée) : Codes indiquant la race de la personne, avec des catégories pour les Blancs, Noirs, Amérindiens, Asiatiques, etc.

1.Data visualization

1.1 Data.info()

Renvoie le nombre et le type des lignes , ainsi le nombre de valeurs non nulles de chaque colonne

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 166315 entries, 0 to 166314
Data columns (total 10 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   AGEP    166315 non-null  float64
 1   COW     166315 non-null  float64
 2   SCHL    166315 non-null  float64
 3   MAR     166315 non-null  float64
 4   OCCP    166315 non-null  float64
 5   POBP    166315 non-null  float64
 6   RELP    166315 non-null  float64
 7   WKHP    166315 non-null  float64
 8   SEX     166315 non-null  float64
 9   RAC1P   166315 non-null  float64
dtypes: float64(10)
memory usage: 12.7 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 166315 entries, 0 to 166314
Data columns (total 1 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   PINCP   166315 non-null  bool
dtypes: bool(1)
memory usage: 162.5 KB
(None, None)
```

1.2 Data.head()

Renvoie les 5 premieres lignes du dataset

```
(   AGEP  COW  SCHL  MAR   OCCP   POBP  RELP  WKHP  SEX  RAC1P
0  41.0  4.0  24.0  1.0  2555.0   6.0   1.0  60.0  2.0    1.0
1  77.0  7.0  22.0  1.0  4920.0  39.0   0.0  35.0  1.0    1.0
2  38.0  1.0  18.0  1.0   440.0   6.0   1.0  50.0  1.0    1.0
3  30.0  1.0  22.0  5.0  1555.0   6.0   2.0  80.0  1.0    6.0
4  36.0  1.0  16.0  1.0  4030.0 314.0   1.0  70.0  2.0    1.0,
   PINCP
0   True
1   True
2  False
3   True
4  False)
```

1.3 Data.describe()

Donne des informations utiles quant au caracteres des variables

```
(              AGEP            COW           SCHL            MAR  \
count  166315.000000  166315.000000  166315.000000  166315.000000
mean       42.736235       2.144551      18.470054       2.653633
std        14.882790       1.888220       3.938362       1.846417
min        17.000000       1.000000       1.000000       1.000000
25%        30.000000       1.000000      16.000000       1.000000
50%        42.000000       1.000000      19.000000       1.000000
75%        55.000000       3.000000      21.000000       5.000000
max        94.000000       8.000000      24.000000       5.000000

              OCCP           POBP           RELP           WKHP  \
count  166315.000000  166315.000000  166315.000000  166315.000000
mean     4019.729279      94.364718       2.506617      37.859255
std      2638.167883     123.472067       4.443905      13.014087
min        10.000000       1.000000       0.000000       1.000000
25%      2014.000000       6.000000       0.000000      32.000000
50%      4110.000000       9.000000       1.000000      40.000000
75%      5521.000000     212.000000       2.000000      40.000000
max      9830.000000     554.000000      17.000000      99.000000

                SEX          RAC1P
count  166315.000000  166315.000000
mean        1.471972       3.071623
std         0.499215       2.916045
min         1.000000       1.000000
25%         1.000000       1.000000
50%         1.000000       1.000000
75%         2.000000       6.000000
max         2.000000       9.000000 ,
         PINCP
count   166315
unique       2
top      False
freq     98112)
```

1.4 Relation entre features

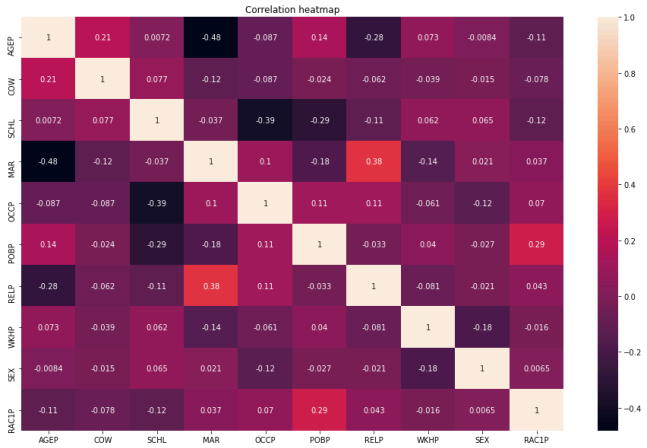


Figure 1: Heatmap de Corrélation des features

Les variables **AGEP**, **MAR**, **POBP** et **RELP** montrent les corrélations les plus **significatives**, en particulier vis-à-vis du statut marital et de la race. Les autres variables, comme **SEX**, **SCHL**, et **OCCP**, présentent des relations plus **faibles**, indiquant des interactions moins directes avec les autres caractéristiques. Ces informations peuvent être utiles pour identifier les **biais** ou les facteurs importants lors de la construction du modèle.

2.Data preprocessing § Préparation des données

2.1 Nettoyage des variables nulles

- data.isnull().sum() renvoie le nombre de donnée qui n'ont pas de valeur nulle

```
dataset.isnull().sum()
```

	0
AGEP	166315
COW	166315
SCHL	166315
MAR	166315
OCCP	166315
POBP	166315
RELP	166315
WKHP	166315
SEX	166315
RAC1P	166315
PINCP	166315

dtype: int64

2.2 Mélanger et travailler sur une fraction du code

- **Mélanger les indices**: Réduire les **biais** potentiels liés à l'ordre initial des données et pour s'assurer que les modèles d'apprentissage ne soient pas influencés par des séquences spécifiques présentes dans le dataset.

```
indices = shuffle(range(len(features)), random_state=1)
X_all = features.iloc[indices]
y_all = labels.iloc[indices]
```

- Partitionner le jeu de données : **100%**

```
num_samples = int(len(X_all) ) # * 0.1 si 10%
X, y = X_all.iloc[:num_samples], y_all.iloc[:num_samples]
```

2.3 Séparer le set

- Division des ensembles de données en sous-ensembles d'entraînement et de test (80%)

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
print("Taille de l'ensemble d'entraînement :", len(X_train))
print("Taille de l'ensemble de test :", len(X_test))
```

✓ 0.0s

```
Taille de l'ensemble d'entraînement : 133052
Taille de l'ensemble de test : 33263
```

2.4 Fitting “StandardScaler”

- **StandardScaler** : Harmoniser les échelles des différentes variables numériques. Cela permet d’éviter les distorsions dues aux différences d’échelles.

