Analyse descriptive et predictive
Apprentissage supervisé

TP Apprentissage supervisé

Étudiant :
Ralaimady SETA

Enseignants :
Marie-José HUGUET

Table des matières

1 Introduction **2**

2 Préparation de la dataset **3**

3 Application des différents algorithmes d'apprentissage **3**
3.1 SVM . . . 3
3.2 RandomForest . . . 3
3.3 AdaBoost . . . 4
3.4 GradientBoosting . . . 4
3.5 Comparaison des modèles . . . 4

4 Application des modèles entraînés sur d'autres jeux de données **5**
4.1 Jeu de données du Nevada . . . 5
4.2 Jeu de données du Colorado . . . 6
4.3 Comparaison avec des résultats avec les différents jeux de données . . . 6

5 Explicabilité des modèles **7**
5.1 Coefficient de corrélation . . . 7
5.2 Permutation$_i$*mportance* . . . 7

6 Équité des modèles **8**
6.1 Test des modèles en considérant le feature *SEX* comme feature sensible . . . 8
6.1.1 Individus *Homme* retirés du jeu de données . . . 8
6.1.2 Individues *Femme* retirées du jeu de données . . . 8
6.1.3 Feature *SEX* retiré . . . 9

7 Conclusion **10**

1 Introduction

Dans le cadre du cours d'Analyse descriptive et prédictive de ce premier semestre de cinquième année Systèmes Distribués et Big Data, je me suis penché sur plusieurs méthodes d'apprentissage supervisé.

L'idée était d'appréhender les différents algorithmes sur un même jeu de données afin d'analyser et comparer les résultats obtenus. Je voulais également explorer l'explicabilité des modèles en analysant les différentes importances que chaque modèle donne à chaque feature du jeu de données et enfin voir l'impact d'une feature selon ses différentes valeurs sur le résultat de la prédiction.

Mon projet peut se retrouver sur un dépôt git au lien suivant :
https ://github.com/MadySeta/TP-Apprentissage-supervis-.git

2 Préparation de la dataset

Le jeu de données que je vais utiliser pour entraîner mes différents modèles est décrit dans l'article Ding, Frances, et al. "Retiring adult : New datasets for fair machine learning." Advances in neural information processing systems 34 (2021) : 6478-6490.

Le jeu de données étant trop volumineux, je n'ai travaillé que sur 1% du dataset initial.

Tout d'abord, j'ai séparé le jeu de données en 2 ensembles, l'ensemble d'entraînement (80%) et l'ensemble de test (20%). J'ai ensuite standardisé l'ensemble d'entraînement via un *fit_ transform()* de *scikit-learn* pour continuer sur l'ensemble de test via un *transform()*. Il faut savoir que la standardisation se fait de cette manière pour ne pas biaiser les résultats étant donné que les paramètres de la standardisation sont calculés sur l'ensemble d'entraînement uniquement et pas sur les 2 ensembles.

3 Application des différents algorithmes d'apprentissage

Pour chaque modèle, je vais appliquer un GridSearchCV avec une validation croisée (5 folds) afin de trouver de meilleurs hyperparamètres pour mon jeu de données.

Je passe ensuite à l'étape de test, où je fais une prédiction sur une partie du jeu de données non-utilisé lors de l'entraînement, et avec les résultats, je calcule ces trois différentes métriques :

— Accuracy
— Classification report
— Confusion matrix

3.1 SVM

```
Meilleurs hyperparamètres :  {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Accuracy :  0.7576530612244898
 Rapport de classification :
              precision    recall  f1-score   support

       False       0.77      0.82      0.79       220
        True       0.75      0.68      0.71       172

    accuracy                           0.76       392
   macro avg       0.76      0.75      0.75       392
weighted avg       0.76      0.76      0.76       392

Confusion matrix :
[[180  40]
 [ 55 117]]
```

