CAYLA GINESTET Thomas
MECALIFF Marie
5SDBD
2023 / 2024

Rapport de Travaux Pratiques

Apprentissage Supervisé

Lien dépôt github : https://github.com/thomasclgnt/AS.git

Sommaire :

I. Préparation et compréhension des données

A. Présentation du dataset

Dans le cadre de ces travaux pratiques, nous travaillons sur le jeu de données ACSIncome, afin de prédire si un individu possède un revenu annuel supérieur à 50,000$ par an, ici pour l'Etat de Californie. En effet, ce dataset contient des données sur les revenus de personnes vivant aux Etats-Unis, ainsi que des informations individuelles complémentaires tels que l'âge, le niveau d'éducation, ou encore le sexe. L'objectif est donc d'analyser ce dataset en utilisant des méthodes d'apprentissage supervisé de classification, afin d'essayer de prédire le niveau de revenu annuel (supérieur à 50,000$ ou non), et d'identifier les attributs les plus décisifs vis-à-vis de ce niveau de revenu.

Les différents attributs considérés dans le dataset sont les suivants : âge (AGEP), classe de travailleur (COW), niveau d'études (SCHL), état civil (MAR), profession (OCCP), lieu de naissance (POBP), lien de parenté à la personne référente (RELP), heures travaillées par semaine au cours des 12 derniers mois (WKHP), sexe (SEX) et ethnie (RACIP).

Le dataset est donc divisé en attributs (features) et en cibles (labels), ces dernières correspondant aux valeurs booléennes qualifiant le niveau de revenu annuel de l'individu (supérieur à 50,000$ ou non). Il est composé de 195 665 échantillons, et cette taille très importante nous a obligés à travailler sur de plus petites portions du dataset selon les méthodes, comme expliqué dans la suite du rapport.

B. Analyse exploratoire

Pour commencer une rapide analyse exploratoire des données, nous avons examiné la répartition des échantillons en fonction du revenu annuel. Sur les 195,665 échantillons disponibles, nous constatons que 115 330 individus ont un revenu annuel inférieur à 50,000$, tandis que 80 335 individus ont un revenu annuel supérieur à 50,000$, soit un peu moins d'un tiers du dataset. Cette répartition déséquilibrée entre les deux classes doit être prise en compte lors de la modélisation et de l'évaluation des performances des différents modèles de classification.

Nous avons également choisi de visualiser la répartition des données selon la cible (PINCP) pour chaque feature. Cette approche nous a permis de comprendre la distribution de chaque variable et d'identifier des différences significatives entre les deux tranches de revenus, en affichant les courbes de densité par noyau (KDE). Cela nous a aidé à détecter des motifs et des relations potentielles entre les features et les labels. De plus, cette méthode permet d'identifier facilement des valeurs aberrantes (outliers) et d'informer les étapes de prétraitement des données, comme la normalisation et la transformation.

Par exemple, comme visible dans les graphiques ci-dessous, la visualisation de l'âge (AGEP) en fonction des différentes tranches de revenus (PINCP) peut révéler des tendances spécifiques : les individus de moins de 30 sont en très grande majorité dans la catégorie de revenus inférieurs, tandis que la répartition semble s'égaliser à partir de 40 ans.

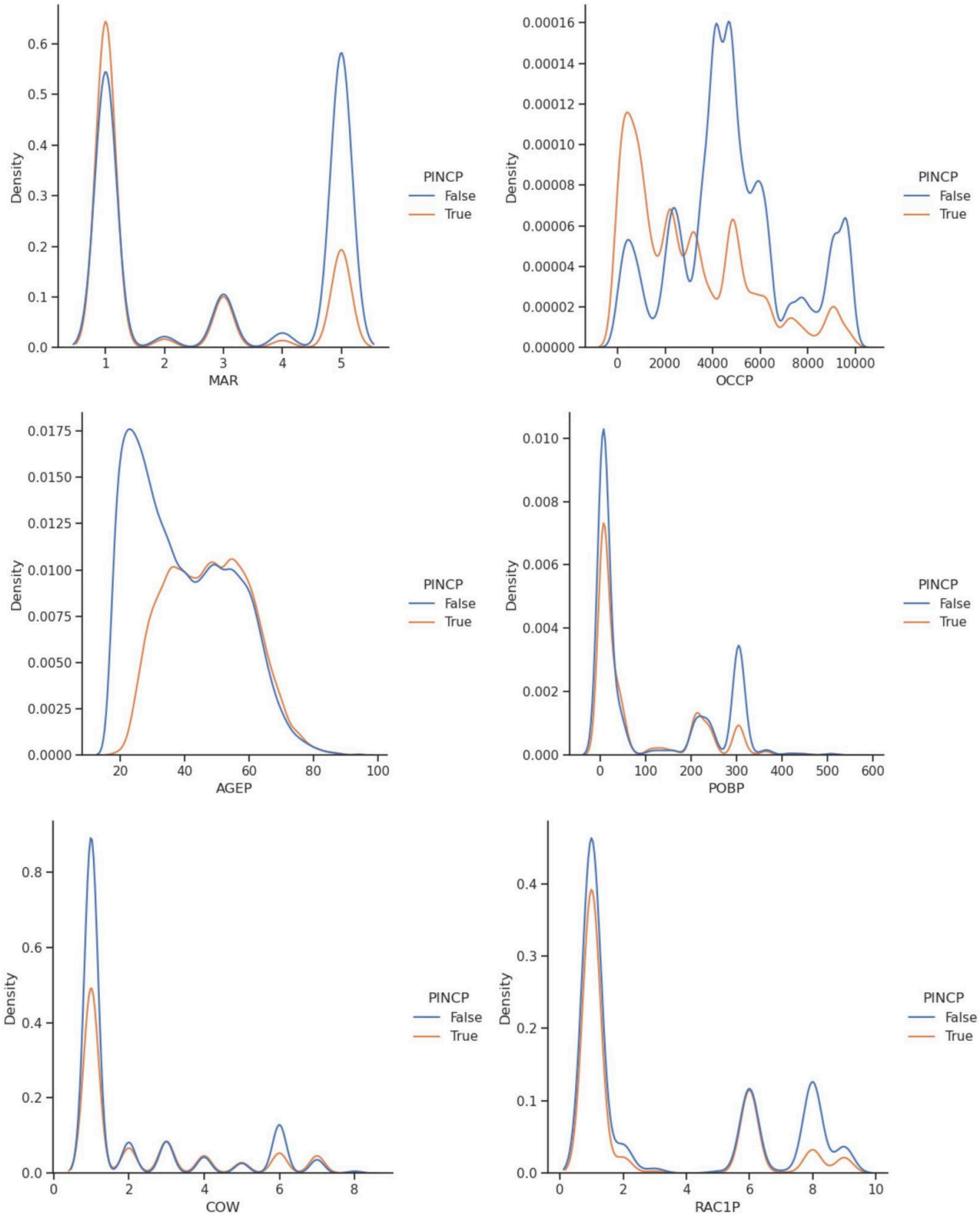


Fig. 1 : Répartition de la densité par label sur les features AGEP, MAR, COW, OCCP, POBP et RAC1P

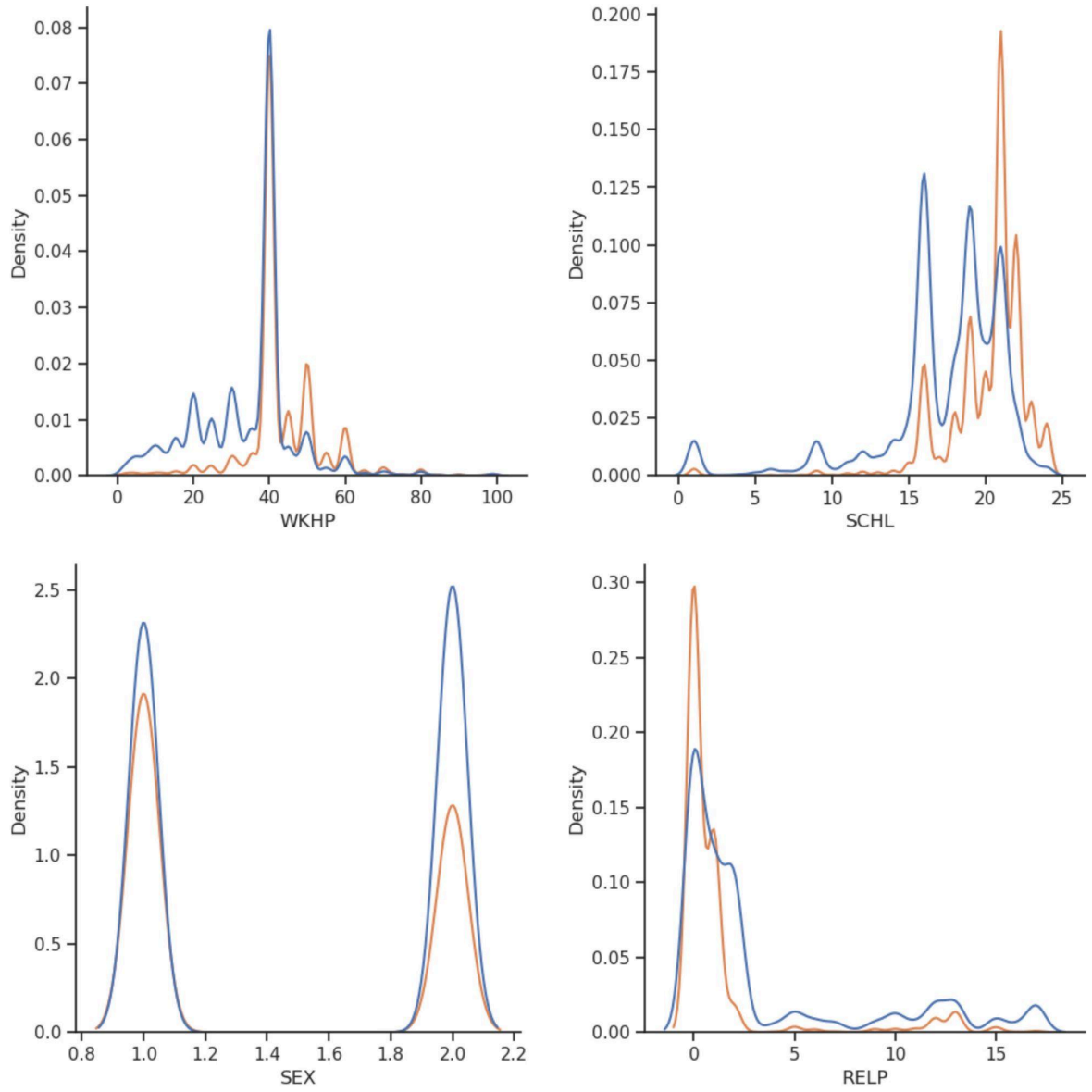


Fig. 2 : Répartition de la densité par label sur les features WKHP, SCHL, SEX et RELP

Nous pouvons également noter la différence de répartition entre niveaux de revenus entre les sexes, avec un écart de distribution entre les deux catégories beaucoup plus important pour les femmes, ainsi que l'importance apparente du niveau d'étude.

C. Préparation des données

Tout d'abord, nous avons mélangé les données à l'aide de la méthode *shuffle* de scikit-learn. En effet, cela permet de garantir que l'ordre des échantillons n'influence pas le modèle, évitant ainsi tout biais potentiel dans l'apprentissage.