    - Standardiser *X_train* et *X_test* en transformant tous les features sauf SEX X_train[columns]=ss.fit_transform(X_train[columns]): X_test[columns]=ss.transform(X_test[columns]): Applique uniquement les transformations calculées par **fit_transform** sur l'ensemble d'entraînement.

- Transformer les label en int ({ False, True } -> { 0 , 1 }) : **y.astype(int)**

3.Model Checking : Validation croisée

Nous allons utiliser 4 modèles:

- **SVC (Support Vector Classifier)** : Un classificateur basé sur les machines à vecteurs de support, qui sépare les classes en maximisant la marge.

- **AdaBoostClassifier** : Une méthode d'ensemble learning qui combine plusieurs classificateurs faibles, en ajustant chaque modèle successif pour corriger les erreurs.

- **GradientBoostingClassifier** : Un modèle d'ensemble learning où chaque nouvel arbre réduit les erreurs des modèles précédents.

- **RandomForestClassifier** : Un ensemble d'arbres de décision, où les prédictions sont combinées pour améliorer la performance et réduire le surapprentissage. (Bagging)

- **Stacking Model “Meta-Model”: LinearRegressionClassifier** qui repose sur les prédictions des **3** derniers modèles énoncé ci-dessus

3.1 Test sans choix de hyparamètres

- Obtention de pourcentage d’accuracy et d’écart-type

```
SVM: Mean Accuracy: 0.8032197883883058, Standard Deviation: 0.0014841515828304963
Adaboost: Mean Accuracy: 0.8088040667644554, Standard Deviation: 0.00140749773100127
GradientBoosting: Mean Accuracy: 0.8150722943050044, Standard Deviation: 0.001136220
RandomForest: Mean Accuracy: 0.8132008397292594, Standard Deviation: 0.0012099582322
```

- Evaluation suivant differents metriques (accuracy_score, confusion_matrix, classification_report)

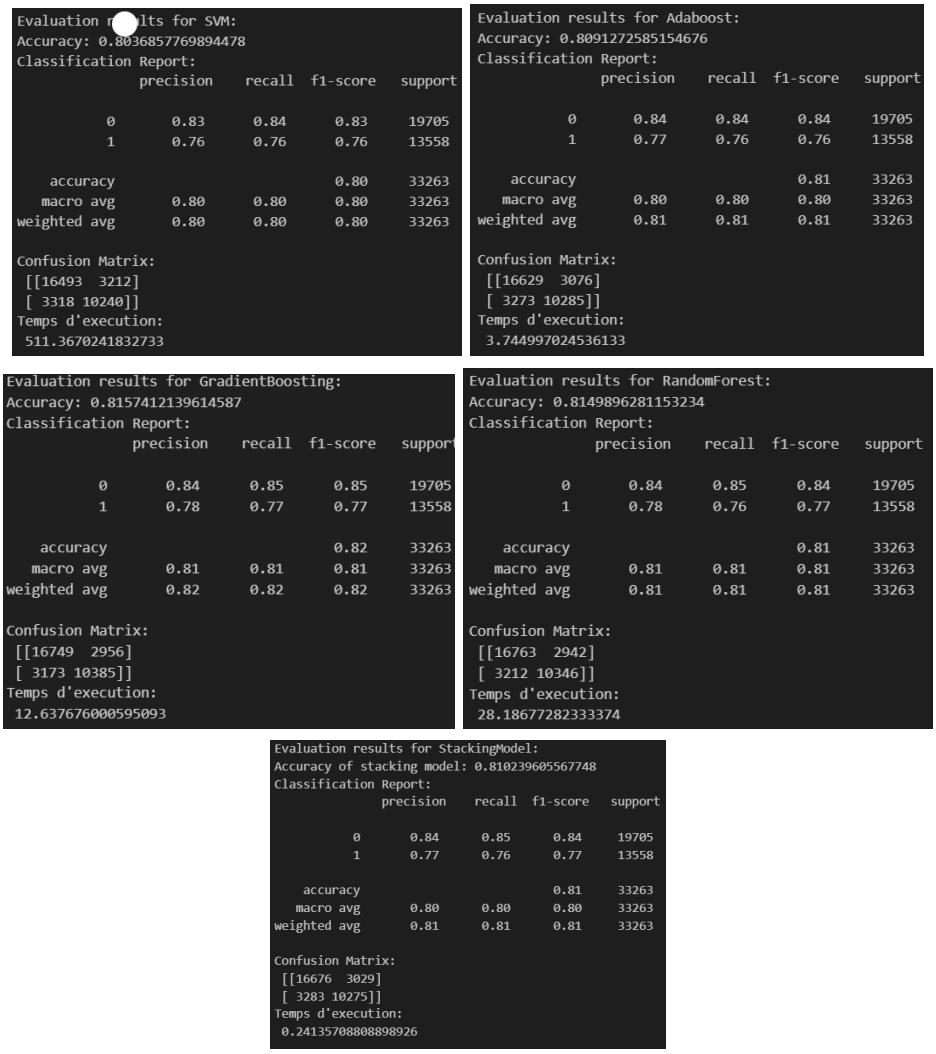


Figure 2 : Résultats des métriques d'évaluation des différents modèles

Modèle	Précision sur 10% du dataset	Précision sur tout le dataset	Ecart-type	Temps d'exec
SVM	79.56%	80.37%	0.15%	511.38s
Adaboost	80.66%	80.91%	0.14%	3.74s
GradientBoost	81.05%	81.57%	0.11%	12.63s
RandomForest	80.02%	81.50%	0.12%	28.18s

Table 1 : Résultats des scores de la validation croisée des différents modèles sans hyperparamètres