FIGURE 1 – Résultats du modèle SVM

3.2 RandomForest

```
Meilleurs hyperparamètres :  {'criterion': 'gini', 'max_depth': 100, 'min_samples_split': 50, 'n_estimators': 100}
Accuracy :  0.7806122448979592
 Rapport de classification :
              precision    recall  f1-score   support

       False       0.78      0.84      0.81       220
        True       0.78      0.70      0.74       172

    accuracy                           0.78       392
   macro avg       0.78      0.77      0.77       392
weighted avg       0.78      0.78      0.78       392

Confusion matrix :
[[185  35]
 [ 51 121]]
```

FIGURE 2 – Résultats du modèle RandomForest

3.3 AdaBoost

```
Meilleurs hyperparamètres :  {'learning_rate': 0.5, 'n_estimators': 50}
Accuracy :  0.7678571428571429
 Rapport de classification :
              precision    recall  f1-score   support

       False       0.77      0.84      0.80       220
        True       0.76      0.68      0.72       172

    accuracy                           0.77       392
   macro avg       0.77      0.76      0.76       392
weighted avg       0.77      0.77      0.77       392

Confusion matrix :
[[184  36]
 [ 55 117]]
```

FIGURE 3 – Résultats du modèle AdaBoost

3.4 GradientBoosting

```
Meilleurs hyperparamètres :  {'criterion': 'friedman_mse', 'learning_rate': 0.01, 'loss': 'deviance', 'n_estimators': 500}
Accuracy :  0.7806122448979592
 Rapport de classification :
              precision    recall  f1-score   support

       False       0.79      0.84      0.81       220
        True       0.77      0.71      0.74       172

    accuracy                           0.78       392
   macro avg       0.78      0.77      0.77       392
weighted avg       0.78      0.78      0.78       392

Confusion matrix :
[[184  36]
 [ 50 122]]
```

FIGURE 4 – Résultats du modèle GradientBoosting

3.5 Comparaison des modèles

Les différents modèles ont des performances plus ou moins similaires sur le jeu de données que j'utilise. On remarque cependant que les modèle d'apprentissage *RandomForest* et *GrandientBoosting* ont légèment de meilleurs résultats que les deux autres modèles *SVM* et *AdaBoost*.

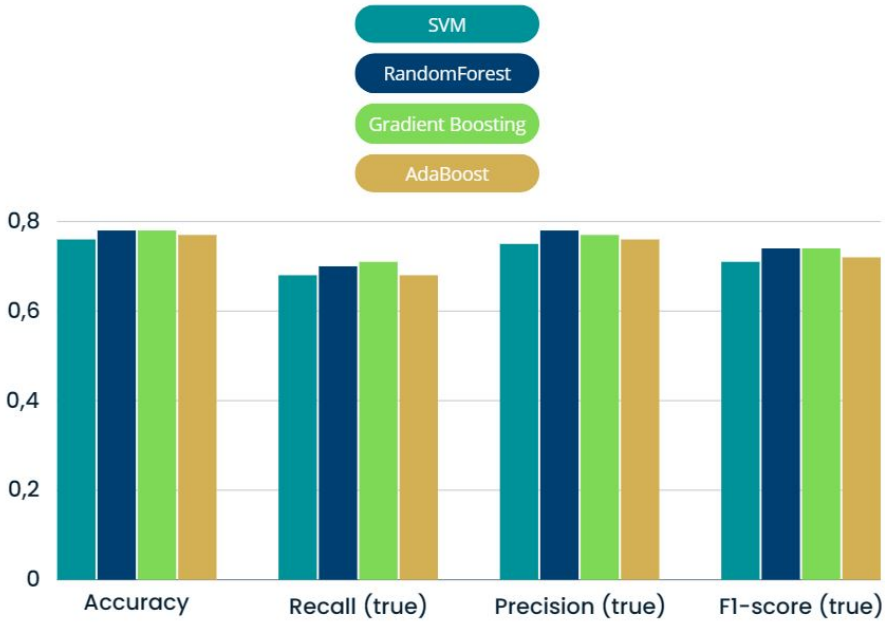

FIGURE 5 – Comparaison des "Accuracy" des 4 modèles sur les jeux de données de la Californie, du Nevada et du Colorado

4 Application des modèles entraînés sur d'autres jeux de données

Afin d'analyser les performances des différents modèles, je vais appliquer les modèles sur deux jeux de données qui n'ont pas servi à l'entraînement et au test de nos 4 modèles. Les jeux de données correspondent aux jeux de données du Nevada et du Colorado.