Comme mentionné précédemment, de part la taille importante du dataset et pour maintenir des temps d'exécution raisonnables, nous avons dû découper le dataset afin d'effectuer nos études que sur une portion de ce dernier, variant de 1 à 50% selon la méthode et les étapes concernées. Dans cette portion du dataset traitée, on utilise la

méthode *train_test_split* afin d'en allouer 80% à l'entraînement des modèles, et de garder les 20% de données restantes pour le test.

Enfin, nous avons mis à l'échelle les données grâce au *StandardScaler*, afin de s'assurer que toutes les features contribuent équitablement au modèle en les plaçant sur la même échelle. Cela permet d'améliorer la convergence ainsi que la performance globale des modèles. Nous n'avons cependant pas standardisé le set de labels, car ces derniers correspondent à des valeurs booléennes qui ne nécessitent donc pas de mise à l'échelle.

II. Recherche des meilleurs modèles

Dans cette partie, nous détaillons les étapes de notre travail de recherche des meilleurs paramètres pour chaque modèle sur le dataset considéré.

Nous avons travaillé sur quatre modèles de classification : le modèle Support Vector Machine, le modèle Random Forest, Adaboost et Gradient Boosting. Nous sommes partis du résultat du modèle paramétré par défaut, puis nous avons essayé d'approcher individuellement le meilleur réglage pour chaque paramètre du modèle et enfin nous avons comparé les résultats avec la recherche *GridsearchCV* des meilleurs hyperparamètres.

A. Support Vector Machine

Avant toute chose, nous avons entraîné le modèle avec les paramètres par défaut sur différents pourcentages du dataset afin de se faire une première idée de son efficacité et du temps nécessaire à son exécution selon la quantité de données fournies.
Voici les résultats que nous avons obtenu sur le jeu de données test selon le pourcentage du dataset considéré : **77.5%** de réussite sur 1% du dataset, **79.7%** de réussite sur 10% du dataset, **80.4%** de réussite sur 20% du dataset et **80.6%** de réussite sur 50% du dataset. De par le niveau très important du temps d'exécution observé sur 50%, nous n'avons pas tenté de lancer l'entraînement et le test de SVM sur l'entièreté du dataset.

1. Validation croisée du modèle par défaut

Nous sommes ensuite passés à l'étape de validation croisée qui minimise les risques de surajustement et fournit une estimation fiable des performances du modèle, en divisant les données en ensembles d'entraînement et de test multiple. Pour ces mêmes raisons de temps d'exécution, nous avons réalisé cette validation croisée sur 5% du dataset. Nous avons alors obtenu un score de validation croisée de **80%** pour le modèle par défaut.

2. Repérage des paramètres décisifs

Par la suite, avant de nous lancer dans une recherche des meilleurs hyperparamètres avec la méthode *GridsearchCV*, nous avons voulu évaluer le niveau d'influence de chacun des paramètres sur le résultat obtenu par SVM en traçant une courbe de validation, afin de savoir quels paramètres renseigner sur *GridsearchCV* et sur quel intervalle de valeurs affiner la recherche. Nous n'avons pour cela pas utilisé la méthode *validation_curve* mais nous avons effectué un équivalent par nous-même à l'aide d'une boucle for de validation croisée sur différentes valeurs de chaque hyperparamètre.

Voici les courbes que nous avons obtenu pour chaque paramètre :

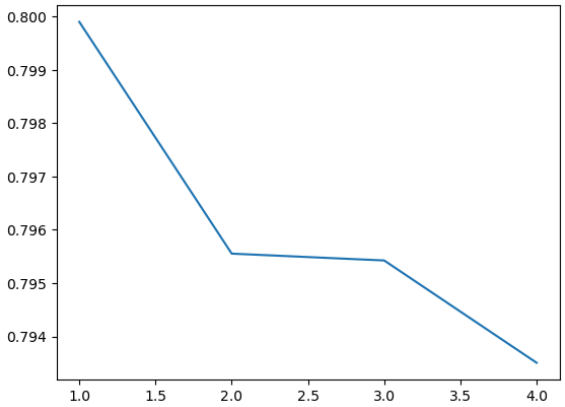


Fig. 3 : Variation du résultat de validation croisée selon les valeurs de C

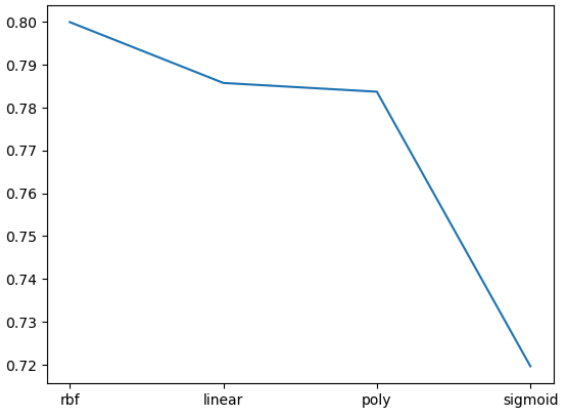


Fig. 4 : Variation du résultat de validation croisée selon les valeurs de kernel

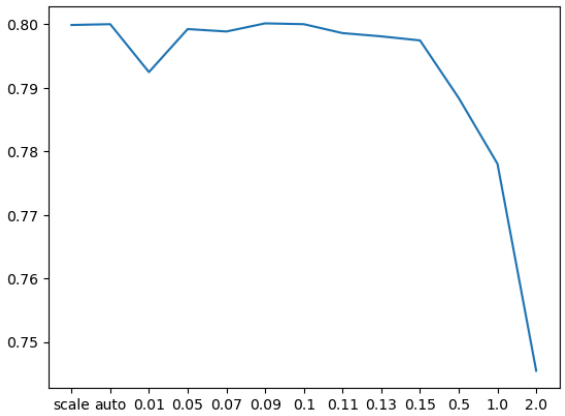


Fig. 5 : Variation du résultat de validation croisée selon les valeurs de gamma

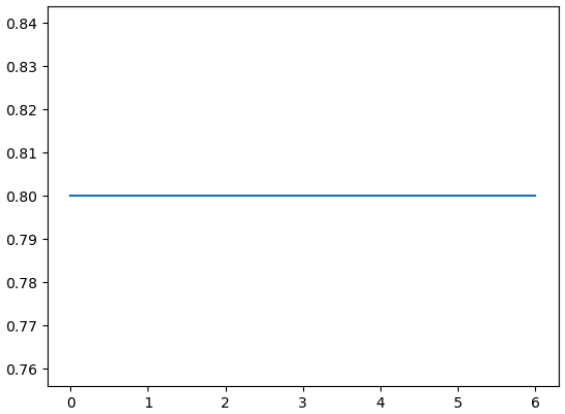


Fig. 6 : Variation du résultat de validation croisée selon les valeurs de degree

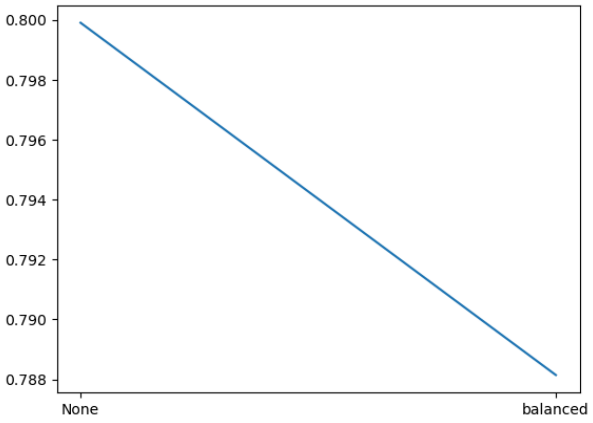


Fig. 7 : Variation du résultat de validation croisée selon les valeurs de class_weigth

Nous avons donc exclu *degree* et *class_weigth* de la future recherche *GridsearchCV,* de part l'absence d'influence du premier et le fait que le deuxième ne puisse prendre que deux valeurs, dont la meilleure est "None".

Par ailleurs, ces courbes de validation nous ont permis de construire notre propre meilleur modèle “manuel”, en se basant sur les maximum de chaque courbe obtenue, à comparer avec le meilleur modèle obtenu par *GridsearchCV* par la suite.

Nous avons donc réentraîné SVM avec les paramètres correspondant à notre meilleur modèle, soit C=1, kernel=rbf, gamma=0.09 et class_weight=None, en obtenant cette fois-ci le score de validation croisée de **80.02%**.

3. Gridsearch : recherche des meilleurs hyperparamètres

Nous avons mené une recherche des meilleurs hyperparamètres en utilisant la fonction *GridsearchCV*. Nous avons réalisé ceci sur une portion de 1 puis de 5% du dataset pour SVM (*voir Annexe 1*), mais nous avons dû nous arrêter à ce dernier pourcentage, son exécution ayant duré près d’1h30. Cela nous a permis d’obtenir, pour 1% du dataset, le meilleur score de **74.2%**, associé aux paramètres suivants : {'C': 1.0, 'gamma': 0.8, 'kernel': 'rbf'}.
Pour 5% du dataset, nous avons obtenu un meilleur score de **78.2%**, associé aux paramètres suivants : {'C': 1.5, 'gamma': 0.85, 'kernel': poly}.

B. Random Forest

Avant toute chose, comme pour le modèle précédent, nous avons entraîné le modèle avec les paramètres par défaut sur différents pourcentages du dataset. Voici les résultats que nous avons obtenu sur le jeu de données test selon le pourcentage du dataset considéré : **75.8%** de réussite sur 1% du dataset, **79.6%** de réussite sur 10% du dataset, **80.8%** de réussite sur 20% du dataset, **81.1%** de réussite sur 50% du dataset, et **81.2%** sur 100%.

1. Validation croisée du modèle par défaut

Nous sommes ensuite passés à l’étape de validation croisée sur 5% du dataset. Nous avons obtenu un score de validation croisée de **79.8%**.

2. Repérage des paramètres décisifs

Par la suite, avant de nous lancer dans une recherche des meilleurs hyperparamètres avec la méthode *GridsearchCV*, nous avons réalisé nos courbes de validation afin de repérer les paramètres à forte influence et de repérer leur potentielle meilleure valeur.

