SORBONNE UNIVERSITÉ

POLYTECH SORBONNE

Projet Apprentissage Statistique

Apprentissage par transfert de modèles pour forêts d'arbres décisionnels

Par :
Achille Baucher
Homer Durand

Encadrant :
Olivier Schwander

Janvier 2020

Contents

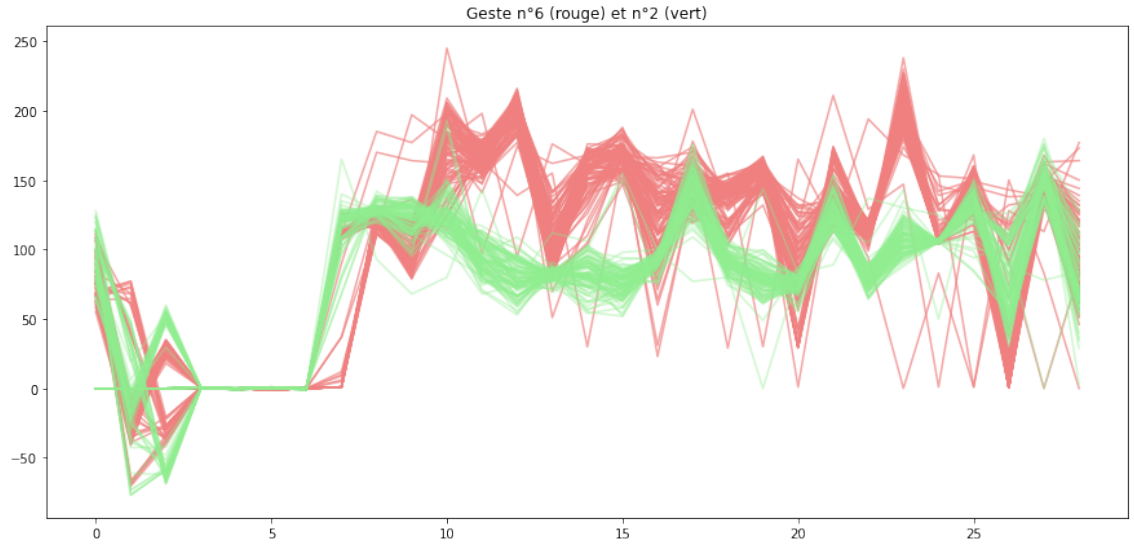

Figure 1: Différence entre les répétitions de deux gestes différents

1 Description et analyse des données

Nous avons analysé les caractéristiques des données afin de comprendre quels algorithmes étaient plus pertinent à appliquer et pourquoi. Le jeu de données utilisé, *UC2017, Static and Dynamic Hand Gestures* [1], est constitué d'un ensemble de gestes statiques et dynamiques de la main. Nous nous concentrerons pour cette étude sur les gestes statiques uniquement. Il y a 24 types de positions (indifféremment appelé *target* ou *labels*) possibles, effectuées plusieurs fois par 8 utilisateurs (*users*) différents. Les positions sont représentées par les variables suivantes:

- 7 premières colonnes : position de la main, 3 translation, 3 rotations, et une variable inconnue qui semble être une variable de position
- 22 dernières colonnes : angles des articulations

Les mesures d'angles sont représentées par des valeurs entières proportionnelles à des angles. Les répétitions de gestes différents suivent des chemins assez distinguables, quoique très variables, comme montré sur la figure 1.

Les variations entre les utilisateurs pour un même type de geste sont visibles, comme montré dans la figure 2, qui représente les positions des mêmes gestes pour 2 utilisateurs différents.

2 Choix de l'algorithme

2.1 Arbre de décision et forêt d'arbres décisionnels

Nous pouvons remarquer sur les images représentant les différents types de gestes que ce sont les angles des articulations (image 9 en annexe) qui vont souvent être les plus cruciaux pour déterminer à quelle catégorie appartient une position. Ils permettent en effet de distinguer

[1] Disponible à l'adresse `https://zenodo.org/record/1319659#.YBmHUtYo-V6`

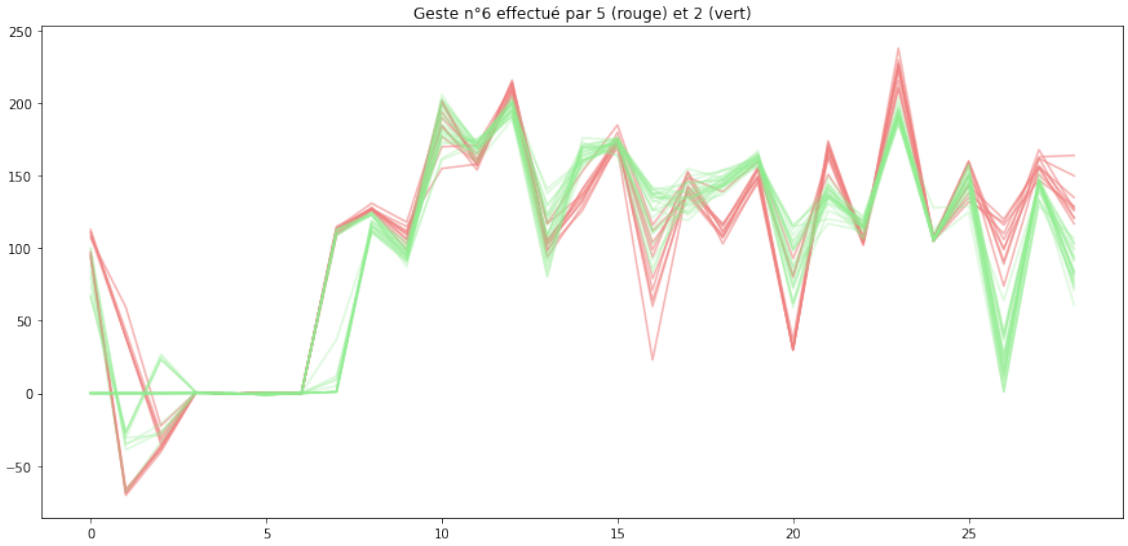

Figure 2: Différence entre les répétitions du même geste par deux utilisateurs différents

quels doigts sont levés et dans quelles directions. Cette intuition est renforcée par la mesure d'importance des variables dans le random forest classifier, qui attribue plus d'importance aux features qui représente des angles [fig 10 et fig 11], même si ce n'est pas le cas pour différencier tous les types de gestes.