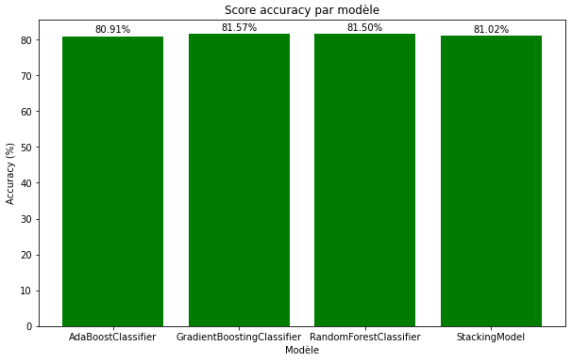


Figure 3 : Score de précision pour chaque modèle

- **SVM**
  - Accuracy : **80.37%**. Cette précision est modérément basse.
  - Écart type : **0.15%**, indiquant une stabilité relative des performances sur différents ensembles de données.
  - faux positifs (**3212**) et faux négatifs **comparable** (**3318)** sachant que classe “0” est **majoritaire**
  - Temps d'exécution : **511.38** secondes, le plus **lent** parmi les modèles.
- **Adaboost**
  - Accuracy : **80.91**%. Cette précision est correcte.
  - Écart type : **0.14%**, reflétant une stabilité notable des performances sur différents tests.
  - Moins de faux positifs (**3076**) et faux négatifs (**3273**), indiquant un meilleur **équilibre** dans la détection des deux classes.
  - Temps d'exécution : **3.74** secondes, le plus rapide parmi les modèles.
- **GradientBoost**
  - Accuracy : **81.57**%. Le modèle offre la précision la plus élevée, bien adaptée aux cas où une meilleure performance est requise.
  - Écart type : **0.11**%, montrant une variabilité légère mais mais meilleure parmi toutes les performances.
  - Moins de faux négatifs (**3173**) que **Adaboost** , ce qui indique une meilleure capacité à détecter les deux classes.
  - Temps d'exécution : **12.63 s**, un compromis entre **Adaboost** et **Random Forest**
- **RandomForest**
  - Accuracy : **81.50**%. Cette précision est pareille que celle de Gradient Boosting et reste élevée.
  - Écart type : **0.12**%, montrant une stabilité satisfaisante, bien que plus variable que Gradient Boosting.
  - Plus de faux négatifs (**3212**) mais moins de faux positifs (**2942**), indiquant une détection meilleure mais perfectible des cas négatifs de la classe **majoritaire** “0”.
  - Temps d'exécution : **28.18** secondes, légèrement supérieur à Gradient Boosting.

- **Stacking Meta-Model:**
    - Accuracy : **81.02**%. Cette précision est pareille que celle de **Adaboost** et reste élevée.
    - Plus de faux négatifs (**3283**) mais moins de faux positifs (**3029**) par rapport à **Adaboost**, indiquant une détection meilleure mais perfectible des cas négatifs de la classe **minoritaire** “1”.

3.2 Amelioration des hyperparameters

- Méthode GridSearchcv() pour identification des meilleurs paramètres

```
param_grid = {
    'SVM':{
        'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
        'kernel': ['rbf', 'sigmoid'],
        'gamma': ['scale', 'auto']
    },
    'Adaboost':{
        'n_estimators':[50,100,150,200],
        'learning_rate':[0.1,0.2,0.5]
    },
    'GradientBoosting':{
        'n_estimators':[100,200],
        'learning_rate':[0.1,0.2,0.5],
        'max_depth':[1,2,5]
    },
    'RandomForest': {
        'n_estimators': [50, 100 , 200],
        'criterion': ['log_loss', 'gini'],
        'max_depth': [1,2,5,10],
        'min_samples_split':[2,5,10],
        'max_features':['sqrt','log2']
    }
}
```

Code 1: Hyperparamètres de Test

- Résultat:

Modèle	Hyperparamètres optimaux
SVM	*C*: 1, *kernel*: ‘rbf”, *gamma*: ‘scale’
Adaboost	*n_estimators*: 200, *learning_rate*: 0.5
GradientBoost	*n_estimators*: 200, *learning_rate*: 0.2, *max_depth*: 5
RandomForest	*n_estimators*: 200, *criterion*: ’gini’, *max_depth*: 10, *min_samples_split*: 2, *max_features*: ‘log2’

Table 2 : Hyperparamètres optimaux

3.3 Test avec hyperparamètre optimaux

- Comparaison entre les évaluations

```
SVM: Mean Accuracy: 0.7727750605374768, Standard Deviation: 0.0026353069651131288
Adaboost: Mean Accuracy: 0.8098318230751627, Standard Deviation: 0.00395357900375865
GradientBoosting: Mean Accuracy: 0.8141913125542153, Standard Deviation: 0.004324042
RandomForest: Mean Accuracy: 0.8050214600763465, Standard Deviation: 0.0048975661052
```

```
Evaluation results for Adaboost:
Accuracy: 0.8132158855184439
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.85      0.84     19705
           1       0.78      0.76      0.77     13558

    accuracy                           0.81     33263
   macro avg       0.81      0.81      0.81     33263
weighted avg       0.81      0.81      0.81     33263

Confusion Matrix:
 [[16703  3002]
 [ 3211 10347]]
Temps d'execution:
 15.199462175369263
```

```
Evaluation results for GradientBoosting:
Accuracy: 0.8278567778011604
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.85      0.85     19705
           1       0.79      0.79      0.79     13558

    accuracy                           0.83     33263
   macro avg       0.82      0.82      0.82     33263
weighted avg       0.83      0.83      0.83     33263

Confusion Matrix:
 [[16836  2869]
 [ 2857 10701]]
Temps d'execution:
 41.95641350746155
```

```
Evaluation results for RandomForest:
Accuracy: 0.8149295012476325
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.85      0.85     19705
           1       0.78      0.76      0.77     13558

    accuracy                           0.81     33263
   macro avg       0.81      0.81      0.81     33263
weighted avg       0.81      0.81      0.81     33263

Confusion Matrix:
 [[16784  2921]
 [ 3235 10323]]
Temps d'execution:
 22.169021368026733
```

```
Evaluation results for StackingModel:
Accuracy of stacking model: 0.8274659531611701
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.86      0.85     19705
           1       0.79      0.78      0.79     13558

    accuracy                           0.83     33263
   macro avg       0.82      0.82      0.82     33263
weighted avg       0.83      0.83      0.83     33263

Confusion Matrix:
 [[16916  2789]
 [ 2950 10608]]
Temps d'execution:
 0.19603753089904785
```