4.1 Jeu de données du Nevada

```
SVM Accuracy :  0.7289719626168224
RDF Accuracy :  0.7383177570093458
ADB Accuracy :  0.7570093457943925
GDB Accuracy :  0.7476635514018691
SVM Confusion matrix :
[[184  36]
 [ 50 122]]
RDF Confusion matrix :
[[184  36]
 [ 50 122]]
ADB Confusion matrix :
[[184  36]
 [ 50 122]]
GDB Confusion matrix :
[[184  36]
 [ 50 122]]
```

FIGURE 6 – Résultats des modèles sur le jeu de données du Nevada

4.2 Jeu de données du Colorado

```
SVM Accuracy :  0.7252396166134185
RDF Accuracy :  0.7507987220447284
ADB Accuracy :  0.731629392971246
GDB Accuracy :  0.7412140575079872
SVM Confusion matrix :
[[184  36]
 [ 50 122]]
RDF Confusion matrix :
[[184  36]
 [ 50 122]]
ADB Confusion matrix :
[[184  36]
 [ 50 122]]
GDB Confusion matrix :
[[184  36]
 [ 50 122]]
```

FIGURE 7 – Résultats des modèles sur le jeu de données du Colorado

4.3 Comparaison avec des résultats avec les différents jeux de données

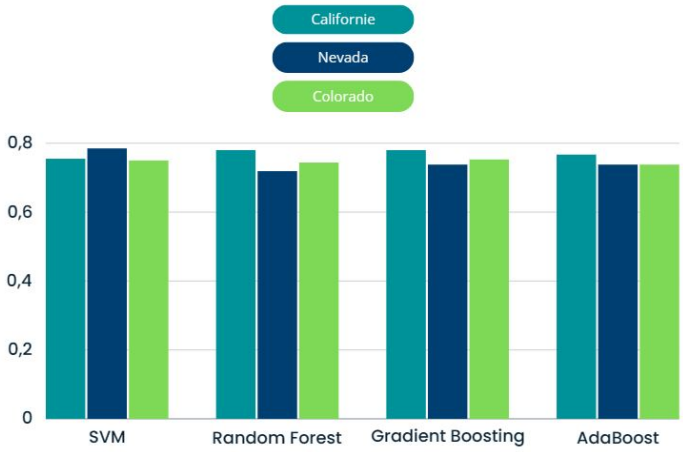

FIGURE 8 – Graphique de comparaison des "accuracy" sur les différents jeux de données et sur les différents modèles

On remarque que pour *RandomForest, GradientBoosting et AdaBoost*, l'*accuracy* sur les jeux de données du Colorado et du Nevada sont légèrement inférieurs à l'*accuracy* sur l'ensemble test de la Californie. Ce qui était prévisible étant donné que nos modèles sont entraînés avec les données de la Califonie. On remarque cependant que pour le modèle *SVM*, l'*accuracy* qu'on obtient avec données du Nevada est meilleur que l'*accuracy* obtenu avec l'ensemble test du jeu de données de la Californie.

5 Explicabilité des modèles

5.1 Coefficient de corrélation

TABLE 1 – Coefficient de corrélation

Feature	Label	Grandient Boosting	AdaBoost	SVM	Random Forest
AGEP	0.264362	0.204070	0.214495	0.207672	0.181922
COW	0.128665	0.042642	0.087596	0.024466	0.038564
SCHL	0.376713	0.471732	0.488184	0.437549	0.460195
MAR	-0.283184	-0.265245	-0.284715	-0.330393	-0.313749
OCCP	-0.329998	-0.589933	-0.602938	-0.563285	-0.565999
POBP	0.008443	-0.119144	-0.123028	-0.140495	-0.146538
RELP	-0.200241	-0.273018	-0.238265	-0.350600	-0.311299
WKHP	0.411818	0.414855	0.395772	0.393914	0.374438
SEX	-0.085003	-0.051640	-0.049350	-0.083008	-0.061084
RAC1P	-0.040897	-0.149184	-0.179422	-0.154656	-0.141276

En ce qui concerne, les coefficients de corrélation entre les différentes features et le label, on peut dire que le feature *WKHP* a une plus grande corrélation (influence) avec le label comparé aux autres features. On a ensuite le features *SCHL et OCCP* qui ont des coefficients de corrélation plus ou moins élevées.