Le fait que la majorité des variables importantes soient des valeurs proportionnelles à des angles, nous donne des indications sur ce que doit être capable d'effectuer l'algorithme de classification pour distinguer les gestes. Premièrement, il semble que la capacité à comparer ces angles à des seuils à peu près fixes puisse être importante. Il paraît en effet préférable, pour distinguer un geste, de savoir si un doigt est plié, et à quel point, avec des seuils donc, plutôt que de n'avoir accès qu'à des calculs relatifs entre les angles. Les arbres de décision paraissent de ce point de vue assez appropriés. Ensuite, il apparaît peu intuitif d'effectuer des combinaisons linéaires de mesures d'angles pour connaître la position d'une main. Cette intuition est renforcée par les relativement mauvaises performances du perceptron et des perceptrons multi-couche (12), et c'est pourquoi nous avons laissé de côté les méthodes utilisant des réseaux neuronaux, ne voyant pas quel avantage ce type de calcul pouvait apporter. Par ailleurs les arbres de décision ont une forte tendance à sur-apprendre les données d'apprentissage et le modèle *Random Forest* [1] permet d'éviter cela en rendant ainsi le modèle bien plus performant.

Ces premières considérations, ainsi que la performance des modèles Random Forest et Extremely Randomized Trees [2] par rapport aux autres algorithmes dans nos premiers tests [fig 12], nous ont orienté vers des algorithmes de classification utilisant des arbres de décision.

2.2 Utilisateurs singuliers

Nous avons tenté d'identifier si certains utilisateurs ou certains gestes plombaient particulièrement les résultats du modèle Random Forest. Nous avons pour cela affiché la répartition du taux des erreurs selon l'utilisateur et le type de geste sur la figure 3. Cependant, même s'il y a des différences, il n'apparaît pas que la singularité de certains utilisateurs ou gestes soient à l'origine d'une quantité trop importante d'erreurs.

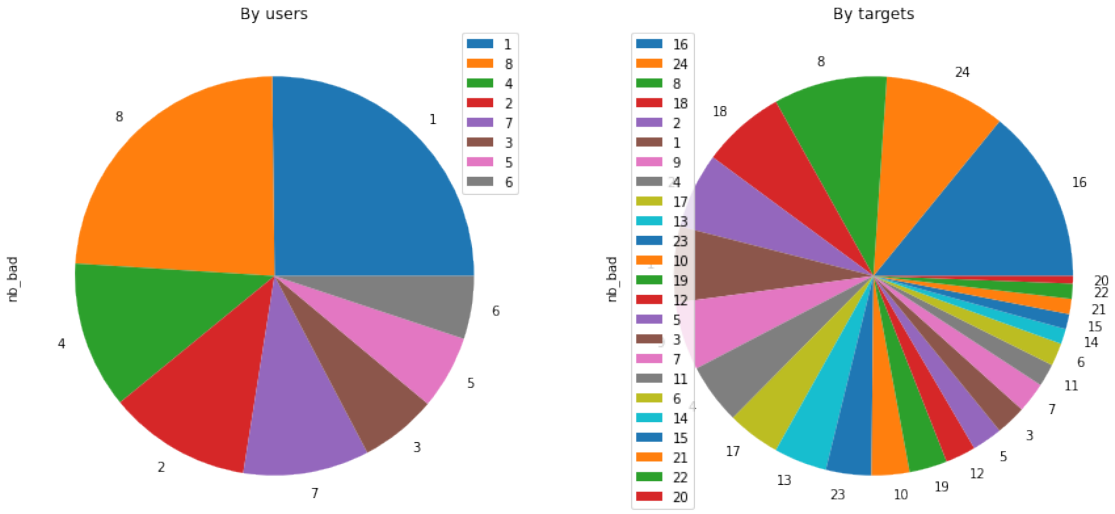

Figure 3: Répartition des erreurs selon les users et les targets

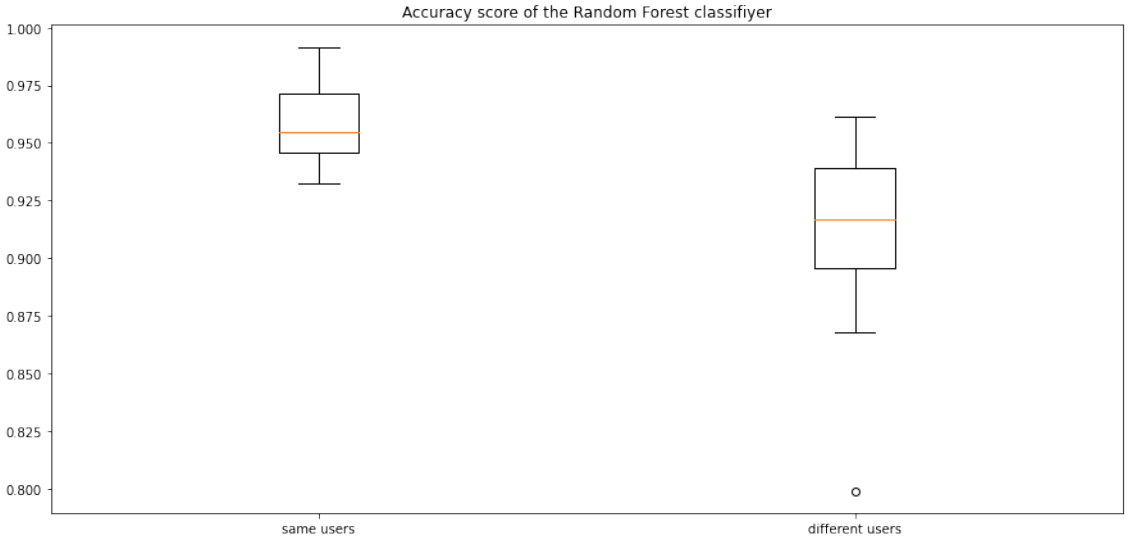

Figure 4: Scores de précision sur des données de validation contenant les mêmes utilisateurs que lors de l'entraînement (à gauche) et des utilisateurs différents (à droite)

2.3 Variation entre utilisateurs

Nous avons voulu vérifier si la connaissance de l'utilisateur était une information cruciale pour classifier ses gestes. Nous avons entraîné le classifieur sur un jeu d'apprentissage ne contenant pas certains utilisateurs, et obtenu les scores de classification pour deux jeux de tests, un contenant les mêmes utilisateurs que dans le jeu d'entraînement, et un autre ne contenant que des gestes d'utilisateurs inconnus. Nous avons répété l'expérience en formant plusieurs jeux d'entraînement éliminant différents utilisateurs à chaque fois, et obtenu les résultats du graphique 4. Nous pouvons remarquer que la présence de l"utilisateur dans le jeu d'entraînement permet d'avoir une bien meilleur score de précision.