Voici les courbes que nous avons obtenu pour chaque paramètre :

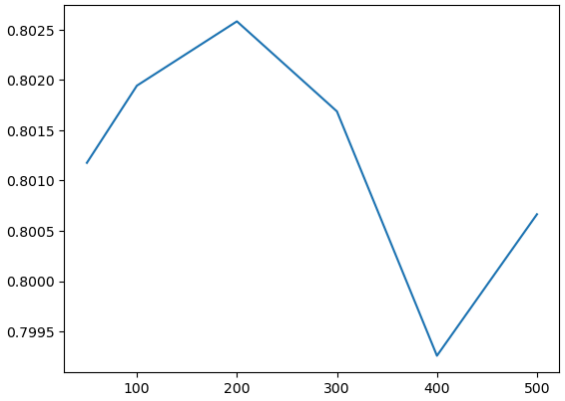


Fig. 8 : Variation du résultat de validation croisée selon les valeurs de n_estimators

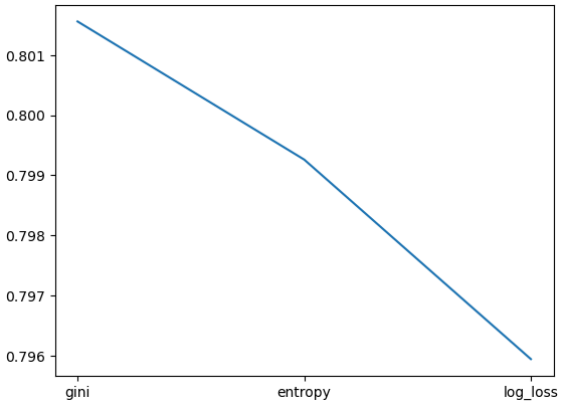


Fig. 9 : Variation du résultat de validation croisée selon les valeurs de criterion

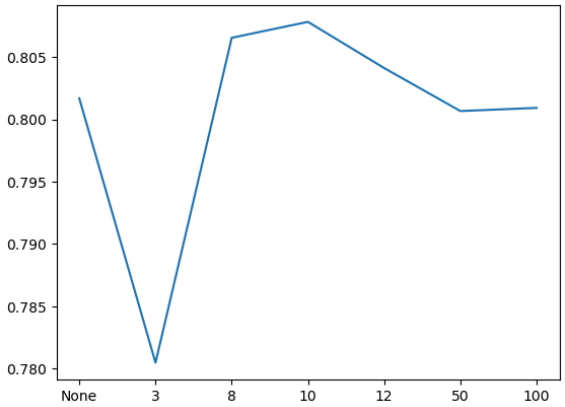


Fig. 10 : Variation du résultat de validation croisée selon les valeurs de max_depth

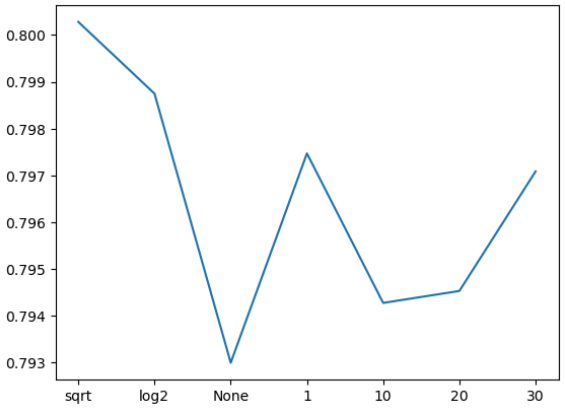


Fig. 11 : Variation du résultat de validation croisée selon les valeurs de max_features

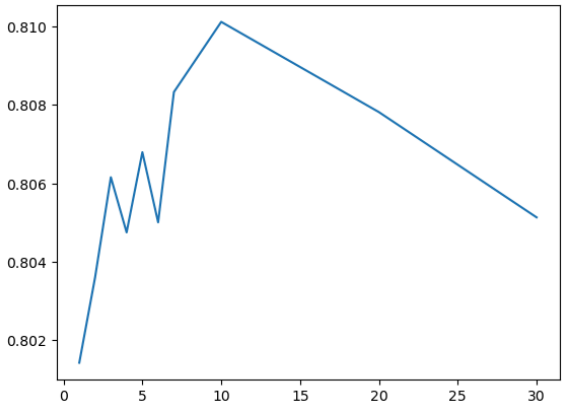


Fig. 12 : Variation du résultat de validation croisée selon les valeurs de min_samples_leaf

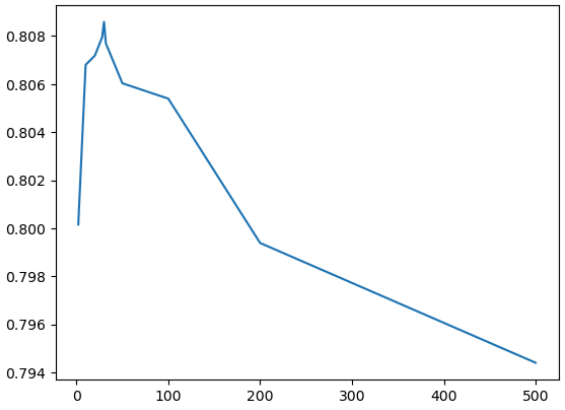


Fig. 13 : Variation du résultat de validation croisée selon les valeurs de min_samples_split

Nous avons donc exclu *max_features* de la future recherche *GridsearchCV,* la meilleure valeur du paramètre semblant être celle par défaut, et par soucis d'économie de temps sur l'exécution de la recherche gridsearch.

En nous basant sur ces courbes, nous avons réentraîné RandomForest avec les paramètres correspondant à notre meilleur modèle “manuel”, soit n_estimators=200, criterion='log_loss', max_depth=10, min_samples_split=30, min_samples_leaf=6, et max_features='sqrt' , en obtenant cette fois-ci le score de validation croisée de **80.8%**.

3. Gridsearch : recherche des meilleurs hyperparamètres

Nous avons mené une recherche des meilleurs hyperparamètres en utilisant la fonction *GridsearchCV*. Nous avons réalisé ceci sur une portion de 1 puis de 20% du dataset pour RandomForest (*voir Annexe 2*), mais nous avons dû nous arrêter à ce dernier pourcentage, son exécution ayant duré 1h40. Par ailleurs, nous n’avons pas ajouté l’exploration du paramètre critérion sur 20%, pour limiter ce temps d’exécution. Cela nous a permis d’obtenir, pour 1% du dataset, le meilleur score de **77.3%,** associé aux paramètres suivants : {'criterion': 'log_loss', 'max_depth': 12, 'min_samples_leaf': 4, 'min_samples_split': 32, 'n_estimators': 100}. Pour 5% du dataset, nous avons obtenu un meilleur score de **81.4%**, associé aux paramètres suivants : {'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 40, 'n_estimators': 100}.

C. Adaboost

Avant toute chose, comme pour les modèles précédents, nous avons entraîné et évalué le modèle avec les paramètres par défaut sur différents pourcentages du dataset. Voici les résultats que nous avons obtenu sur le jeu de données test selon le pourcentage du dataset considéré : **73.2%** de réussite sur 1% du dataset, **80.3%** de réussite sur 10% du dataset, **80.8%** de réussite sur 20% du dataset, **80.6%** de réussite sur 50% du dataset, et **80.6%** sur 100%. Nous observons ainsi qu’à partir de 20% du dataset, le score diminue légèrement, ce qui peut être le signe d’un léger sur-apprentissage.

1. Validation croisée du modèle par défaut

Nous sommes ensuite passés à l’étape de validation croisée sur 5% du dataset. Nous avons alors obtenu un score de validation croisée de **80,4%** pour le modèle par défaut.

2. Repérage des paramètres décisifs

Par la suite, avant de nous lancer dans une recherche des meilleurs hyperparamètres avec la méthode *GridsearchCV*, nous avons réalisé nos courbes de validation afin de repérer les paramètres à forte influence et de repérer leur potentielle meilleure valeur.

Voici les courbes que nous avons obtenu pour chaque paramètre :

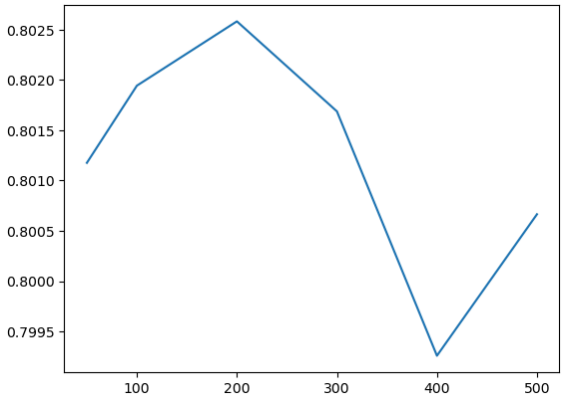


Fig. 14 : Variation du résultat de validation croisée selon les valeurs de n_estimators

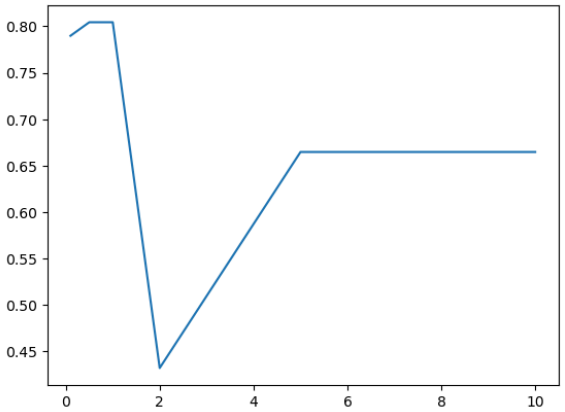


Fig. 15 : Variation du résultat de validation croisée selon les valeurs de learning_rate

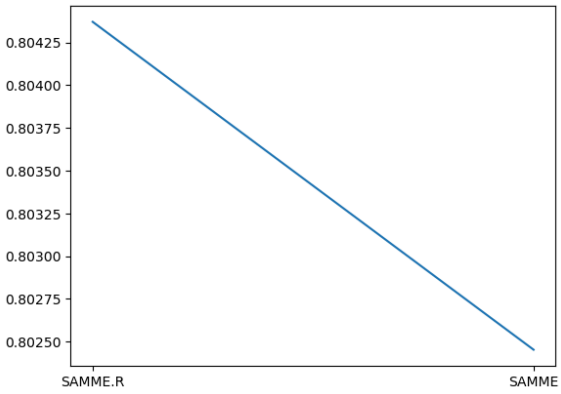


Fig. 16 : Variation du résultat de validation croisée selon les valeurs de algorithm

Nous avons donc considéré ces 3 paramètres pour la future recherche *GridsearchCV.* La documentation de scikit learn précise pour le modèle Adaboost un lien particulier de trade-off entre *n_estimators* et *learning_rate*, un changement de valeur d'un des paramètres peut alors impacter négativement l'autre paramètre. Ce lien n'est ainsi pas visible dans la recherche manuelle des paramètres que nous venons de faire, nous aurons besoin de gridsearch pour s'assurer de trouver la meilleure compatibilité entre ces paramètres.

En nous basant sur ces courbes, nous avons réentraîné Adaboost avec les paramètres correspondant à notre meilleur modèle "manuel", soit n_estimators=600, learning_rate=1 et algorithm='SAMME.R', en obtenant cette fois-ci le score de validation croisée de **81.5%**.

3. Gridsearch : recherche des meilleurs hyperparamètres

Nous avons mené une recherche des meilleurs hyperparamètres en utilisant la fonction *GridsearchCV*. Nous avons réalisé ceci sur une portion de 1% du dataset. Nous n'avons pas pu lancer la recherche des meilleurs hyperparamètres sur une plus grande portion du dataset pour les mêmes raisons qu'évoquées précédemment, liées au temps d'exécution. Cela nous a permis d'obtenir le meilleur score de **76.8%**, associé aux paramètres suivants : {'algorithm': 'SAMME', 'learning_rate': 1.0, 'n_estimators': 600}.

D. Gradient Boosting

Avant toute chose, comme pour les modèles précédents, nous avons entraîné le modèle avec les paramètres par défaut sur différents pourcentages du dataset. Voici les résultats que nous avons obtenu sur le jeu de données test selon le pourcentage du dataset considéré : **77%** de réussite sur 1% du dataset, **80.8%** de réussite sur 10% du dataset, **81.6%** de réussite sur 20% du dataset, **81.5%** de réussite sur 50% du dataset, et **81.3%** sur 100%.