Les coefficients changent lorsqu'on calcule ces derniers sur les labels, cette fois prédits par les différents modèles d'apprentissage. On a par exemple la feature *OCCP* dont le coefficient a beaucoup augmenté par rapport au coefficient calculé avec le label original.

5.2 Permutation$_{i}importance$

TABLE 2 – Permutation importance

Feature	Grandient Boosting	AdaBoost	SVM	Random Forest
AGEP	0.007653	0.008163	-0.005102	0.015306
COW	0.005102	-0.003061	-0.010714	0.005102
SCHL	0.046429	0.026020	0.035204	0.043878
MAR	0.004592	0.007143	0.013776	0.007143
OCCP	0.046429	0.019388	0.021939	0.053061
POBP	0.004082	0.005612	-0.009694	0.009184
RELP	0.011735	0.006122	0.006122	0.015816
WKHP	0.055612	0.046429	0.041327	0.049490
SEX	0.005612	0.002041	0.018878	-0.002551
RAC1P	0.000000	0.000000	0.000510	-0.002551

On remarque, les features *WKHP, SCHL et OCCP* ont les valeurs de permutation les plus importantes sur les différents modèles d'apprentissage, ce qui a été également montré grâce aux calculs des différents coefficients de corrélation.

6 Équité des modèles

6.1 Test des modèles en considérant le feature *SEX* comme feature sensible

Pour chaque modèle et pour chaque cas, j'ai calculé la matrice de confusion ainsi que le rappel (True Positive Rate = True Positive/(True Positive+False Negative) et la précision (True Positive /(True Positive+False Positive)).

6.1.1 Individus *Homme* retirés du jeu de données

```
Confusion matrix on test:
[[215  35]
 [ 36  84]]
Rappel = 0.7
Précision = 0.7058823529411765
-------------------------------------------
Confusion matrix on train:
[[845 107]
 [146 379]]
Rappel = 0.7219047619047619
Précision = 0.779835390946502
```

FIGURE 9 – SVM

```
Confusion matrix on test:
[[222  28]
 [ 39  81]]
Rappel = 0.675
Précision = 0.7431192660550459
-------------------------------------------
Confusion matrix on train:
[[916  36]
 [ 56 469]]
Rappel = 0.8933333333333333
Précision = 0.9287128712871288
```

FIGURE 10 – Random Forest

```
Confusion matrix on test:
[[222  28]
 [ 39  81]]
Rappel = 0.675
Précision = 0.7431192660550459
-------------------------------------------
Confusion matrix on train:
[[916  36]
 [ 56 469]]
Rappel = 0.8933333333333333
Précision = 0.9287128712871288
```

FIGURE 11 – Gradient Boosting

```
Confusion matrix on test:
[[225  25]
 [ 38  82]]
Rappel = 0.6833333333333333
Précision = 0.7663551401869159
-------------------------------------------
Confusion matrix on train:
[[834 118]
 [137 388]]
Rappel = 0.7390476190476191
Précision = 0.766798418972332
```

FIGURE 12 – AdaBoost

6.1.2 Individues *Femme* retirées du jeu de données

```
Confusion matrix on test:
[[180  46]
 [ 40 148]]
Rappel = 0.7872340425531915
Précision = 0.7628865979381443
-------------------------------------------
Confusion matrix on train:
[[777 123]
 [147 605]]
Rappel = 0.8045212765957447
Précision = 0.8310439560439561
```