Figure 4 : Résultats des métriques d'évaluation des différents modèles

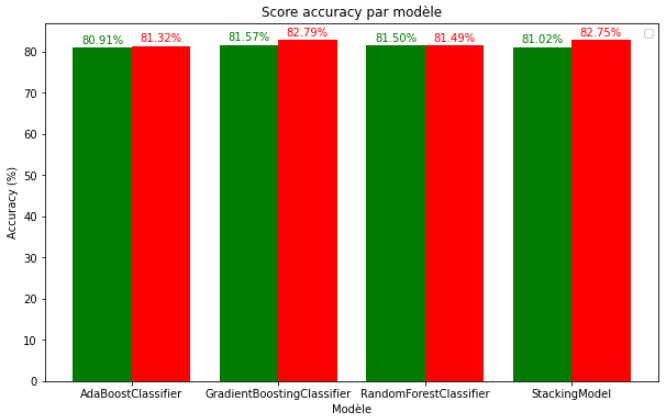


Figure 5 : Score de précision pour chaque modèle avec Hyperparamètres optimaux

Tous les modèles montrent une amélioration notable avec l'optimisation des hyperparamètres

- **SVM**

    - L'utilisation du **SVM** entraînait des temps d'exécution prolongés et produisait des résultats erronés, ce qui a conduit à la décision de ne plus l'utiliser.

- **Adaboost**

    - Accuracy : **81.32**%. La précision a augmenté par rapport à l'ancien **80.91**%, boostant légèrement sa fiabilité comme modèle performant.
    - Écart type : **0.1%**, plus stable par rapport aux résultats précédents.
    - Moins de faux positifs et de faux négatifs (**3211** et **3273** FN) montrant une **légère amélioration** dans la classification des deux classes.
    - Temps d'exécution : **15.19** secondes, plus lent que l'ancien de 14 secondes.

- **GradientBoost**

    - Accuracy : **82.79**%. Une augmentation de **1%** par rapport à **81.57**%, confirmant la capacité du modèle à s'adapter efficacement.
    - Écart type : **0.22**%, moins stable cela est surement dû à un sur-apprentissage sur la classe majoritaire ”0”.
    - Réduction notable des faux négatifs (**2857** contre **3173**) , **améliorant** encore la performance de la détection de la classe minoritaire ‘1’ , et amélioration légère dans la détection des faux négatifs
    - Temps d'exécution : **41.95** secondes, augmenté par rapport à **13.63** secondes.

- **RandomForest**

    - Accuracy : **81.61**%. Similaire à AdaBoost, résultat de **81.45**%, montrant une performance fiable.
    - Écart type : **0.12**%, moins stable par rapport aux précédents.
    - Une **légère amélioration** dans la classification des deux classes.
    - Temps d'exécution : **33.13** secondes, comparable aux anciens résultats.

- **Stacking Meta-Model**

    - Accuracy: **82.75%.** Similaire à **GradientBoost** avec une augmentation de **1.7**%
    - Meilleure détection de la classe **minoritaire** “1” avec pourcentage de **f1-score +2%**

3.4 Test avec différents datasets

Pour vérifier la qualité de nos modèles et de leur entraînement, il nous faut tester les prédictions sur de nouveaux datasets. Les datasets Colorado et Nevada ont la même forme que le dataset initial Californie. Ce qui est important pour que la comparaison ne soit pas entravée et que les résultats soient utiles à l'analyse de notre entrainement.

Test sur Nevada

```
Evaluation results for AdaBoostClassifier:
Accuracy: 0.7351877607788595
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.69      0.78      7417
           1       0.55      0.83      0.66      3368

    accuracy                           0.74     10785
   macro avg       0.73      0.76      0.72     10785
weighted avg       0.79      0.74      0.74     10785

Confusion Matrix:
 [[5122 2295]
 [ 561 2807]]
```

```
Evaluation results for GradientBoostingClassifier:
Accuracy: 0.7499304589707928
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.70      0.79      7417
           1       0.57      0.85      0.68      3368

    accuracy                           0.75     10785
   macro avg       0.74      0.78      0.74     10785
weighted avg       0.81      0.75      0.76     10785

Confusion Matrix:
 [[5209 2208]
 [ 489 2879]]
```

```
Evaluation results for RandomForestClassifier:
Accuracy: 0.7375985164580435
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.69      0.78      7417
           1       0.55      0.84      0.67      3368

    accuracy                           0.74     10785
   macro avg       0.73      0.77      0.73     10785
weighted avg       0.79      0.74      0.75     10785

Confusion Matrix:
 [[5126 2291]
 [ 539 2829]]
```

```
Evaluation results for StackingModel:
Accuracy of stacking model: 0.7518776077885952
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.71      0.80      7417
           1       0.57      0.85      0.68      3368

    accuracy                           0.75     10785
   macro avg       0.74      0.78      0.74     10785
weighted avg       0.80      0.75      0.76     10785

Confusion Matrix:
 [[5248 2169]
 [ 507 2861]]
```

Figure 7 : Résultats des métriques d'évaluation des différents modèles pour la dataset de Nevada (1 :SVM, 2 :Adaboost, 3 :GradientBoost, 4 :RandomForest)

Test sur Colorado

```
Evaluation results for AdaBoostClassifier:
Accuracy: 0.7780617134095701
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.73      0.79     18334
           1       0.69      0.84      0.76     12972

    accuracy                           0.78     31306
   macro avg       0.78      0.79      0.78     31306
weighted avg       0.79      0.78      0.78     31306

Confusion Matrix:
 [[13428  4906]
 [ 2042 10930]]
```

```
Evaluation results for StackingModel:
Accuracy of stacking model: 0.7912860154602952
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.74      0.81     18334
           1       0.70      0.86      0.77     12972

    accuracy                           0.79     31306
   macro avg       0.79      0.80      0.79     31306
weighted avg       0.81      0.79      0.79     31306

Confusion Matrix:
 [[13654  4680]
 [ 1854 11118]]
```

Figure 6 : Résultats pertinents des différents modèles pour la dataset de Colorado