1. Validation croisée du modèle par défaut

Nous sommes ensuite passés à l'étape de validation croisée sur 5% du dataset. Nous avons alors obtenu un score de validation croisée de **80,9%** pour le modèle paramétré par défaut.

2. Repérage des paramètres décisifs

Par la suite, avant de nous lancer dans une recherche des meilleurs hyperparamètres avec la méthode *GridsearchCV*, nous avons réalisé nos courbes de validation afin de repérer les paramètres à forte influence et de repérer leur potentielle meilleure valeur.

Voici les courbes que nous avons obtenu pour chaque paramètre :

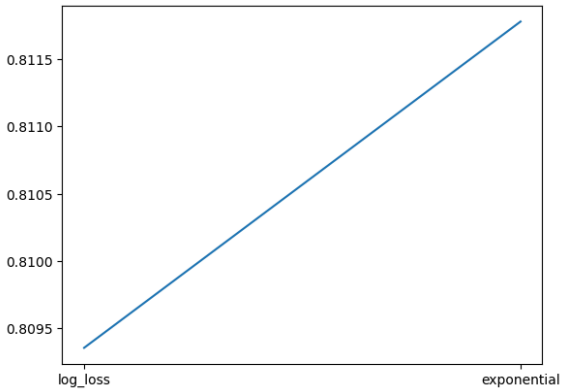


Fig. 17 : Variation du résultat de validation croisée selon les valeurs de loss

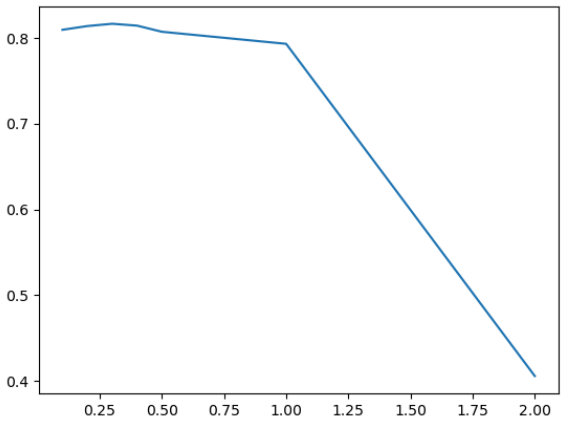
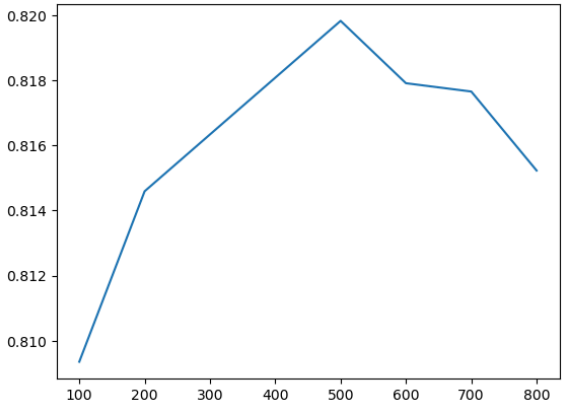


Fig. 18 : Variation du résultat de validation croisée selon les valeurs de learning_rate



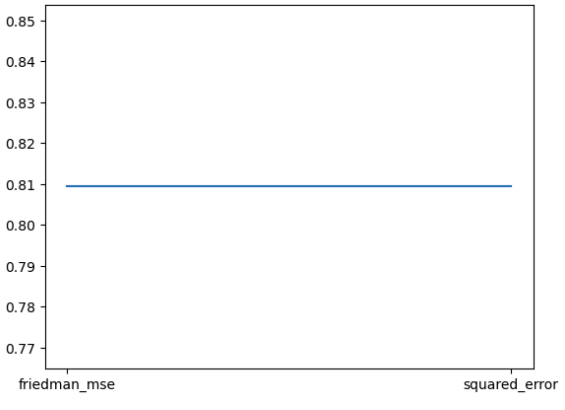


Fig. 19 : Variation du résultat de validation croisée selon les valeurs de n_estimators

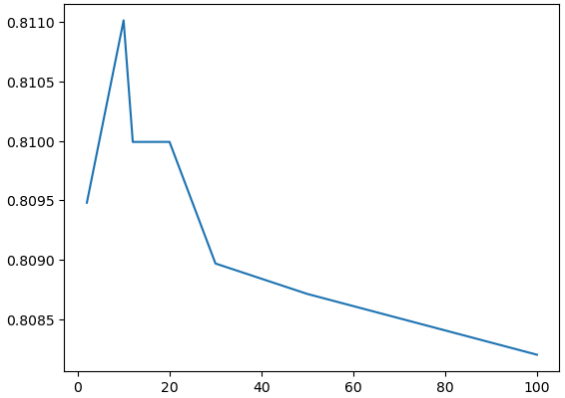


Fig. 21 : Variation du résultat de validation croisée selon les valeurs de min_samples_split

Fig. 20 : Variation du résultat de validation croisée selon les valeurs de criterion

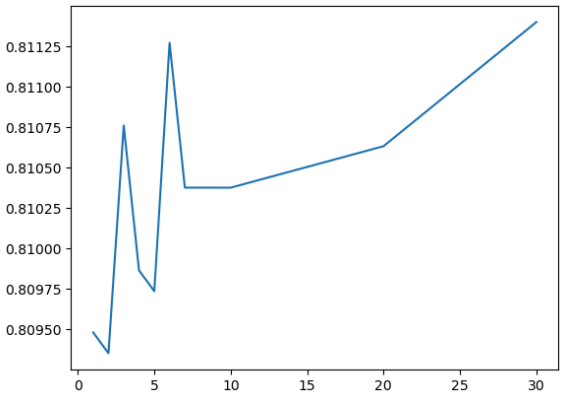


Fig. 22 : Variation du résultat de validation croisée selon les valeurs de min_samples_leaf

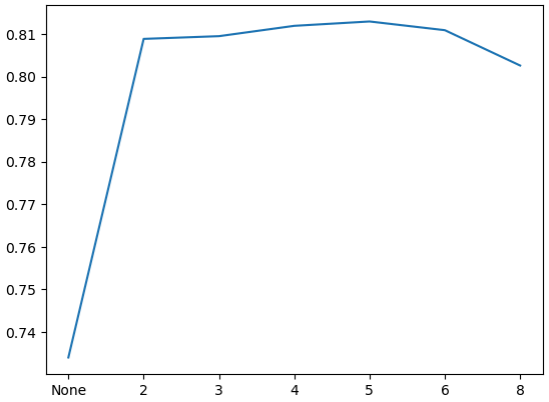


Fig. 23 : Variation du résultat de validation croisée selon les valeurs de max_depth

Nous avons donc exclu *criterion* de la future recherche *GridsearchCV,* de part son absence d'influence sur le score de validation.

En nous basant sur ces courbes, nous avons réentraîné Gradient Boosting avec les paramètres correspondant à notre meilleur modèle "manuel", soit loss='exponential', n_estimators=500, criterion='friedman_mse', min_samples_split=10, min_samples_leaf=6 et max_depth=6, en obtenant cette fois-ci le score de validation croisée de **80.4%**.

3. Gridsearch : recherche des meilleurs hyperparamètres

Nous avons mené une recherche des meilleurs hyperparamètres en utilisant la fonction *GridsearchCV*. Nous avons réalisé ceci sur une portion de 1% du dataset , pour les mêmes raisons qu'évoquées précédemment liées au temps d'exécution. Cela nous a permis d'obtenir le meilleur score de **76.5%**, associé aux paramètres suivants : {'learning_rate': 0.3, 'loss': 'exponential', 'max_depth': 5, 'min_samples_leaf': 3, 'min_samples_split': 8, 'n_estimators': 550}.

III. Analyse et comparaison des modèles

A. Évaluation et comparaison des modèles

Afin d'évaluer et comparer nos différents modèles, nous avons utilisés les métriques suivantes :

- **Accuracy** : Cette métrique permet d'évaluer la qualité d'un modèle d'apprentissage supervisé en classification binaire, en évaluant le taux de prédictions correctes (True Positive et True Negative sur l'ensemble des productions)
- **Classification_report** : Ce tableau recense plusieurs métriques :
    - la précision représente la proportion de prédictions positives (ou négatives) correctes parmi toutes les prédictions positives (ou négatives)
    - le recall représente la proportion de prédictions positives correctes parmi les valeurs positives réelles ;
    - le f1-score combine les mesures de précision et de rappel ;
    - la moyenne macro ("macro avg") représente la moyenne des mesures pour chaque classe sans prendre en compte leur taille.
    - la moyenne pondérée ("weighted avg") prend en compte la taille de chaque classe. La colonne support représente le nombre total d'échantillons réels appartenant à chaque classe.
- **Matrice de confusion** : Cette matrice permet d'évaluer la performance d'un modèle en montrant le nombre de prédictions correctes et incorrectes par rapport aux données réelles.

1. SVM

Pour notre meilleur modèle obtenu de manière "manuelle", nous avons obtenu les résultats suivants :

	Précision	Recall	f1-score	Support
False	0.83	0.83	0.83	1165
True	0.75	0.75	0.75	792
Accuracy			0.80	1957
Macro avg	0.79	0.79	0.79	1957
Weighted avg	0.80	0.80	0.80	1957

Fig. 24 : Tableau "classification_report" pour meilleur modèle manuel sur SVM

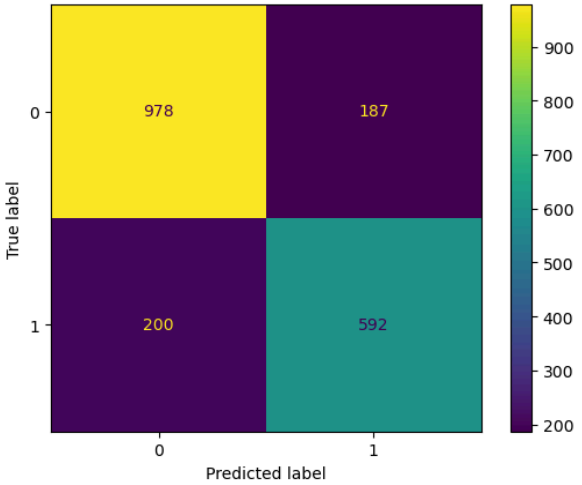


Fig. 25 : Matrice de confusion pour meilleur modèle manuel sur SVM

Pour le meilleur modèle obtenu par la recherche *GridsearchCV* nous avons obtenu les résultats suivants :

	Précision	Recall	f1-score	Support
False	0.73	0.86	0.79	217
True	0.77	0.60	0.68	175
Accuracy			0.74	392
Macro avg	0.75	0.73	0.73	392
Weighted avg	0.75	0.74	0.74	392

Fig. 26 : Tableau "classification_report" pour meilleur GridsearchCV sur SVM

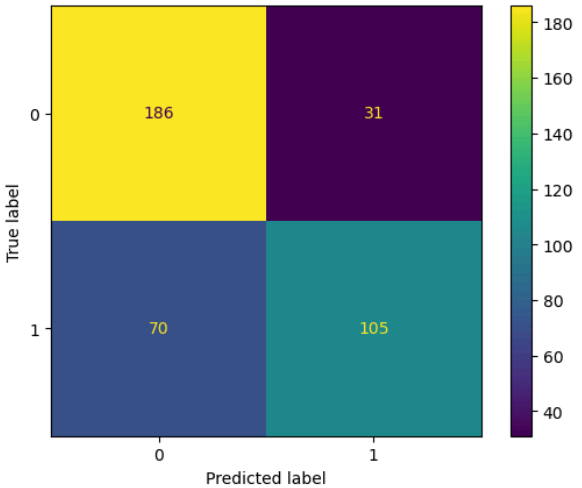


Fig. 27 : Matrice de confusion pour meilleur modèle GridsearchCV sur SVM

Pour le modèle SVM, nous remarquons que notre modèle manuel a un meilleur score d'accuracy (+ 6%), une meilleure précision sur la prédiction des faux, mais une moins bonne précision sur les cas True.