Nous avons voulu voir à quel point les variations entre utilisateurs pouvaient être déterminantes pour l'apprentissage. Nous avons, pour chaque utilisateur, entraîné et testé le random forest seulement sur des données de cet utilisateur. Les résultats montrés dans la figure [fig 5] sont

	1	2	3	4	5	6	7	8
Cible	0.848	0.96	0.978	0.975	0.987	0.983	0.917	0.992
Source	0.907	0.94	0.974	0.980	0.961	0.991	0.965	0.994

Figure 5: Comparaison score source only sible only, cross validation, k = 10, n_estimators = 50

comparés aux précisions obtenues sur les mêmes données de test et avec un random forest entraîné sur tous les utilisateurs. Pour certains utilisateurs, on peut voir que le random forest est plus précis en étant entraîné seulement sur les données de l'utilisateur, malgré le fait qu'il dispose d'environ 8 fois moins de données d'entraînement. Nous avons interprété ce résultat comme indiquant que les variations entre utilisateurs sont tellement importantes qu'il est parfois préférable de ne pas apprendre les gestes des autres pour être précis sur un utilisateur. Cependant, le jeu de donnée d'entraînement est beaucoup plus petit si on ne se concentre que sur un seul utilisateur. De plus, la connaissance des gestes des autres utilisateurs est tout de même utile puisqu'elle permet parfois d'obtenir un meilleur score.

2.4 Apprentissage par transfert

Les résultats présentés dans la section précédente nous ont orienté vers un modèle d'apprentissage par transfert. Les deux étapes de cette méthode semblent répondre aux problèmes soulevés:

- **Apprendre sur tout le jeu de donnée:** Cela permet de disposer d'un maximum de données d'entraînement et d'éviter le sur-apprentissage. Les résultats montrés dans la figure 4 montrent qu'il existe suffisamment de similarité entre les gestes d'utilisateurs différents pour que l'entraînement sur certains d'entre eux apportent une connaissance sur les autres.
- **Affiner le classifieur sur un utilisateur spécifique:** À cause des très grandes variations entre utilisateurs, nos conclusions en 2.3 paraissent rendre pertinent de ré-entraîner le classifieur sur l'utilisateur dont nous voulons classifier les gestes.

Ce modèle nécessite de disposer d'un minimum de données sur l'utilisateur dont nous voulons classifier les gestes.

Par ailleurs, il nous semble que l'apprentissage par transfert est pertinent dans le cadre de la reconnaissance de gestes et de positions de mains. Considérons par exemple l'application à l'interaction homme-robot proposé dans [5]. Si la collaboration entre le robot et un utilisateur est régulière il peut être pertinent de chercher à étalonner le classifieur afin que ses prédictions soient plus précises pour cet utilisateur. De manière plus générale, dès que le classifieur est très majoritairement utilisé sur un utilisateur spécifique, il semble important de chercher à ce que celui-ci soit le plus précis possible pour cet utilisateur.

Enfin, le problème de l'apprentissage par transfert de modèles de forêts d'arbres décisionnels nous a semblé étonnamment peu étudié dans la littérature au regard des nombreux avantages qu'offrent ce type de modèle (performance, temps de calcul, interprétabilité et explicabilité [3] et [1]). Il nous semble alors intéressant de s'y attarder et de déterminer des méthodes efficaces et généralisables.

3 Méthodologie

3.1 Formalisation du problème d'apprentissage par transfert

On définit un domaine par $\mathcal{D} = \{\mathcal{X}, \mathcal{P}\}$ avec $\mathcal{X} \subset \mathbb{R}^d$ l'espace des observations et $\mathcal{P}$ une loi de distribution. On notera $X = \{x_1, x_2, ..., x_n\} \in \mathcal{X}$ notre jeu d'observation. Pour un domaine donné, on associe une tâche $\mathcal{T} = \{\mathcal{Y}, f(.)\}$ où $\mathcal{Y} \subset \mathbb{N}$ est l'espace des classes et $f(.)$ est une fonction de prédiction que l'on détermine à partir d'un ensemble d'observation et de classes associés. Dans le cadre de l'apprentissage par transfert [6], nous considérons d'une part un domaine source $\mathcal{D}_\mathcal{S}$ et une tâche associée $\mathcal{T}_\mathcal{S}$ et d'autre part un domaine cible $\mathcal{D}_\mathcal{C}$ et une tâche associée $\mathcal{T}_\mathcal{C}$. Nous faisons l'hypothèse que les domaines sources et cibles sont différent, plus particulièrement nous considérons pour ce projet le cadre de l'apprentissage par transfert homogène pour lequel les espaces des observations et des classes sources et cibles sont les même ($\mathcal{X}_\mathcal{S} = X_C$ et $\mathcal{Y}_\mathcal{S} = \mathcal{Y}_\mathcal{C}$) mais en revanche on considérera que les loi de distribution $\mathcal{P}_\mathcal{S}$ et $\mathcal{P}_\mathcal{C}$ peuvent différer. Nous cherchons à améliorer la fonction de prédiction cible $f_C(.)$ en utilisant le domaine source $\mathcal{D}_\mathcal{S}$ et sa tâche associée $\mathcal{T}_\mathcal{S}$.

3.2 Algorithme de transfert pour forêts d'arbres décisionnels

Pour les raison que nous avons précédemment citées, la fonction de prédiction source $f_T(.)$ que nous allons utiliser et une forêt d'arbres décisionnels (Random Forest [1]). Comme nous l'avons expliqué précédemment, il ne semble pas y avoir de méthode "état de l'art" concernant l'apprentissage par transfert de modèles de forêts d'arbres décisionnels. Nous proposons donc d'étudier trois algorithmes proposées dans [4], structure transfer (STRUT), structure expansion/reduction (SER) et une approche mixte des deux algorithmes précédents (MIX). Nous donnons une rapide description de ces algorithmes dans la section qui suit ainsi qu'une explication intuitive de leur pertinence. Nous vous invitons à vous reporter à l'article susmentionné pour une description plus détaillée.