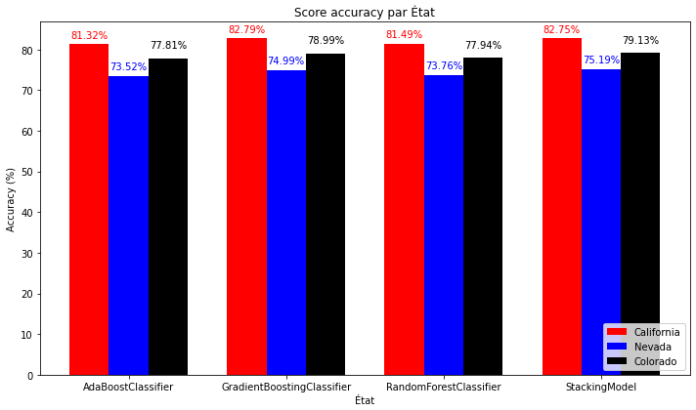


Figure 8 : Score de précision pour différents états

- **Adaboost**
  - Pour **Nevada,** plus biaisé vers la classe majoritaire "0" avec **f1-score 75%** pour cette classe contre seulement **66%** pour la classe “1”.
- **GradientBoost**
  - Semblable aux résultats précédents, avec des biais marqués à détecter la classe "0" (**2208** prédictions fausses) plus performant sur cette classe que **Adaboost** et reste le plus fiable entre tous les modèles.
- **RandomForest**
  - Semblable aux résultats précédents, avec des biais marqués à détecter la classe "0" (**67% f1-score** pour la classe “1”)
- **Stacking Meta-Model:**
  - La matrice montre que le modèle est biaisé vers la classe majoritaire "0".
  - Meilleure performance sur le dataset **Colorado,** avec un équilibre entre la détection des deux classes (**f1-score 81%** pour “0”**,** et **77%** pour “1”**)**

Tous les modèles souffrent de **biais liés au déséquilibre des classes** :

- Tous les modèles **SVM, Adaboost**, **Gradient Boosting** et **Random Forest** favorisent la classe majoritaire "0".

Explication:

- **Déséquilibre des classes:** Le déséquilibre des classes est très marqué dans le sous-ensemble "Nevada", comme le montre la matrice de confusion où la classe 0 a un nombre beaucoup plus élevé d'exemples que la classe 1
- **Choix des hyperparamètres :** Des hyperparamètres comme la profondeur des arbres dans **RandomForest** ou le taux d'apprentissage dans **GradientBoosting** peuvent aussi affecter les résultats
- **Overfitting:** Comme les modèles ont été ajustés sur le dataset de "California" et ne sont pas capables de bien généraliser au dataset "Nevada", ils peuvent montrer de bonnes performances sur un sous-ensemble similaire (comme "Colorado") mais échouer sur un autre (comme "Nevada")

4.Interprétabilité

4.1 Corrélation entre les features et label

- **Le premier élément** représente le coefficient de corrélation: Une corrélation **positive** signifie que lorsque la valeur d'une caractéristique **augmente**, la valeur du label tend également à **augmenter**. En revanche, une corrélation **négative** indique que lorsque la valeur d'une caractéristique **augmente**, la valeur du label a tendance à **diminuer**.
- **Le deuxième élément** est la p-valeur, qui mesure la significativité statistique. Une p-valeur proche de **0** indique une forte probabilité que la **corrélation** soit **significative**.

```
AGEP : PearsonRResult(statistic=0.2656844999163617, pvalue=0.0)
COW : PearsonRResult(statistic=0.04987436872025718, pvalue=8.9106863892193e-20)
SCHL : PearsonRResult(statistic=0.35084179185855735, pvalue=0.0)
MAR : PearsonRResult(statistic=-0.2696529916497584, pvalue=0.0)
OCCP : PearsonRResult(statistic=-0.3443212012548947, pvalue=0.0)
POBP : PearsonRResult(statistic=-0.09457974505044879, pvalue=5.8222588110188704e-67)
RELP : PearsonRResult(statistic=-0.22771882839052798, pvalue=0.0)
WKHP : PearsonRResult(statistic=0.33825275412297917, pvalue=0.0)
SEX : PearsonRResult(statistic=-0.11198033201351397, pvalue=2.821746882290791e-93)
RAC1P : PearsonRResult(statistic=-0.10123867920667112, pvalue=1.6853798915183768e-76)
```

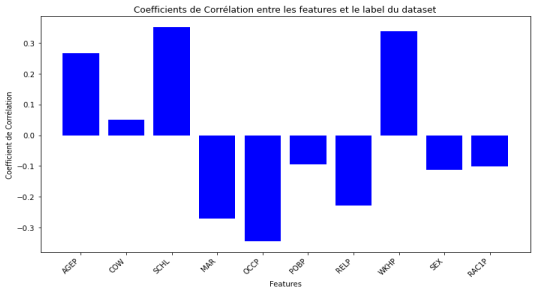


Figure 9 : Corrélation entre les features et les labels

Corrélations fortes :

- SCHL (r=0.35) : forte corrélation positive, p = 0 significative.
- WKHP (r=0.34) : forte corrélation positive, p = 0 significative.
- OCCP (r=−0.34) : forte corrélation négative, p = 0 significative.

Corrélations modérées :

- AGEP (r=0.27) : corrélation positive modérée.
- MAR (r=−0.27) : corrélation négative modérée.
- RELP (r=−0.23) : corrélation négative modérée.

Corrélations faibles mais significatives :

- COW (r=0.049) : très faible corrélation positive, mais significative vu la p-valeur.
- RAC1P, SEX et POBP : faibles corrélations, toutes significatives

Certaines caractéristiques, comme l'âge, le niveau d'études et le temps de travail, ont une **influence positive** sur le revenu, ce qui semble cohérent. En revanche, d'autres caractéristiques, telles que les relations au sein du foyer, le statut marital et le métier occupé, exercent une **influence négative**. On peut supposer, pour la variable OCCP, que plus sa valeur augmente, moins le métier est associé à un revenu élevé.