2. RandomForest

Pour notre meilleur modèle obtenu de manière “manuelle”, nous avons obtenu les résultats suivants :

	Précision	Recall	f1-score	Support
False	0.83	0.84	0.83	1165
True	0.76	0.75	0.75	792
Accuracy			0.80	1957
Macro avg	0.80	0.79	0.79	1957
Weighted avg	0.80	0.80	0.80	1957

Fig. 28 : Tableau “classification_report” pour meilleur modèle manuel sur RandomForest

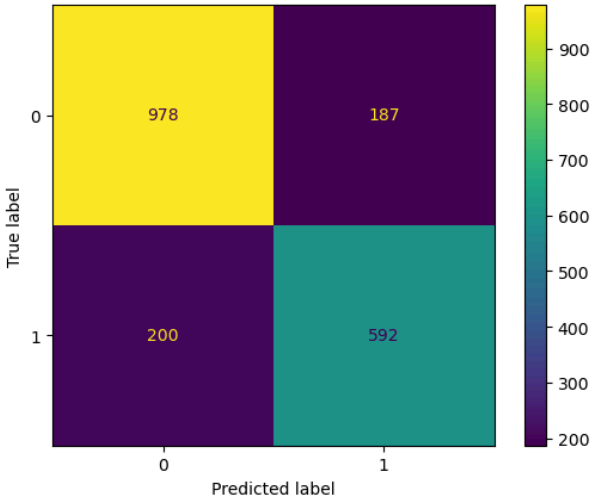


Fig. 29 : Matrice de confusion pour meilleur modèle manuel sur RandomForest

Pour le meilleur modèle obtenu par la recherche *GridsearchCV* nous avons obtenu les résultats suivants :

	Précision	Recall	f1-score	Support
False	0.75	0.88	0.81	217
True	0.81	0.65	0.72	175
Accuracy			0.77	392
Macro avg	0.78	0.76	0.76	392
Weighted avg	0.78	0.77	0.77	392

Fig. 30 : Tableau “classification_report” pour meilleur GridsearchCV sur RandomForest

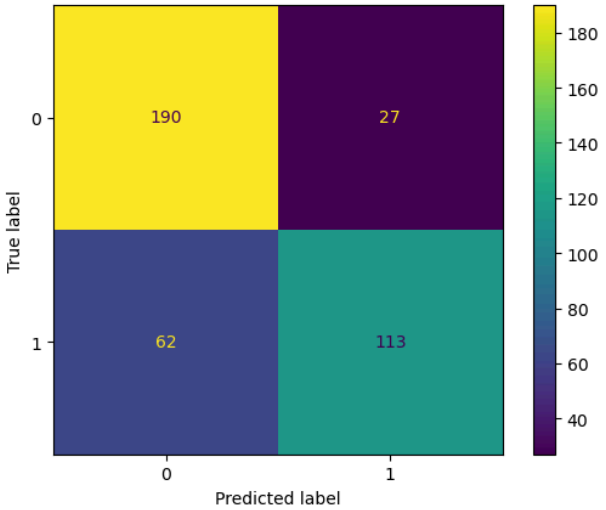


Fig. 31 : Matrice de confusion pour meilleur modèle GridsearchCV sur RandomForest

Pour le modèle Random Forest, le constat est identique au modèle SVM, mais l'écart d'accuracy entre le meilleur modèle manuel et le modèle trouvé avec gridsearch est plus faible.

3. Adaboost

Pour notre meilleur modèle obtenu de manière "manuelle", nous avons obtenu les résultats suivants :

	Précision	Recall	f1-score	Support
False	0.84	0.84	0.84	1165
True	0.76	0.75	0.76	792
Accuracy			0.81	1957
Macro avg	0.80	0.80	0.80	1957
Weighted avg	0.81	0.81	0.81	1957

Fig. 32 : Tableau "classification_report" pour meilleur modèle manuel sur Adaboost

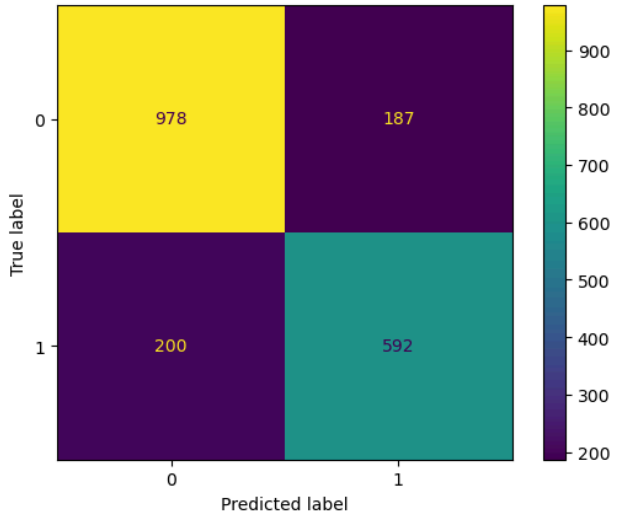


Fig. 33 : Matrice de confusion pour meilleur modèle manuel sur Adaboost

Pour le meilleur modèle obtenu par la recherche *GridsearchCV* nous avons obtenu les résultats suivants :

	Précision	Recall	f1-score	Support
False	0.74	0.90	0.81	217
True	0.83	0.60	0.72	175
Accuracy			0.77	392
Macro avg	0.79	0.75	0.76	392
Weighted avg	0.78	0.77	0.76	392

Fig. 34 : Tableau "classification_report" pour meilleur GridsearchCV sur Adaboost

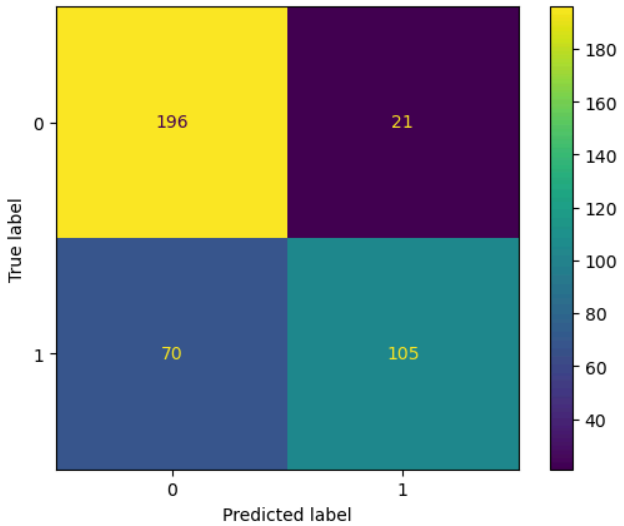


Fig. 35 : Matrice de confusion pour meilleur modèle GridsearchCV sur Adaboost

Pour le modèle Adaboost, le modèle manuel est légèrement meilleur en accuracy (+ 4%) et meilleur sur la précision de prédiction des False, mais moins efficace que le modèle gridsearch pour la précision de prédiction positive.

4. Gradient Boosting

Pour notre meilleur modèle obtenu de manière “manuelle”, nous avons obtenu les résultats suivants :

	Précision	Recall	f1-score	Support
False	0.84	0.81	0.82	1165
True	0.73	0.78	0.76	792
Accuracy			0.80	1957
Macro avg	0.79	0.79	0.79	1957
Weighted avg	0.80	0.80	0.80	1957

Fig. 36 : Tableau “classification_report” pour meilleur modèle manuel sur Gradient Boosting

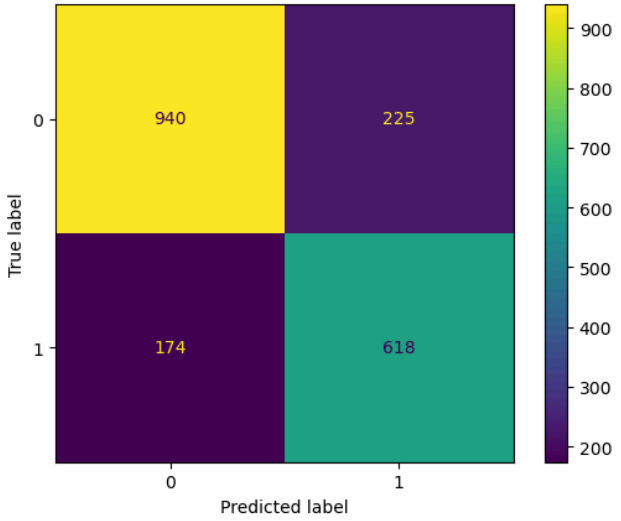


Fig. 37 : Matrice de confusion pour meilleur modèle manuel sur Gradient Boosting

Pour le meilleur modèle obtenu par la recherche *GridsearchCV* nous avons obtenu les résultats suivants :

	Précision	Recall	f1-score	Support
False	0.75	0.88	0.81	217
True	0.80	0.63	0.71	175
Accuracy			0.77	392
Macro avg	0.77	0.75	0.76	392
Weighted avg	0.77	0.77	0.76	392

Fig. 38 : Tableau “classification_report” pour meilleur GridsearchCV sur Gradient Boosting

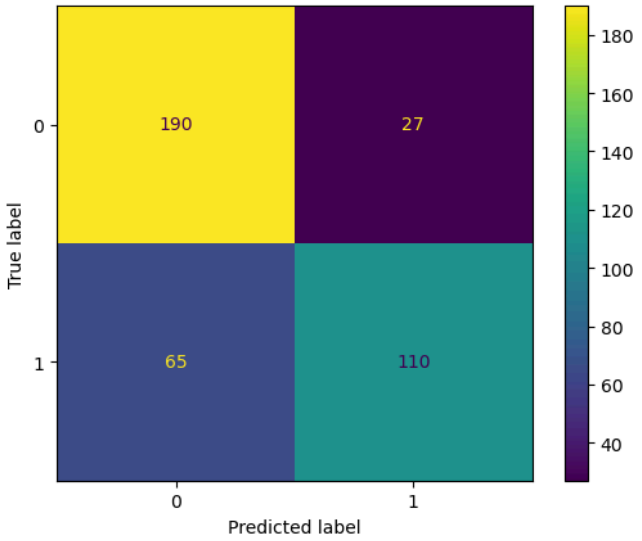


Fig. 39 : Matrice de confusion pour meilleur modèle GridsearchCV sur Gradient Boosting

Pour le modèle Gradient Boosting, comme pour les modèles précédents, nous retrouvons une meilleure valeur d'accuracy pour le modèle manuel (+ 3%), une moins bonne précision sur la prédiction des True mais une meilleure pour la prédiction négative.