3.2.1 STRUT : Algorithme de transfert de la structure d'un arbre décisionnel

L'idée intuitive derrière cet algorithme est qu'un arbre de décision entraîné pour des problèmes similaires devrait avoir une structure similaire. Nous cherchons donc à garder la structure des arbres décisionnels entraînés sur les couples $\{(x_1, y_1), ..., (x_n, y_n)\} \in \mathcal{X}_\mathcal{S} \times \mathcal{Y}_\mathcal{S}$. Pour cela nous parcourons chaque arbre de décision de haut en bas (de la racine aux feuilles) en cherchant pour chaque noeud à déterminer un seuil optimal pour l'ensemble d'apprentissage cible. Afin de garder la structure de l'arbre entraîné sur les données sources, nous cherchons un seuil τ_C tel que les distributions des labels Q'_G et Q'_R induites pour les noeuds gauche et droit soient le plus proches possibles des distributions initialement induites par le jeu de données source Q_G et Q_D. On cherchera également à ce que le seuil τ_C soit un maximum local du gain d'information qui dépendra du critère utilisé (gini, entropy). Pour chaque noeud v on détermine donc un nouveau seuil τ_C pour le feature ϕ_C et le jeu d'apprentissage cible atteignant ce noeud S_v^Ctel

que

$$\begin{aligned} &\text{Max}_{\ x} DG(S_v^C, \phi_C, x, Q_L, Q_R) \\ &s.c. \quad x \in \mathbb{R} \\ &\qquad \forall x' \in [x-\epsilon; x+\epsilon] \quad \forall \epsilon > 0 \\ &\qquad IG(S_v^C, \phi_C, x') \leq IG(S_v^C, \phi_C, x) \end{aligned}$$

où $DG(S_v^C, \phi_C, \tau, Q_L, Q_R)$ est le gain de divergence qui évalue les proximités respective des distributions Q'_G et Q'_R et des distributions Q_G et Q_R et $IG(S_v^C, \phi_C, \tau)$ est le gain d'information induit par le seuil τ pour le critère utilisé.

3.2.2 SER : Algorithme d'expansion/réduction de structure d'arbres décisionnels

L'algorithme structure expansion/reduction propose d'effectuer deux transformations locales sur les feuilles des arbres décisionnels entraînés sur les données sources. La première, l'expansion, cherche à spécialiser l'arbre entraîné sur les données sources aux données cibles. Pour cela on détermine pour chaque feuille f l'ensemble des données S_f^C qui atteignent cette feuille puis on créée un nouvel arbre décisionnel à partir de ces données et on le raccorde à la feuille f. La seconde transformation, la réduction, cherche à généraliser l'arbre induit par l'étape d'expansion. Pour cela on élague l'arbre de bas en haut au regard de deux types d'erreurs. La première, la *subtree error*, est l'erreur empirique induite pour chaque noeud v de l'arbre pour le jeu de données cible atteignant ce noeud S_v^C. La seconde, la *leaf error* est l'erreur induite pour chaque noeud v s'il devait être élagué en feuille. Si la *leaf error* est inférieure ou égale à la *subtree error* on élague l'arbre en transformant ce noeud en feuille.

3.2.3 MIX : Approche mixte

L'auteur propose une approche mixte (MIX) qui consiste simplement à générer deux forêts par les algorithmes STRUT et SER et de procéder à un vote sur l'ensemble des arbres décisionnels ainsi créés afin de déterminer la classe prédite par l'algorithme MIX.

3.3 Évaluation des modèles

Nous évaluerons les performances des différents modèles au regard de trois mesures de performance : Le score F1, la variance inter-utilisateur et le minimum de performance.

- **Score F1**

Nous choisissons pour mesure de la performance globale des modèles le score F1. Cela peut sembler étonnant étant donné que les classes sont parfaitement équilibrées pour l'ensemble des targets (200 exemples pour chaque position). Ce n'est en revanche pas nécessairement le cas dans le jeu de test lors de la validation croisée. Certains utilisateurs ayant un nombre limité de chacune des positions (*targets*), cela peut conduire a un fort déséquilibre dans la distributions des targets dans le jeu de test pour ces utilisateurs.

- **Variance de performance inter-utilisateurs**

Lors de l'évaluation des résultats nous voulons être en mesure de sélectionner un modèle qui nous assure un certain seuil de performance face à un nouvel utilisateur. Si on compare deux fonctions de prédiction ayant le même score F1 moyen nous préférerons celle dont les scores F1 moyens par utilisateur sont les moins dispersés autour de la moyenne. Nous introduisons pour cela l'indice de dispersion des performances $V_{inter-user}$:

$$V_{inter-user} = \frac{1}{|U|+1} \sum_{u \in \mathcal{U}} (\bar{S}^{(u)} - \bar{S})^2$$

avec $\bar{S}^{(u)}$ le score moyen pour la classe $u \in \mathcal{U}$ (l'ensemble des utilisateur) et $\bar{S}$ le score moyen pour l'ensemble des données de test.

- **Minimum de performance**

Afin de déterminer si le modèle évalué peut conduire à un score F1 trop faible en fonction de l'utilisateur, il nous semble pertinent d'observer le score F1 moyen minimal pour l'ensemble des utilisateurs. Cela étant un bon indicateur de la robustesse du classifieur face à de nouveaux utilisateurs.

- **Sur-apprentissage**

Par ailleurs nous analyserons les modèles au regards du sur-apprentissage qu'ils entraînent en comparant les performances sur le jeu d'apprentissage et celles sur le jeu de test.

- **Durées d'apprentissage**

Nous évaluerons également les durées d'apprentissage des différents algorithmes. L'évaluation des durées de prédiction ne nous semble pas pertinente a étudié car la transformation des forêts par les différents algorithmes proposés ne les fait pas varier.