4.2 Corrélation entre les features et les prédictions

Nous avons aussi calculé les indices de corrélations des features avec les prédictions des différents modèles avec les Hyperparamètres optimaux pour voir si les paternes trouvées étaient similaires à ceux observés sur le dataset d'entraînement

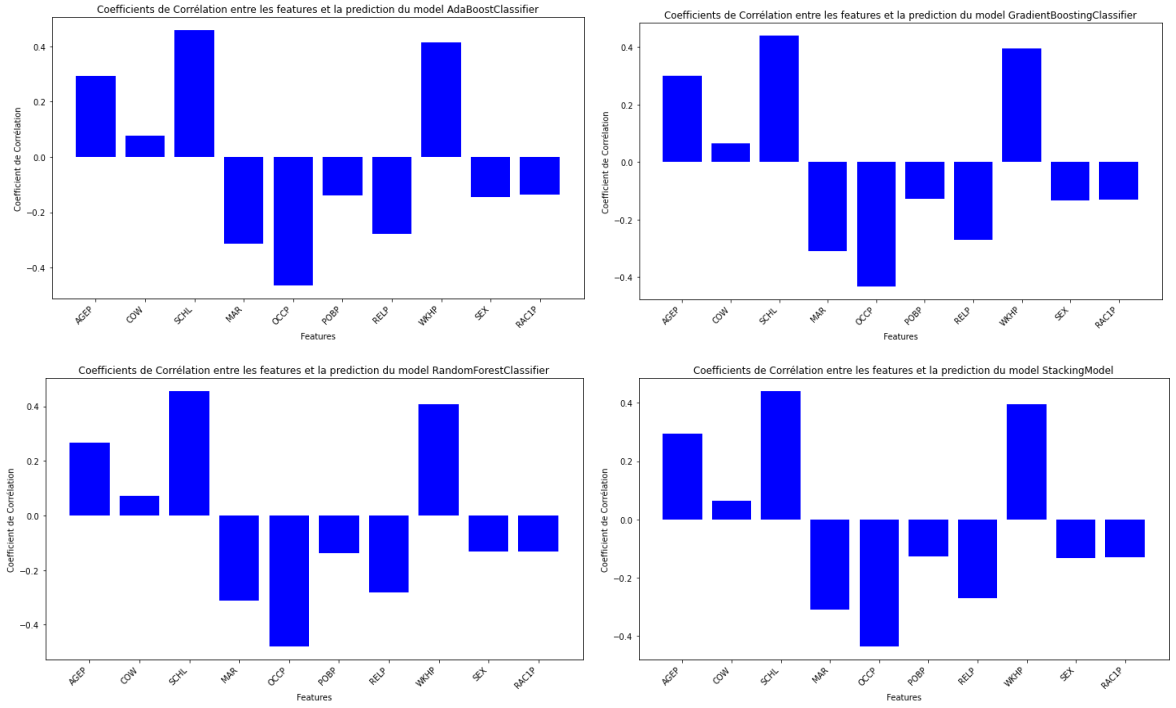
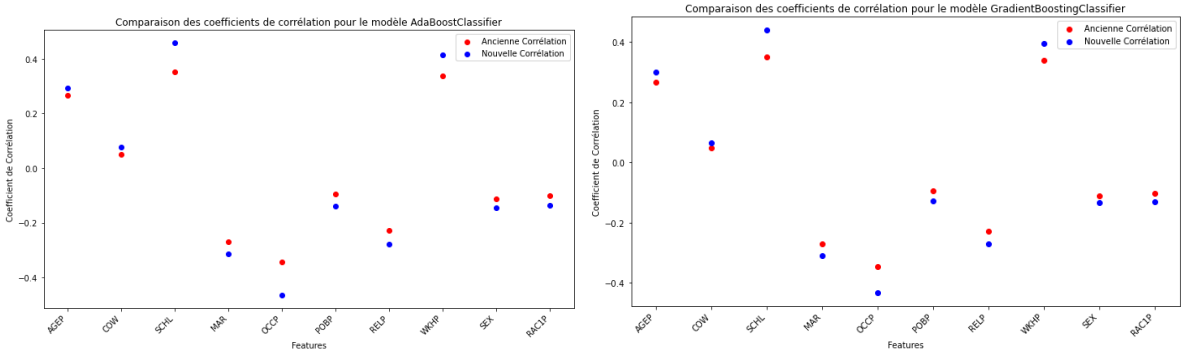


Figure 10: Corrélation entre les features et les prédictions des différents modèles

Les 4 modèles sont quasiment identiques entre eux et très proches de la corrélation avec les labels. Cela se voit aussi avec les scores d'accuracy qui sont très bons sur ces modèles

- Comparer entre les 2



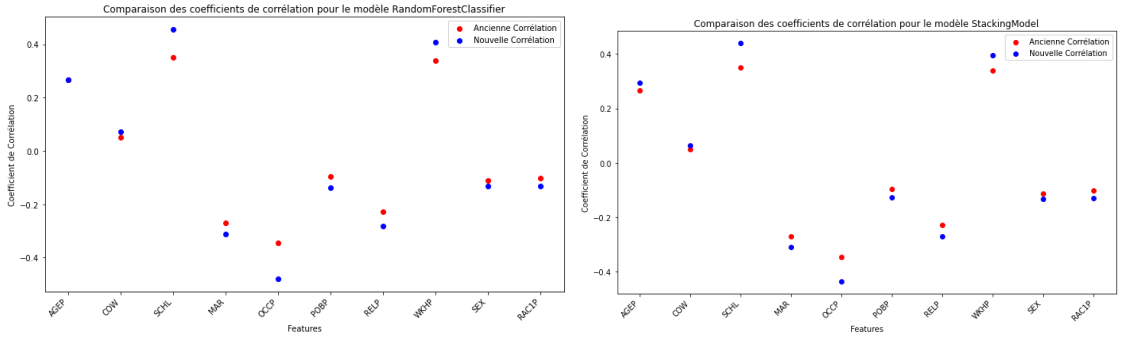


Figure 11: Différences de Corrélation (1 :Adaboost, 2 :GradientBoost, 3 :RandomForest, 4: StackingModel)

- **Adaboost** semble **plus sensible** aux features, en attribuant un poids plus élevé, par rapport aux autres modèles, à certaines variables **faiblement corrélées initialement.** **(vu en 1.4)**
- **Gradient Boost équilibre** bien les corrélations et tire profit des variables importantes.
- **Random Forest** reste **conservateur** et **robuste** face aux relations initiales entre les features et la cible.