B. Bilan sur les meilleurs modèles d'apprentissage

Nous allons maintenant comparer les résultats obtenus par la meilleure version de chacun des meilleurs modèles. Ces résultats peuvent être relativisés, nos meilleurs modèles obtenus par *GridsearchCV* obtenant de moins bons résultats que nos meilleurs modèles obtenus manuellement, les premiers ayant été réalisés sur de petites portions du dataset de par les contraintes de temps associées. Pour pouvoir comparer convenablement les différents modèles, il serait nécessaire de réaliser les différentes étapes d'analyse et de recherche sur la plus grande partie possible du jeu de données.

On constate tout de même dès les premiers tests sur les modèles par défaut de 1 à 100% du dataset qu'au-delà de 20%, les gains de performance deviennent négligeables. Cela indique que le modèle atteint un plateau en termes de performance, où ajouter plus de données n'apporte plus de bénéfice significatif. Pour Adaboost et Gradient Boosting, testés avec le paramétrage par défaut, on observe même une diminution du score de prédiction au fur et à mesure de l'augmentation de la taille des données utilisées ce qui peut être le signe d'un phénomène de surapprentissage du modèle.

Score de prédiction du modèle par défaut pour :	Adaboost	Gradient Boosting
1 % du dataset	73.21	77.04
10 % du dataset	80.33	80.76
20 % du dataset	80.84	81.60

50 % du dataset	80.61	81.54
100 % du dataset	80.63	81.32

Fig. 40 : Tableau des scores de prédiction par rapport à l'augmentation de la taille du jeu de donnée pour Adaboost et Gradient Boosting

De plus, d'après le calcul de la métrique "accuracy", en nous basant sur nos meilleurs modèles "manuels", le modèle avec la meilleure exactitude sur ses prédictions est Adaboost avec 81%, suivi de SVM et RandomForest et GradientBoosting à 80%. En nous basant sur les résultats de *GridsearchCV*, les modèles avec la meilleure exactitude sur ses prédictions sont RandomForest, Adaboost et Gradient Boost avec 77%, suivis de SVM à 74%.

Par ailleurs, les valeurs de recall et f1-score sont très proches des valeurs de précision, sauf pour les meilleurs modèles obtenus par *GridsearchCV*.

Enfin, nous remarquons que tous les modèles manuels sont plus précis pour prédire les faux, les revenus annuels inférieurs à 50 000 $, que les vrais, les revenus annuels supérieurs à 50 000 $, alors que les meilleurs modèles *GridsearchCV* ont les résultats inverses.

Sur les modèles manuels, nous observons aussi des écarts importants de précision entre les faux et les vrais. Ces écarts peuvent s'expliquer en partie par le déséquilibre des labels des données d'entraînement. En effet, les valeurs positives et négatives ne sont pas également réparties dans le jeu de données que nous utilisons : près de 60% des données représentent des personnes avec des revenus inférieurs à 50 000 $.

Pour arriver à un meilleur résultat, il pourrait être nécessaire de pondérer les données pour arriver à un équilibre entre les données positives et négatives pour obtenir des résultats plus équilibrés.

IV. Projections sur les données du Nevada et du Colorado

Pour chacun des quatre modèles étudiés, nous avons utilisé le meilleur modèle avec les meilleurs résultats en termes d'accuracy pour essayer de prédire les outcomes dans le cas des États américains du Nevada et du Colorado. Le but est ainsi d'analyser les résultats de prédiction obtenus par nos modèles sans qu'ils ne soient entraînés avec les features de ces deux états. Cette analyse permet ainsi d'observer l'homogénéité des profils socio-économiques entre ces trois états, de comparer les populations de ces états et de comprendre si les données de Californie sont suffisamment représentatives des données de cette région Ouest des Etats-Unis pour que les modèles soient utilisées pour faire des prédictions dans d'autres Etats que la Californie.

Voici la répartition des données pour le Nevada et le Colorado :

Etat	**Nevada**	**Colorado**
Taille du dataset	10 785	31 306
Nombre de personne avec un PINCP > 50 000 $	3 368 (31%)	12 972 (41 %)
Nombre de personnes avec un PINCP < 50 000 $	7 417 (69%)	18 334 (59 %)

Fig. 41 : Tableau de la répartition des données pour le Nevada et le Colorado

Nous remarquons immédiatement la différence de répartition du label dans les données entre ces deux États, puisqu'il y a plus de personnes qui ont un revenu supérieur à 50 000 $ dans les données du Colorado que dans celles du Nevada.

Comme ces deux jeux de données sont sensiblement moins grands que celui de la Californie, les prédictions que nous avons réalisées dans cette partie ont été faites sur l'ensemble des jeux de données.

Avant d'analyser les résultats obtenus, voici les matrices de confusion pour chaque modèle avec les données du Nevada et du Colorado et un tableau récapitulatif des valeurs d'accuracy pour chaque prédiction :

Nevada	***Colorado***

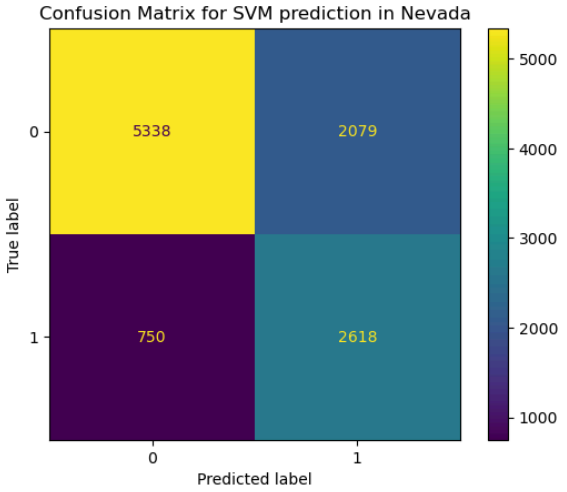
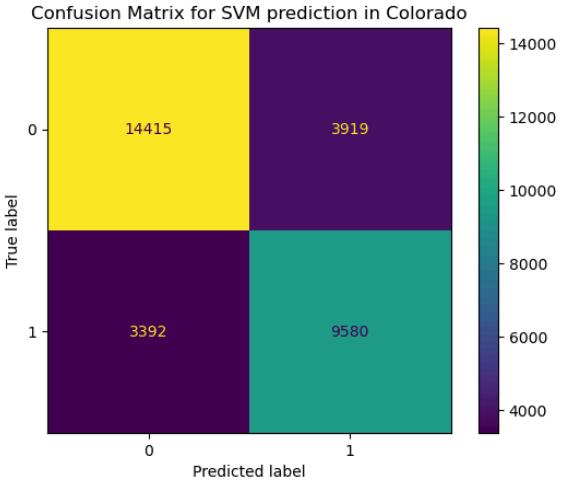

	

Fig. 42 : Matrice de confusion avec SVM	*Fig. 43 : Matrice de confusion avec SVM*

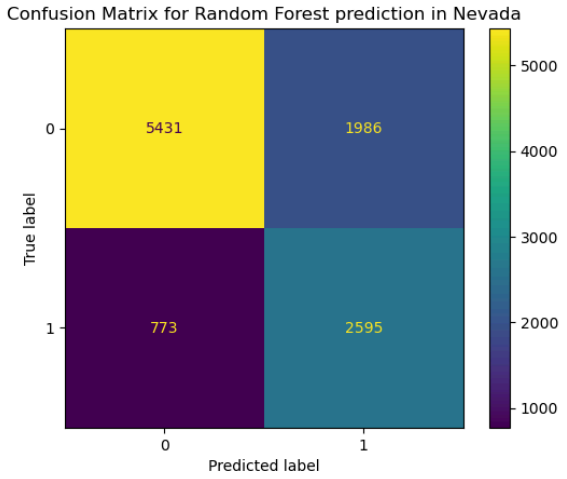
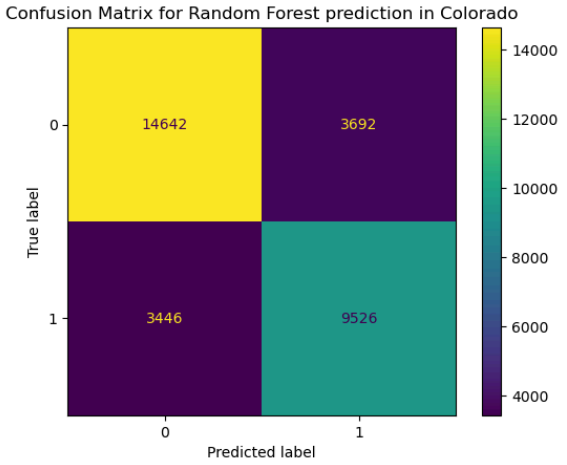

	

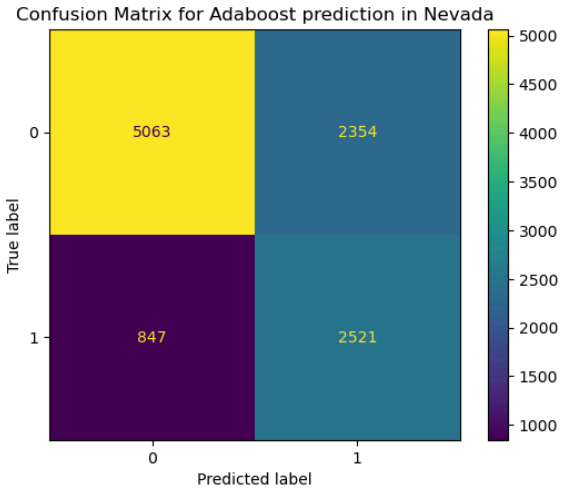
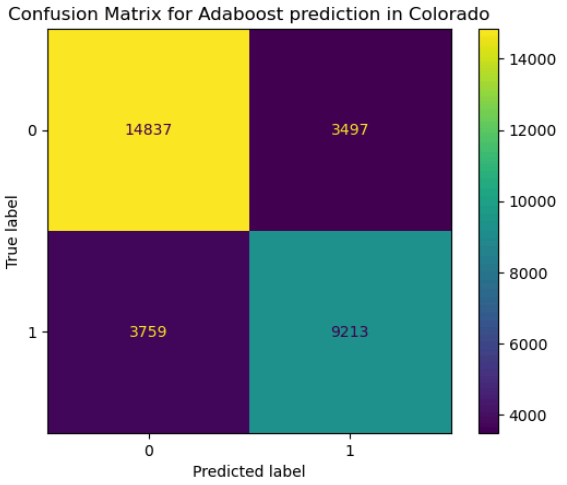
Fig. 44 : Matrice de confusion avec Random Forest	*Fig. 45 : Matrice de confusion avec Random Forest*
	