4 Résultats

Dans cette section nous allons évaluer de manière empirique la performance des différents algorithmes présentés précédemment, à savoir STRUT, SER et MIX. Nous allons pour cela les comparer à une forêt aléatoire entraînée uniquement sur les données sources (*source only*) et à une autre entraînée uniquement sur les données cible (*target only*).

4.1 Performances

Nous observons [fig 14] que même si le score moyen de la forêt entraînées sur les données cibles (*target only*) est supérieur au score moyen de la forêt entraînée sur les données sources, la très grande dispersion des scores par utilisateur en fait un modèle peu intéressant. Cela supporte l'idée qu'un apprentissage par transfert est pertinent pour ce jeu de données afin d'obtenir un modèle précis et robuste.

Comme nous nous y attendions, le modèle *STRUT* et plus performant que le modèle *SER*. Cela s'explique très bien par le type de données considérées ici. Il est raisonnable de penser

	targets only	sources only	SER	STRUT	MIX
mean f1 score	0.917857	0.885269	0.885085	0.917277	**0.934203**
variance inter users	0.003217	0.001428	0.001219	0.002156	**0.000977**
minimum user score	0.805556	0.826296	0.824923	0.816853	**0.872424**

Figure 6: Comparaison des algorithmes STRUT, SER et MIX aux modèles de référence *targets only* et *sources only* - Validation croisée, k = 10, n_estimators = 50

que les différents utilisateurs ayant des tailles de mains différentes auront des angles légèrement différents les uns part rapport aux autres pour une même position d'un doigt. L'algorithme *STRUT* est justement performant dans ce type de situation car il va alors affiner les seuils de chaque noeud de l'arbre afin que ceux-ci correspondent au mieux à l'utilisateur concerné par les données cibles.

En revanche nous ne nous attendions pas à ce que l'algorithme *MIX* surpasse [fig 6] sur chacun des critères l'ensemble des autres algorithmes et des modèles de référence. Cela est d'autant plus étonnant que l'algorithme *SER* est peu performant, l'algorithme *MIX* ne fait donc pas qu'additionner les gains de performance respectif des modèles *SER* et *STRUT*. L'auteur de [4] donne une explication à ce phénomène. Un arbre transformé par l'algorithme *SER* aura certainement une structure éloignée de celle de l'arbre initial contrairement à l'arbre transformé par l'algorithme *STRUT*. On obtient ainsi pour chaque arbre entraîné sur les données sources deux nouveaux arbres ayant une faible corrélation et donc une forêt ayant une plus grande diversité.

4.2 Sur-apprentissage

Les modèles de forêts d'arbres décisionnels ayant été entraîné sur une très une faible quantité de données sont fortement sujet au sur-apprentissage. En revanche dans le cadre d'apprentissage par transfert, les données cible montre parfois une structure éloignée des données sources et la fonction de prédiction entraînée sur les données sources peut se retrouver dans un cas de sous-apprentissage pour les données cibles. Comme nous l'observons [fig 7] le modèle entraîné sur les données cibles souffre fortement de sur-apprentissage. De la même manière l'algorithme *STRUT* souffre du même problème, la faible quantité de données pour chaque utilisateur ne semble pas suffisante pour que celui-ci puisse bien généraliser à de nouvel données d'un même utilisateur. On remarque en revanche que l'algorithme *SER* sur-apprend peu des données, ce modèle est plus robuste. Cela vient certainement du fait qu'il ne modifie pas les seuils des noeuds en spécialisant trop fortement l'arbre aux données d'apprentissage. Enfin l'algorithme *MIX* montre un fort sur-apprentissage même si bien moins important que celui de l'algorithme *STRUT*. On a donc ici une intuition de la raison pour laquelle l'ajout des arbres transformés par l'algorithme *SER* – qui a eux seuls n'apporte pas de gain de performance – à ceux de l'algorithme *STRUT* permette un gain de performance substantiel, les arbres de *SER* permettraient de régulariser la forêt d'arbres décisionnels et ainsi d'éviter un trop fort sur-apprentissage. C'est d'ailleurs ce que propose l'auteur de l'article [4].

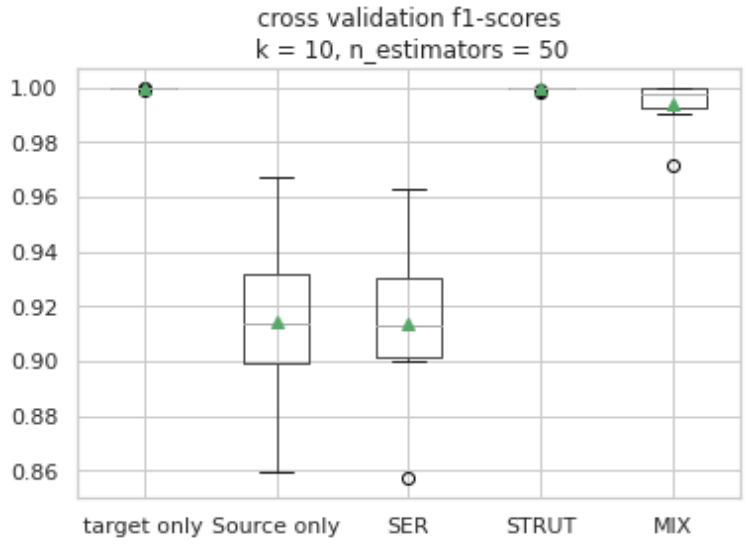

Figure 7: Score F1 sur données d'apprentissage cible

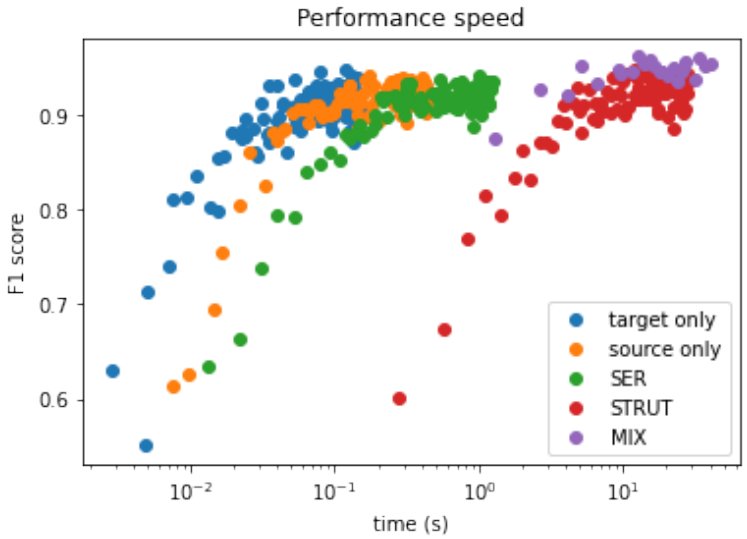

Figure 8: Score F1 par rapport au temps d'apprentissage (échelle logarithmique)