4.3 Evaluation importance des features:permutation_importance()

Cette méthode consiste à mesurer l'impact de chaque caractéristique sur la performance du modèle

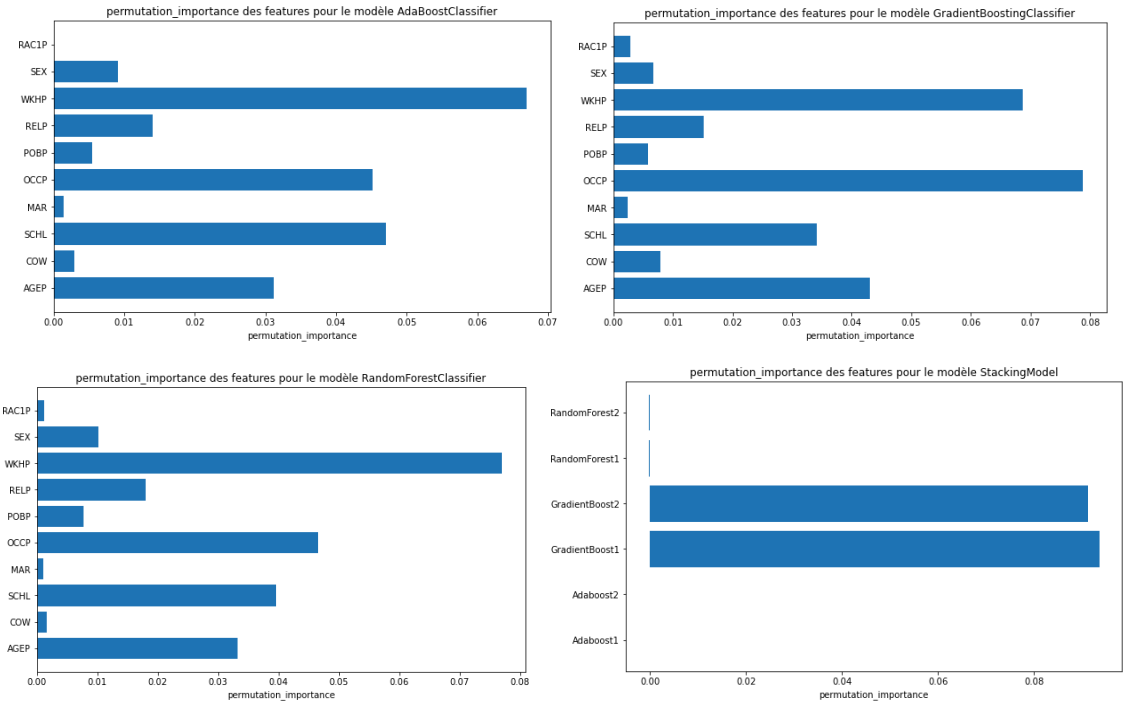


Figure 12: Importance moyenne des features pour chacun des modèles (1 :SVM, 2 :Adaboost, 3 :GradientBoost, 4 :RandomForest)

- Tous les modèles mettent l'emphase sur la feature OCCP et WKHP et SCHL, tout en ayant d'autres similarités sur les autres features.
- De même les similitudes avec les modèles se remarquent aussi grâce à cette métrique puisque nous pouvons observer que, mis à part quelques différences mineures, les valeurs d'importance de chaque feature sont quasiment identiques pour ces 3 modèles.
- On remarque aussi que **GradientBoosting** est énormément influencé par OCCP.
- Le **Stacking Meta-Model** manuellement implémenté dépend surtout des prédictions du modèle le plus performant **GradientBoosting.**

4.4 Bonus : Analyse par Shap

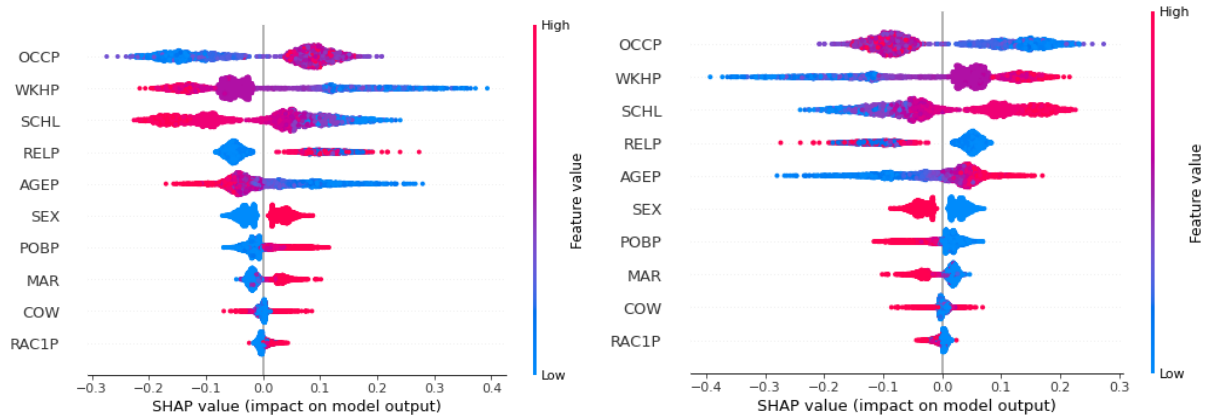


Figure 13 :Shap : Impact des features sur les prédictions d'un modèle RandomForest (1: label '0', 2: label '1')

➔ SEX semble jouer un rôle relativement **modéré** dans la prédiction comparée aux autres variables comme OCCP et WKHP, mais **son impact reste visible.**

➔ La variable SEX reste utilisée par le modèle pour prendre des décisions, ce qui est problématique si l'objectif est d'atteindre **l'équité**.

5. Equité des modèles

5.1 Evaluation de taux d'individus ayant un revenu supérieur à 50k dollars par SEX

- Taux général sur X_train:

```
len(dataset1[dataset1["PINCP"]==True])/len(X_train)
✓ 0.1s
0.4107040856206596
```

- Taux par homme:

```
Homme=dataset[dataset["SEX"]==1]
Homme_T=Homme[Homme["PINCP"]==True]
```

```
len(Homme_T)/len(Homme)
0.0s
0.4667719942496833
```

- Taux par femme:

```
len(Femme_T)/len(Femme)
0.0s
0.34797356477426544
```

5.2 Matrice de confusion de SEX

- Métrique d'équité pour tous les modèles optimaux(True Positive,TN, FP, False Negative)

```
TP :
{'1.0_SVM': 623, '2.0_SVM': 409, '1.0_Adaboost': 611, '2.0_Adaboost': 398, '1.0_G
TN :
{'1.0_SVM': 745, '2.0_SVM': 887, '1.0_Adaboost': 742, '2.0_Adaboost': 929, '1.0_G
FP :
{'1.0_SVM': 170, '2.0_SVM': 164, '1.0_Adaboost': 173, '2.0_Adaboost': 122, '1.0_G
FN :
{'1.0_SVM': 178, '2.0_SVM': 151, '1.0_Adaboost': 190, '2.0_Adaboost': 162, '1.0_G
```