Fig. 46 : Matrice de confusion avec Adaboost

Fig. 47 : Matrice de confusion avec Adaboost

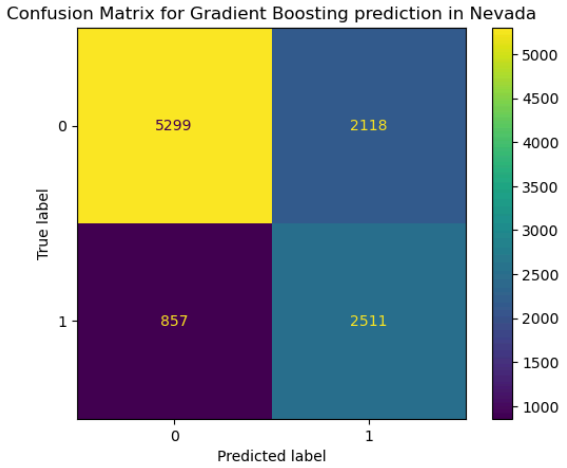


Fig. 48 : Matrice de confusion avec Gradient Boosting

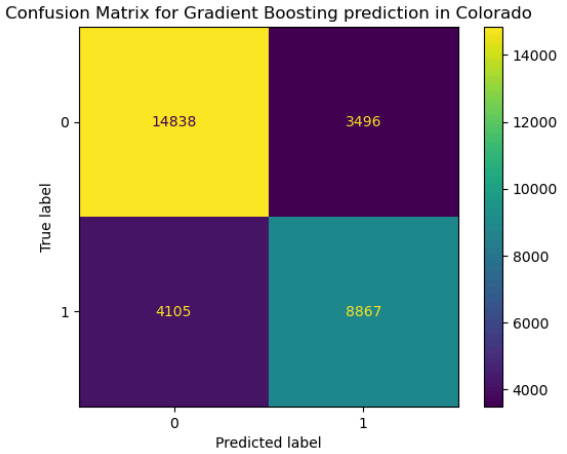


Fig. 49 : Matrice de confusion avec Gradient Boosting

	Nevada				**Colorado**			
Modèles	SVM	RF	AB	GB	SVM	RF	AB	GB
Accuracy (%)	73.77	74.42	70.32	72.42	76.65	77.20	76.82	75.72

Fig. 50 : Tableau représentant les valeurs d'accuracy pour les prédictions des modèles pour le Nevada et le Colorado

Tout d'abord, on remarque que les scores d'accuracy sont globalement bons pour toutes les méthodes et les deux Etats même si les valeurs sont légèrement inférieures aux résultats obtenus avec les données test californiennes. Le modèle qui parvient le mieux à prédire les revenus supérieurs à 50 000 $ à partir d'un entraînement avec des données californiennes est le modèle Random Forest pour le Nevada, et le modèle Adaboost pour le Colorado. Cependant, les résultats pour le Nevada sont moins bons que ceux du Colorado et particulièrement dans la prédiction des positifs comme en attestent les matrices de confusion. En effet, on retrouve sur les *classification reports* des taux très faibles de précision pour les prédictions positives pour le Nevada par rapport au Colorado, représentés dans le tableau suivant :

	Nevada				**Colorado**			
Modèles	SVM	RF	AB	GB	SVM	RF	AB	GB
Précision des prédictions positives(%)	0.56	0.57	0.52	0.54	0.71	0.72	0.76	0.72

Fig. 51 : Tableau représentant les valeurs d'accuracy pour les prédictions positives des modèles pour le Nevada et le Colorado

Cela signifie que seulement un peu plus de 50% des prédictions positives des modèles pour le Nevada concernent des personnes qui sont effectivement rémunérées plus de 50 000 $.
Ce taux est extrêmement faible et dénote l'incapacité de ces modèles à prédire un résultat avec une précision suffisante à partir des données du Nevada. Cela signifie que nos modèles, entraînés avec des données californiennes arrivent à prédire le label convenablement pour des données du Colorado mais pas pour des données du Nevada. Ainsi, il semblerait que la structure socio-économique des populations californiennes et coloradiennes soit suffisamment proche pour que les modèles prédisent bien même en n'étant entraîné uniquement sur les données de Californie alors que ce n'est pas le cas pour le Nevada. Au Colorado et en Californie, la corrélation entre les données représentées dans les *features* de nos jeux de données et notre label est suffisamment forte pour permettre une bonne prédiction. Il semblerait donc qu'au Nevada, les personnes qui reçoivent des revenus supérieurs à 50 000 $ n'ont pas, dans une certaine mesure, le même profil et les mêmes caractéristiques d'âge, de domaine professionnel, de niveau d'éducation qu'en Californie et au Colorado. De plus, il faut noter que la Californie a le plus haut Produit Intérieur Brut par habitant des Etats Américains (100 000 $), le Colorado est le quinzième meilleur Etat (90 000 $) et le Nevada n'est que trente-deuxième (avec 75 000 $ PIB/hab). Ces différences de revenu ne sont qu'un exemple des multiples différences entre ces Etats, qui expliquent les différences que l'on retrouve en exploitant nos prédictions.

V. Explicabilité des modèles

Dans cette partie, nous nous intéressons à la mesure de l'importance des features et au poids de chacune des features pour la prédiction finale.
Pour ces mesures, nous avons travaillé sur 10 % du dataset, un bon rapport entre taille des données et temps d'exécution pour les différentes mesures.

Malheureusement, la version du package seaborn installé dans l'environnement de TP (0.12.0) créé sur Anaconda présente un défaut, les heatmap dont nous nous sommes servis pour afficher les matrices de corrélation n'affichent les données chiffrées que pour la première ligne de la matrice. Nous avons trouvé une trace de ce bug sur le site stackoverflow :
https://stackoverflow.com/questions/77165100/only-the-first-row-of-annotations-displayed-on-seaborn-heatmap

Nous n'avons pas réussi à installer une version plus récente de seaborn sur notre environnement Anaconda, la version 0.12.0 étant la dernière disponible sur notre configuration.

A. Calcul des corrélations

Tout d'abord, nous avons mesuré les corrélations entre les features du dataset et le label réel. Nous obtenons cette matrice de corrélation.

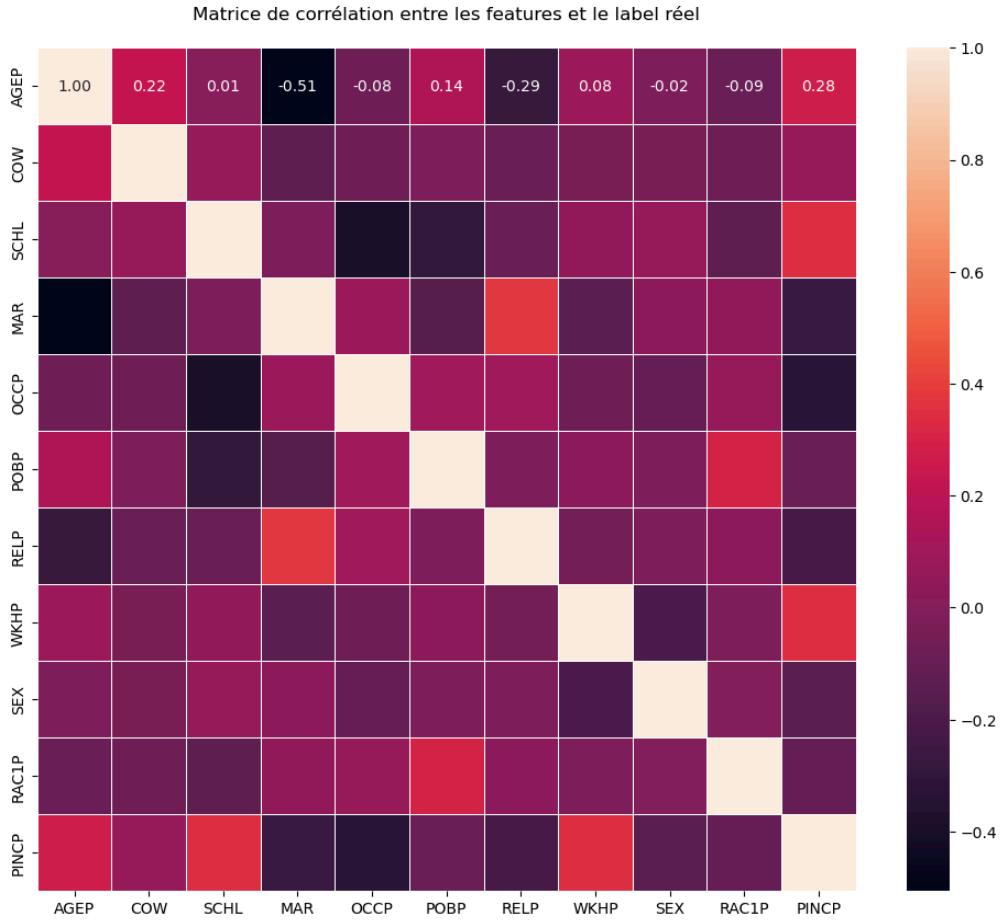


Fig. 52 : Matrice de corrélation entre les features et le label réel

On remarque un lien négatif fort entre l'âge et la situation maritale, ainsi qu'entre l'âge et la relation à la personne responsable du foyer.
Pour ce qui est des liens de corrélation entre les features et le label à prédire, le domaine professionnel (**OCCP**) ainsi que le statut marital (**MAR**) ont un lien négatif important avec le label. Le nombre d'heures travaillées par semaine dans les 12 derniers mois (**WKHP**), le niveau d'éducation (**SCHL**) ainsi que l'âge (**AGEP**) ont une corrélation positive forte avec le label **PINCP**.

Nous avons ensuite mesuré la corrélation pour chacun des modèles entre les features et le label prédit. Nous avons travaillé sur la partie test de nos données, que nous avons standardisées afin de pouvoir réaliser nos prédictions.

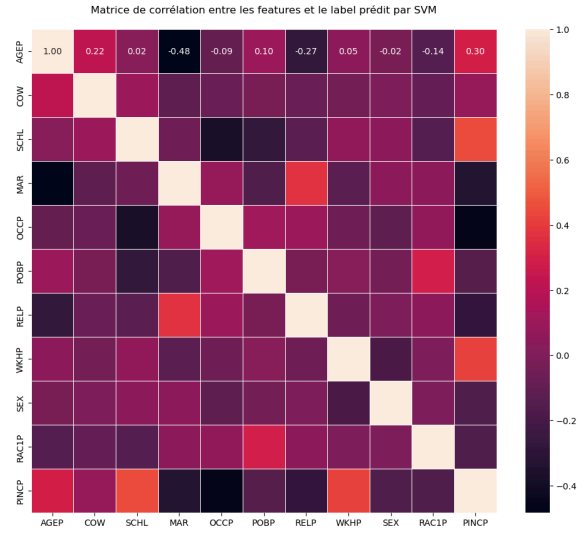


Fig. 53 : Matrice de corrélation entre les features et la prédiction de SVM

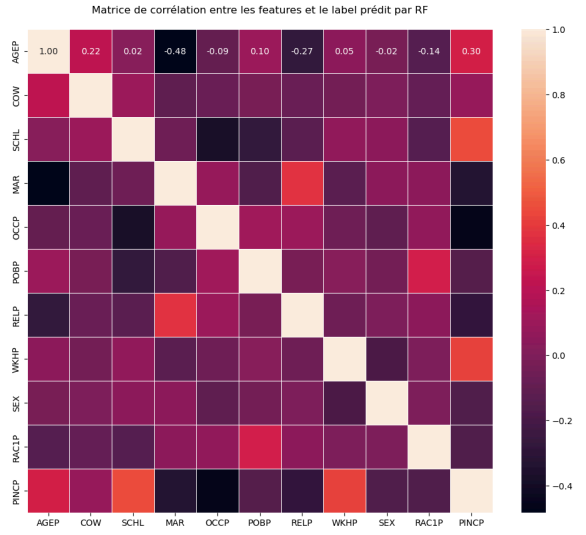


Fig. 54 : Matrice de corrélation entre les features et la prédiction de Random Forest