4.3 Vitesse de performance

Les performances (accuracy, score F1, etc..) des forêts d'arbres décisionnels [1] dépendent jusqu'à un certain seuil du nombre d'arbres composant cette forêt [fig 16] car ceux-ci permettent d'éviter le sur-apprentissage. Cependant les durées d'apprentissage et de prédiction augmente linéairement avec l'augmentation du nombre d'arbre. Il est donc important de trouver un juste compromis entre durée d'apprentissage et performance du modèle. Aussi les performances touche un plafond aux alentours de 50 arbres [fig 16] il ne semble donc pas nécessaire d'aller au delà. Nous proposons donc d'observer la durée d'apprentissage de chacun des algorithmes et modèles de référence que nous avons jusque là analysé afin de déterminer si le sur-coût en temps lié à la transformation des arbres par les algorithmes *STRUT*, *SER* et *MIX* comporte un intérêt. Particulièrement nous voulons déterminer si le gain de performance apporté par l'algorithme *MIX* peut être remplacer par une simple augmentation du nombre d'arbre dans une forêt entraîné sur les données source ou cible uniquement.

La durée d'apprentissage lié à l'éxécution de l'algorithme *STRUT* (et donc celle de *MIX*) est très largement supérieur (d'un facteur 100) à celle de l'algorithme *SER* et des modèles de référence *source only* et *target only* [fig 8]. Cela vient en parti du grand nombre de problème d'optimisation non convexe qu'elle résout pour chaque noeud de chaque arbre. En revanche, si le gain de performance de l'algorithme *STRUT* n'apporte pas grand chose celui de *MIX* est intéressant et ce modèle garde sa pertinence tout en gardant à l'esprit qu'il fait exploser la durée d'apprentissage. Par ailleurs l'apprentissage de forêts aléatoires, de même que la transformation de celles-ci par les algorithmes *STRUT, SER* et *MIX*, est une tâche hautement parallélisable. La durée de la transformation d'une forêt par *MIX* pourrait donc être diminuée avec le nombre de CPUs utilisés ou par l'utilisation de GPUs.

5 Discussion

Nous avons proposé dans ce projet d'étudier le le jeu de données *C2017, Static and Dynamic Hand Gestures* comme un problème d'apprentissage par transfert en utilisant comme fonction de prédiction source une forêt d'arbres décisionnels. Cela découle des différents constats cité dans la section [1]. Le choix d'utiliser les forêts d'arbres décisionnels s'explique par le fait que celles-ci semblent performantes pour ce jeu de données (plus performantes que les modèles du type réseaux de neurones). Nous expliquons cela par le fait qu'il ne semble à priori pas y avoir d'intérêt à calculer des combinaisons linéaires entre différents angles mais que classer ceux-ci avec différents seuils semble par contre pertinent. Nous avons ensuite décidé d'étudier ce jeu de données avec un problème d'apprentissage par transfert car nous avons remarquer que l'apprentissage sur les données cibles uniquement (*source only*) donnait parfois de très bons résultats mais était peu robuste (grande variabilité de performance inter-utilisateurs) et que l'apprentissage sur les données sources donnait des résultats relativement robuste mais pas toujours très précis. Il nous a alors semblé pertinent de chercher à combiné les deux pour obtenir de très bonnes performances quand cela était possible tout en conservant une faible variabilité de performance inter-utilisateurs.

Nous avons pour cela proposé d'étudier trois algorithmes proposés dans [4], à savoir *SER, STRUT* et *MIX*. Nous avons pu observer dans l'évaluation de ces modèles que l'algorithme *MIX* montrait des performances intéressantes au regard des trois indicateurs principaux que nous avons observé (score F1 moyen, variance de performance inter-utilisateurs et score F1 minimal). Il a en effet pu surpasser les performances des modèles de référence (*source only* et *target only*) sur chacun de ces indicateurs. Cet algorithme fait en revanche exploser la durée d'apprentissage ce qui peut rendre son utilisation compliqué dans certain cadre. Nous envisageons que ce problème de durée d'apprentissage puisse être réduit en envisageant de meilleurs heuristiques pour résoudre le problème d'optimisation non-convexe de l'algorithme *STRUT*. Ce problème peut également être traité dans une certaine mesure par l'utilisation de plusieurs CPUs et/ou de GPUs afin de transformer ces arbres de manière parallèle.

Nous tenons à noter que nous n'avons pas comparer ces algorithmes à d'autres méthodes d'apprentissage par transfert de modèle ni à d'autre méthode de transfert de connaissance. Cela principalement par manque de temps. Nous n'avançons donc pas que ces algorithmes sont les plus performants pour résoudre le problème d'apprentissage par transfert que nous nous sommes posé. Nous avons d'ailleurs remarqué [fig 17] que le modèle *Extremely randomized trees* ([2]) permettait de régulariser les forêt entraînées sur les données cibles et d'avoir ainsi de très bonnes performances pour *source only*. Les algorithmes *SER, STRUT* et *MIX* s'adaptent

d'ailleurs bien aux modèles *Extremely randomized trees* même si nous n'avons pas eu le temps de développer plus cet axe.

Nous aurions donc, avec plus de temps, cherché à comparer ces algorithmes à d'autres méthodes d'apprentissage par transfert de modèle (Adapative SVM par exemple) et de transfert de connaissance de manière plus générale. Nous aurions également étudié plus amplement l'adaptation des algorithmes *SER, STRUT* et *MIX* aux forêts d'arbres décisionnels engendré par l'algorithme *Extremely Randomized Trees*.

6 Bibliographie

References

[1] Breiman. Random Forest. 2001.

[2] Pierre Geurts, Damien Ernst, and Louis Wehenkel. *Mach Learn (): DOI 10.1007/s10994-006-6226-1 Extremely randomized trees*. 2006.