- Calcul de sensibilite et specificite

Modéle	Sexe	Sensibilité : Taux de TP	Spécificité Taux de TN
StackingModel	Homme	0.81	0.83
	Femme	0.74	0.89
Adaboost	Homme	0.80	0.81
	Femme	0.71	0.88
GradientBoosting	Homme	0.81	0.82
	Femme	0.75	0.88
RandomForest	Homme	0.79	0.82
	Femme	0.72	0.88

Table 3 : Résultats des métriques d'équité statistique des différents modèles en considérant 'SEX' comme la feature sensible

Stacking Meta-Model

- **Différence : Sensibilité (7%) Spécificité (6.25%)**
  - Détecte mieux les cas positifs chez les hommes **(+7%)**, mais il est plus performant dans l'évitement des faux positifs chez les femmes **(+6%)**.
  - Cela montre une légère tendance à privilégier les femmes en termes de spécificité, mais il est globalement équilibré.

Adaboost

- **Différence : Sensibilité (9.02%) Spécificité (6.7%)**
  - Détecte mieux les cas positifs chez les hommes **(+9%)**, et il est plus performant dans l'évitement des faux positifs chez les femmes **(+7%)**.
  - Cela montre une tendance à privilégier les femmes.

GradientBoost

- **Différence : Sensibilité (6%) Spécificité (5.82%)**
  - Plus sensible aux cas positifs pour les hommes **(+6%**), mais il est plus performant pour éviter les faux positifs chez les femmes **(+6%)**.
  - Les performances sont assez équilibrées, mais avec une légère préférence pour les femmes en termes de détection de cas négatifs.

RandomForest

- **Différence : Sensibilité (7.2%) Spécificité (5.4%)**
  - Détecte mieux les cas positifs chez les hommes **(+7%)**, mais évite mieux les faux positifs chez les femmes **(+5%).**
  - Cela montre un équilibre similaire à **Adaboost**, mais avec des résultats globalement moins marqués.

⇒ Les performances varient systématiquement entre les genres. Cela indique que les modèles utilisent SEX comme une information discriminante, ce qui pourrait refléter ou exacerber des biais dans les données d'origine.

5.3 Test sans la feature SEX

- **Enlever** la colonne “SEX” , **réentraîner** les modèles et **repérer** les lignes de X_test , y_test ainsi que y_pred qui correspondent au SEX

```
y=dataset["PINCP"]
X=dataset.drop(columns="PINCP")
X_train, X_test, y_train, y_test,sex_train,sex_test =train_test_split(
    X, y,features['SEX'], test_size=0.2, random_state=42)
```

- Masque pour filtrer les données selon la colonne sex_test, permettant de calculer la matrice de confusion spécifiquement pour un groupe particulier (homme ,femme)

```
mask = (sex_test == i)
conf_matrix = confusion_matrix(y_test[mask], y_pred[mask])
```

Model_Feature	Sensibilité : True Positive Rate (TPR)	Spécificité :True Negative Rate (TNR)
1.0_AdaBoostClassifier	0.760782	0.846541
2.0_AdaBoostClassifier	0.759581	0.841522
1.0_GradientBoostingClassifier	0.783252	0.855939
2.0_GradientBoostingClassifier	0.791021	0.854631
1.0_RandomForestClassifier	0.763015	0.845863
2.0_RandomForestClassifier	0.765525	0.846318
1.0_StackingModel	0.776971	0.860880
2.0_StackingModel	0.783826	0.858254

Table 4: Résultats des métriques d'équité statistique des différents modèles en considérant' SEX' comme la feature sensible

- **Stacking Meta-Model**
  - Homme : **TPR (-3.3%)** **TNR(+3.1%)**
  - Femme : **TPR (+4.1%)** **TNR(-3.4%)**
  - Les valeurs sont proches et se centralisent autour de **78%** pour la **Sensibilité** et **85.9%** pour la **Spécificité.**
- **Adaboost**
  - Homme : **TPR (-3.9%) , TNR(+3.2%)**
  - Femme : **TPR (+5%) , TNR(-3.8%)**
  - La sensibilité (TPR) est légèrement réduite pour les hommes **(-4%)** et améliorée pour les femmes **(+5%) ,** elle est **centralisée** autour de **76%** .
  - La spécificité (TNR) s'est **uniformisée** autour de **84.3%** pour les deux classes, ce qui montre une **perte de différenciation** entre les sexes.
- **GradientBoosting**
  - Homme : **TPR (-3.1%) , TNR(+3.1%)**
  - Femme : **TPR (+3.92%) , TNR(-2.8%)**
  - La sensibilité (TPR) et la spécificité (TNR) sont devenues très similaires entre les deux classes.
- **RandomForest**
  - Homme : **TPR (-2.7%) , TNR(+2.2%)**
  - Femme : **TPR (+4.7%) , TNR(-3.1%)**
  - La sensibilité (TPR) et la spécificité (TNR) sont devenues exactement pareil entre les deux classes.

➔ La suppression de la colonne SEX a permis de **réduire** considérablement les différences de performance entre les genres, suggérant que les biais observés auparavant étaient en **grande partie** dus à l'inclusion explicite de cette information, et donc **améliore** significativement l'équité des modèles.

➔ Le **biais persiste** partiellement. Cela s'explique par : La présence de variables **corrélées** avec "SEX" (ex : métier: OCCP, WKHP), qui permettent au modèle d'inférer indirectement cette information.

Conclusion

En conclusion, ces travaux pratiques nous ont permis d'approfondir notre compréhension de l'apprentissage supervisé et de ses enjeux. Nous avons appris à diviser un jeu de données en sous-ensembles d'entraînement et de test, ainsi qu'à évaluer les performances des modèles à l'aide de différentes métriques.

En utilisant le dataset ACSIncome, nous avons également exploré d'autres aspects de l'apprentissage supervisé, tout en menant des analyses critiques sur les données et en identifiant les biais potentiels auxquels les modèles peuvent être confrontés lors de leur entraînement.

Finalement, cette étude nous a permis de mettre en évidence des **problématiques sociétales actuelles**, comme les disparités de genre, et d'observer comment ces problématiques se reflètent dans les résultats des modèles.