FIGURE 13 – SVM

```
Confusion matrix on test:
[[186  40]
 [ 40 148]]
Rappel = 0.7872340425531915
Précision = 0.7872340425531915
-------------------------------------------
Confusion matrix on train:
[[846  54]
 [ 58 694]]
Rappel = 0.9228723404255319
Précision = 0.9278074866310161
```

FIGURE 14 – Random Forest

```
Confusion matrix on test:
[[187  39]
 [ 43 145]]
Rappel = 0.7712765957446809
Précision = 0.7880434782608695
-----------------------------------------
Confusion matrix on train:
[[774 126]
 [132 620]]
Rappel = 0.824468085106383
Précision = 0.8310991957104558
```

FIGURE 15 – Gradient Boosting

```
Confusion matrix on test:
[[181  45]
 [ 42 146]]
Rappel = 0.776595744680851
Précision = 0.7643979057591623
-----------------------------------------
Confusion matrix on train:
[[749 151]
 [138 614]]
Rappel = 0.8164893617021277
Précision = 0.8026143790849674
```

FIGURE 16 – AdaBoost

6.1.3 Feature *SEX* retiré

```
Confusion matrix on test:
[[396  64]
 [ 85 238]]
Rappel = 0.7368421052631579
Précision = 0.7880794701986755
-----------------------------------------
Confusion matrix on train:
[[1551  294]
 [ 294  991]]
Rappel = 0.7712062256809339
Précision = 0.7712062256809339
```

FIGURE 17 – SVM

```
Confusion matrix on test:
[[399  61]
 [ 86 237]]
Rappel = 0.7337461300309598
Précision = 0.7953020134228188
-----------------------------------------
Confusion matrix on train:
[[1747   98]
 [ 101 1184]]
Rappel = 0.9214007782101167
Précision = 0.9235569422776911
```

FIGURE 18 – Random Forest

```
Confusion matrix on test:
[[399  61]
 [104 219]]
Rappel = 0.6780185758513931
Précision = 0.7821428571428571
-----------------------------------------
Confusion matrix on train:
[[1604  241]
 [ 299  986]]
Rappel = 0.7673151750972763
Précision = 0.8035859820700897
```

FIGURE 19 – Gradient Boosting

```
Confusion matrix on test:
[[400  60]
 [112 211]]
Rappel = 0.653250773993808
Précision = 0.7785977859778598
-----------------------------------------
Confusion matrix on train:
[[1578  267]
 [ 319  966]]
Rappel = 0.7517509727626459
Précision = 0.7834549878345499
```

FIGURE 20 – AdaBoost

On peut voir que les performances ne sont pas tout à fait égales pour les différentes valeurs du feature *SEX* et également quand on retire cette dernière du jeu de données initial. Prenons par exemple le rappel (TP/(TP+FN)) qui, plus est élévé, plus il y a de vrai positif. On remarque que sur l'ensemble test, le rappel est meilleur pour les 4 modèles d'apprentissage sur le jeu de données où les individus *Femme* ont été retirés par rapport au jeu de données où les individus *Homme* ont été retirés et également au jeu de données où la feature *SEX* a été retirée. On peut par exemple dire que les modèles sont plus ou moins biaisés et qu'ils arrivent à mieux prédire le label pour les individus *Femme*.

7 Conclusion

Lors de ces différentes expérimentations, j'ai pu tester différents modèles d'apprentissage (*SVM, Random Forest, Gradient Boosting et AdaBoost*) pour un problème de classification. J'ai pris conscience de l'importance du choix des hyperparamètres pour avoir les meilleurs résultats qu'un modèle peut fournir selon un dataset donné. J'ai également cherché à expliquer les modèles et donc leurs prédictions, notamment en analysant chacun des features du jeu de données. En effet, les features auront des importances variées d'un modèle à un autre pour la prédiction du label. J'ai enfin traité sur l'équité des modèles en considérant au moins une feature comme sensible et en analysant la performance des modèles vis-à-vis de cette caractéristique sensible pour mettre en évidence ou non un aspect biaisé du modèle.