[3] Christoph Molnar. *4.4 Decision Tree | Interpretable Machine Learning*.

[4] Noam Segev. Transfer Learning Using Decision Forests. 2016.

[5] Gibaru Simao, Neto. Online Recognition of Incomplete Gesture Data to Interface Collaborative Robots. 2019.

[6] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *arXiv:1911.02685 [cs, stat]*, June 2020. arXiv: 1911.02685.

7 Annexes

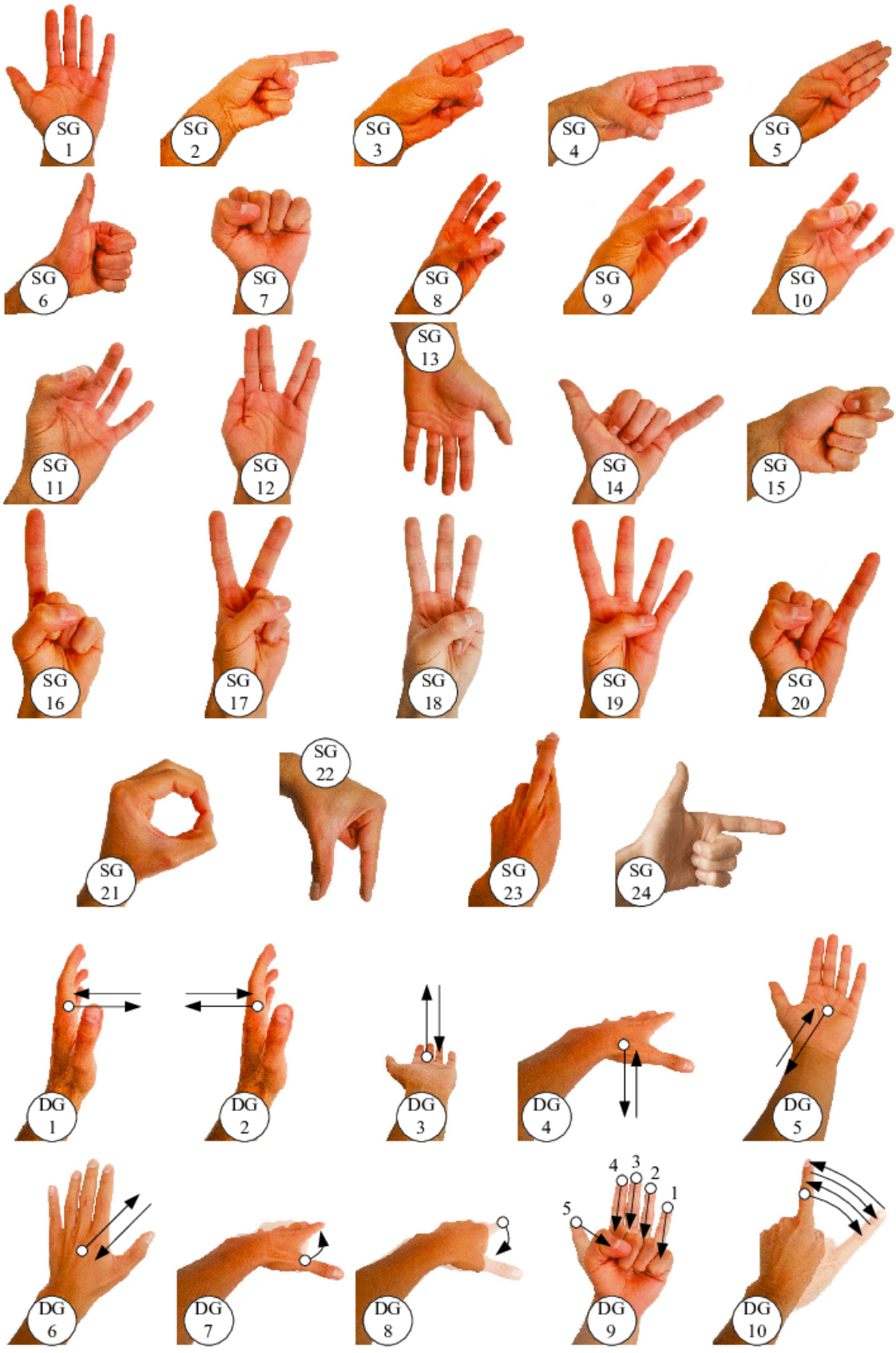

Figure 9: Les différents gestes de la main du jeux static gesture, tiré de l'article [5]