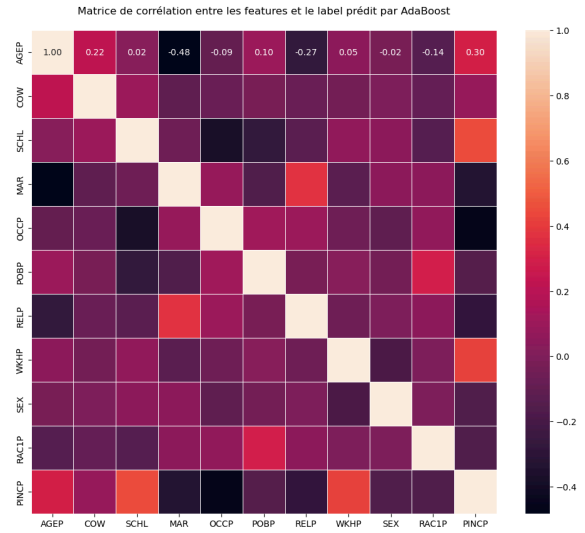


Fig. 55 : Matrice de corrélation entre les features et la prédiction de AdaBoost

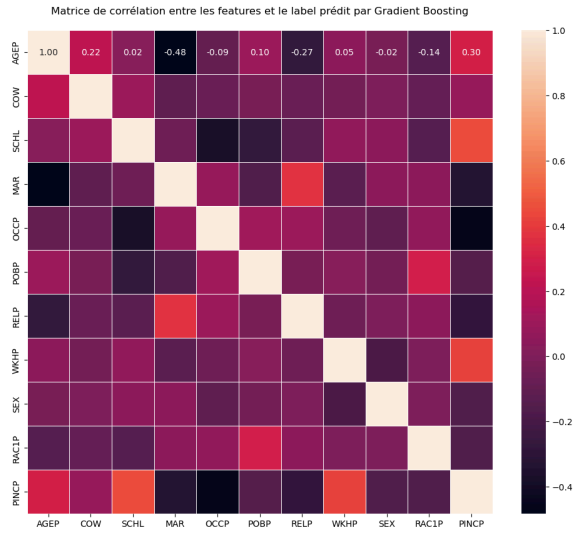


Fig. 56 :Matrice de corrélation entre les features et la prédiction de Gradient Boosting

Nous observons de grandes similitudes entre ces quatre matrices, et l'absence des valeurs chiffrées ne nous permet pas d'analyser une quelconque différence de corrélation selon le modèle.
En revanche, les valeurs de corrélation des features semblent plus importantes avec le label prédit qu'avec le label réel et les données d'entraînement.
Il semblerait donc que les features les plus corrélées au label prédit soient les mêmes que celles pour le label réel, à savoir : le domaine professionnel, le statut marital, le nombre d'heures travaillées par semaine, le niveau d'éducation ainsi que l'âge.

B. Evaluation des features

Nous avons ensuite utilisé la méthode *permutation_importance* de scikit learn afin d'évaluer l'importance de chaque feature. Cette méthode permute chaque feature de notre jeu de données pour quantifier la diminution de la performance du modèle lors de la permutation d'une feature. Cela permet ainsi de mesurer d'une autre façon l'importance de chaque feature.

Cette méthode nous a permis pour chaque modèle de prédiction utilisé de déterminer les features les plus importantes pour la prédiction du salaire. Nous avons ainsi obtenu les graphiques suivants représentant le score d'importance de chaque feature avec la déviation standard pour chaque mesure.

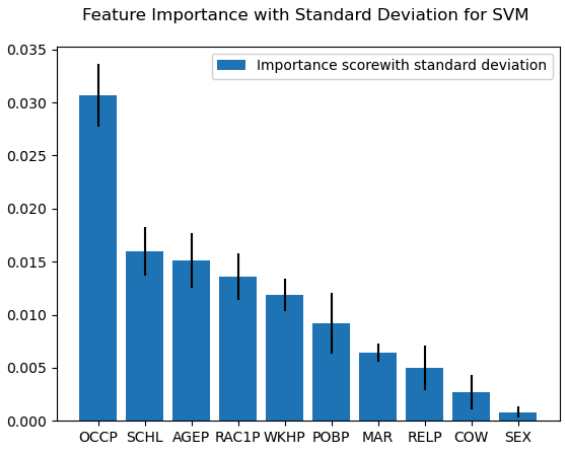


Fig. 57 : Graphique de l'importance des features pour le modèle SVM

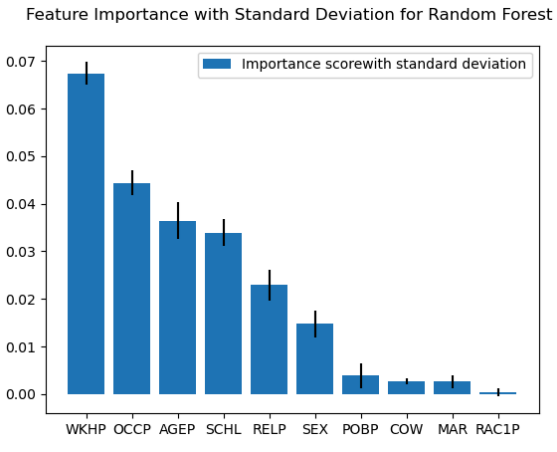


Fig. 58 : Graphique de l'importance des features pour le modèle Random Forest

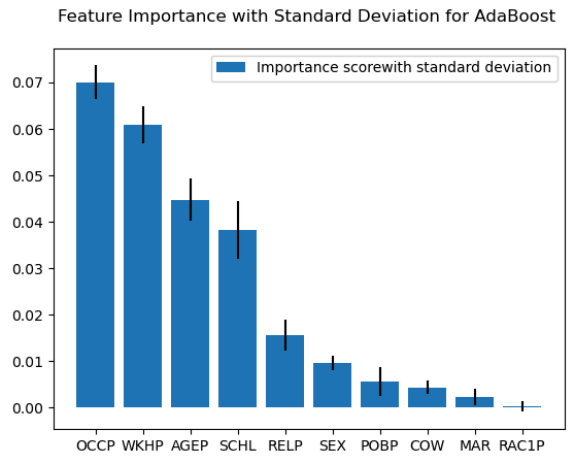


Fig. 59 : Graphique de l'importance des features pour le modèle AdaBoost

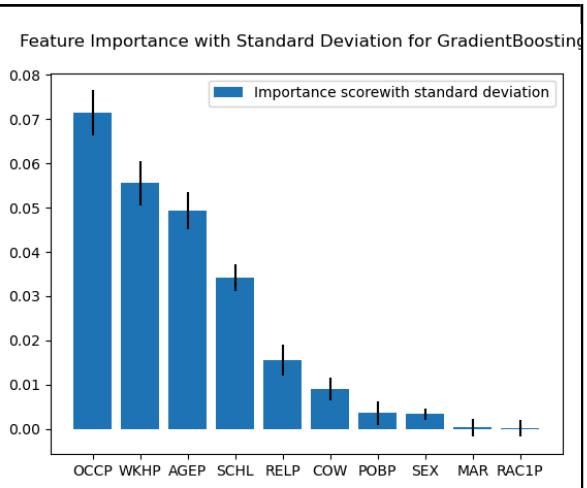


Fig. 60 : Graphique de l'importance des features pour le modèle Gradient Boosting

Tout d'abord, nous observons que le modèle SVM présente les scores de feature importance les plus bas, seul le domaine professionnel dépasse les 3%, la deuxième feature la plus importante, le niveau d'éducation n'étant qu'à 1,5%.

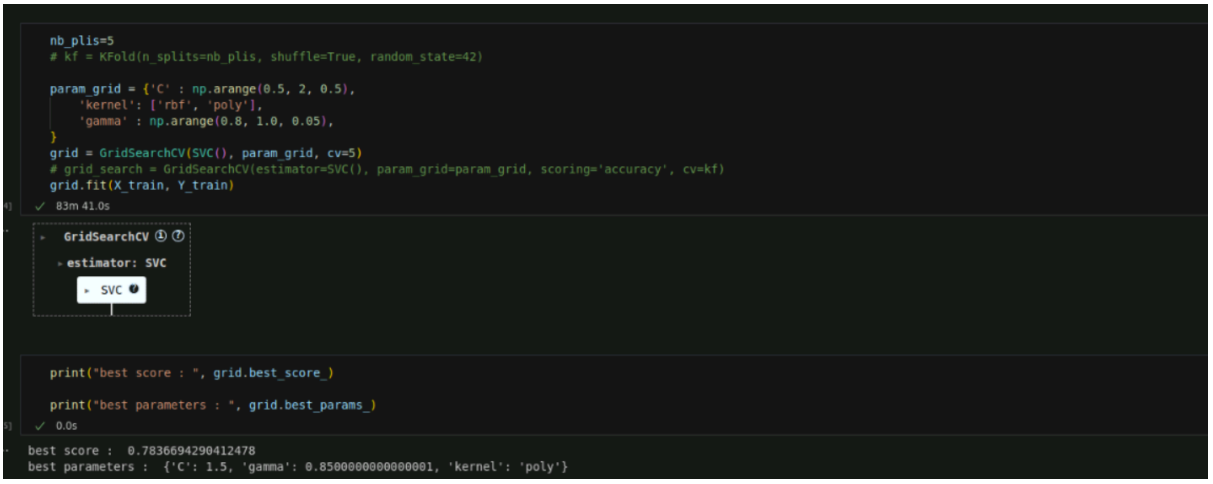
Les trois autres modèles présentent des scores plus importants et une certaine homogénéité dans les valeurs, puisque les quatre features les plus importantes sont le domaine professionnel, le nombre d'heures travaillées par semaine, l'âge et le niveau d'éducation.

SVM		**Random Forest**		**Adaboost**		**Gradient Boosting**	
Feature	*Score*	*Feature*	*Score*	*Feature*	*Score*	*Feature*	*Score*
OCCP	3.96 %	WKHP	6.74 %	OCCP	6.59 %	OCCP	7.14 %
SCHL	1.6 %	OCCP	4.44 %	WKHP	5.75 %	WKHP	5.56 %
AGEP	1.5 %	AGEP	3.64 %	AGEP	4.48 %	AGEP	4.93 %
RAC1P	1.35 %	SCHL	3.39 %	SCHL	3.54 %	SCHL	3.42 %

Fig. 61 : Tableau des scores d'importance pour les 4 features les plus importantes pour chaque modèle

Nous remarquons enfin une importance étonnamment faible des features sur le sexe des personnes et leur ethnie d'origine pour la prédiction du salaire, ce qui est surprenant, ces deux données étant habituellement des composantes importantes dans la répartition des inégalités aux Etats-Unis. Dans le modèle SVM, l'ethnie d'origine est d'ailleurs la 4° plus impactante pour la prédiction du salaire.

Annexe :



Annexe 1 *: Capture d'écran des résultats d'une recherche d'hyperparamètres GridsearchCV pour le modèle SVM sur 5% du dataset.*

Annexe 2 *: Capture d'écran des résultats d'une recherche d'hyperparamètres GridsearchCV pour le modèle Random Forest sur 20% du dataset.*