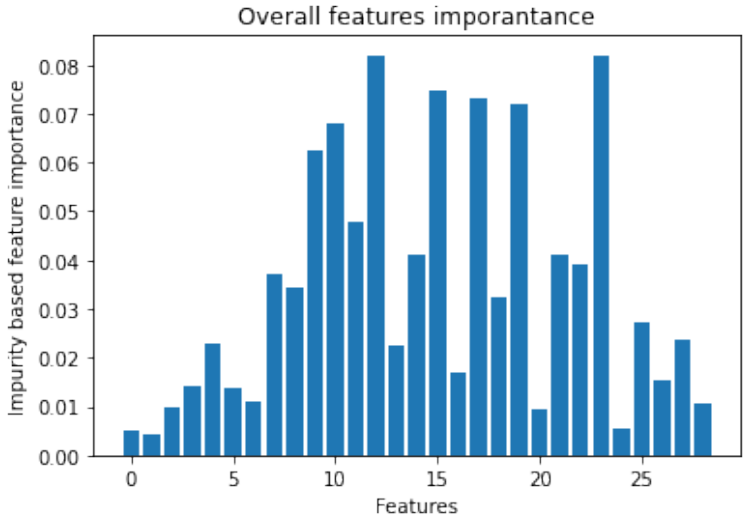

Figure 10: Importance de chaque features par rapport au taux d'impureté

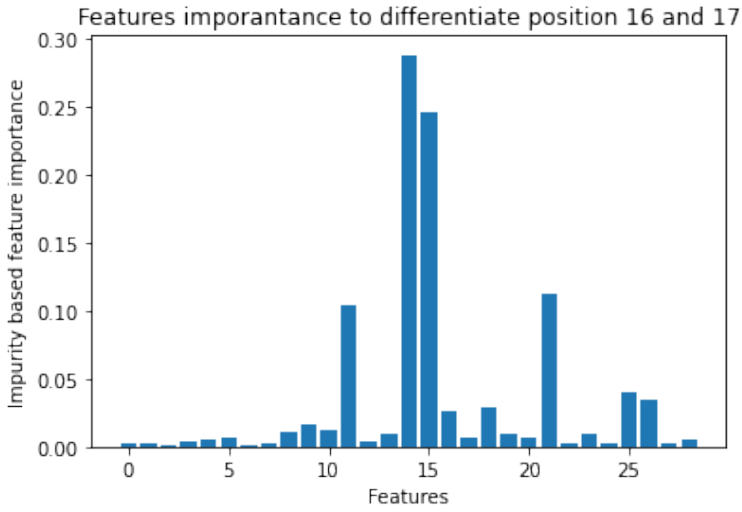

Figure 11: Importance des features pour différencier les positions 16 et 17

	Fitting time	Predict time	CV score
Random Forest	0.672354	0.017398	0.940833
Extra Trees	0.368724	0.019526	0.944583
Gradient Boosting	33.147049	0.008432	0.899167
SVM	0.131729	0.045308	0.913750
Multilayers Perceptron	4.090093	0.001672	0.850417
Logistique regression	0.576416	0.000704	0.880833
KNN	0.001265	0.032276	0.903750
Perceptron	0.072657	0.000489	0.740000

Figure 12: Durées d'apprentissage, durées d'entrainement et accuracies de différents classifieurs - validation croisée, k=10

	CV all users	CV new users
Random Forest	0.944583	0.914361
Extra Trees	0.948333	0.923861
Gradient Boosting	0.897917	0.816881
SVM	0.913750	0.890837
Multilayers Perceptron	0.852500	0.734529
Logistique regression	0.880833	0.813732
KNN	0.903750	0.863208
Perceptron	0.740000	0.713582

Figure 13: Comparaison de l'accuracy de différents classifieurs sur données avec utilisateurs inconnus et sans utilisateurs inconnus

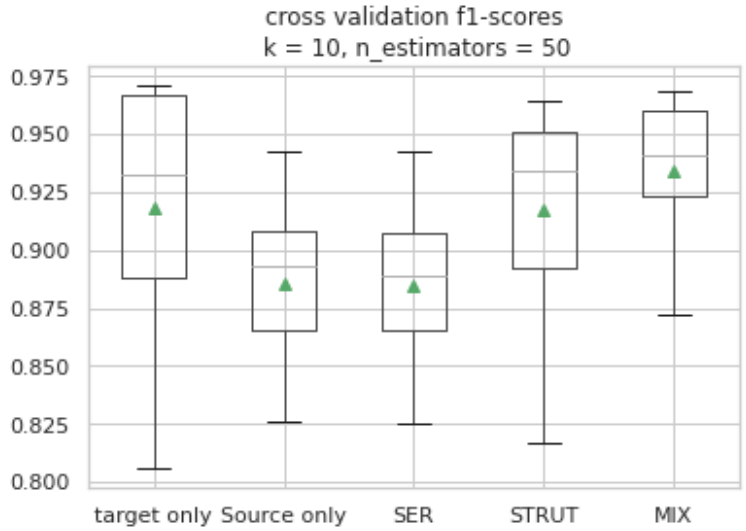

Figure 14: Performances des modèles pour l'ensemble des utilisateurs avec Random Forest

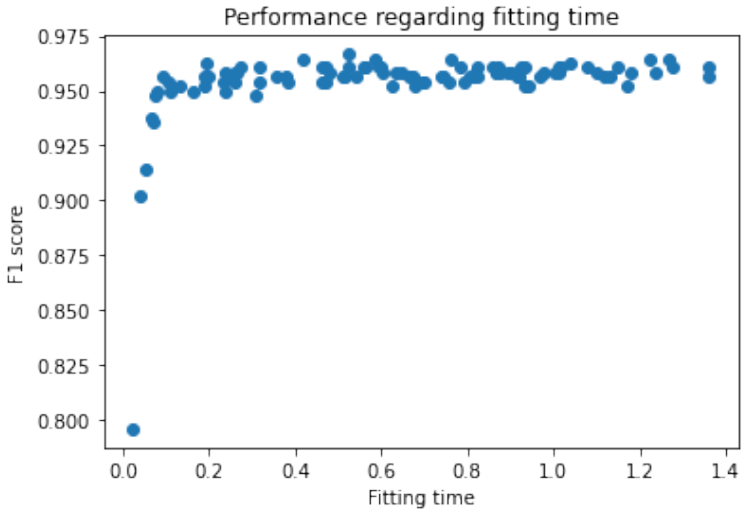

Figure 15: Performances en fonction de la durée d'apprentissage

Figure 16: Performances en fonction du nombre d'arbre

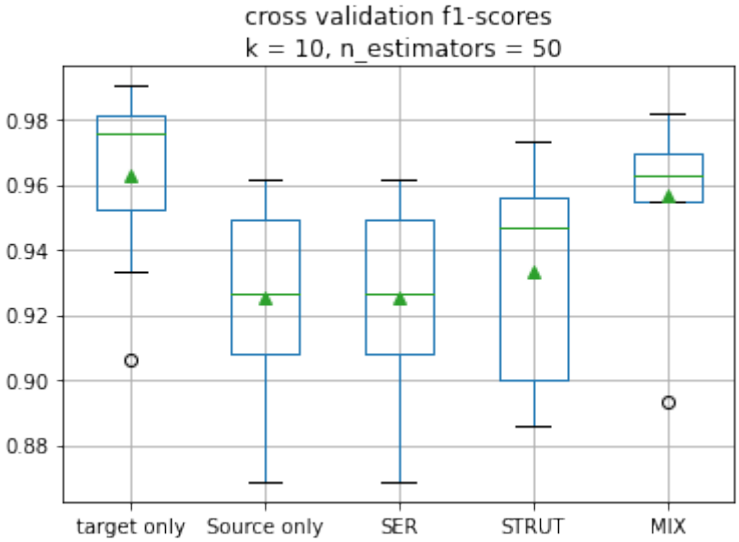

Figure 17: Performances des modèles pour l'ensemble des utilisateurs avec Extra Trees