INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

INP TOULOUSE ENSEEIHT

Projet Fil Rouge

Machine Learning

Auteurs:
Baptiste AUSSEL
Albin CINTAS

Juin 2021

Sommaire

Introduction

L'objectif de ce projet est de modéliser un prédicteur d'activité humaine à partir de signaux provenant d'un capteur de smartphone. 6 activités sont développées :

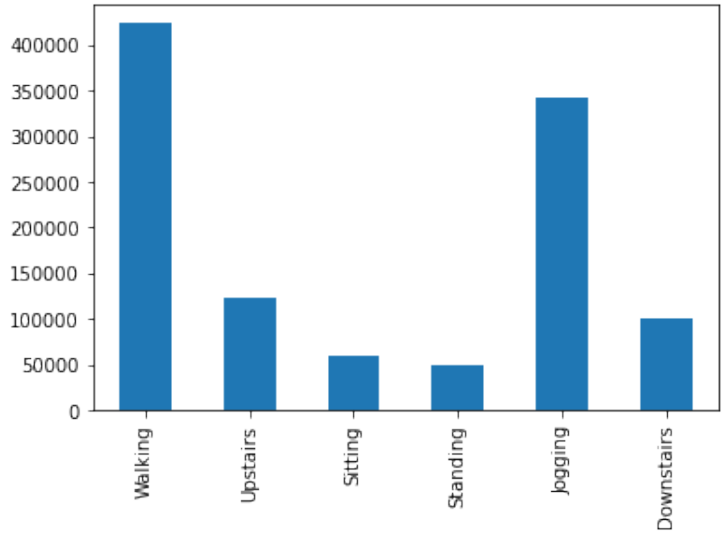
- Descendre
- Courir
- Assis
- Debout
- Monter
- Marcher

Les capteurs des téléphones offrent donc des signaux sur un temps limité, qui ont été enregistrés à une fréquence de 50Hz. . Ces signaux se décomposent sur 3 axes : x, y et z. Ils sont également tous accompagnés d'une étiquette sur laquelle figure l'activité associée au signal.
Ce genre de prédicteur peut être utilisé pour de multiples applications telles que la robotique, la surveillance de personnes agées, les suivis médicaux, etc...
Notre objectif au long de ce projet est d'experimenter plusieurs modèles de machine learning (regréssion, boosting, méthodes à noyaux, réseaux de neurones, etc...) dans le but de les comparer et d'ainsi pouvoir en extraire le plus performant en termes de rapidité et de score. Nous essaierons d'optimiser les paramètres des algorithmes et de les tester sur des données de différentes tailles afin d'obtenir les meilleures performances possibles.
Dans le début de cette étude, nous n'allons considérer que 3 classes et un nombre réduit de données. Nous étudierons plus tard des modèles avec davantage de données et de classes.

1 Exploration des données

1.1 Analyse quantitative des données

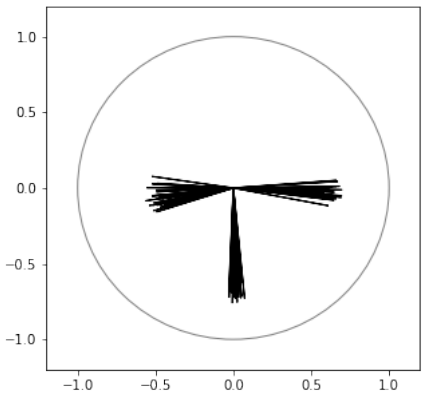
Le jeu de données est composé de : 1098203 individus, répartis de la manière suivante :



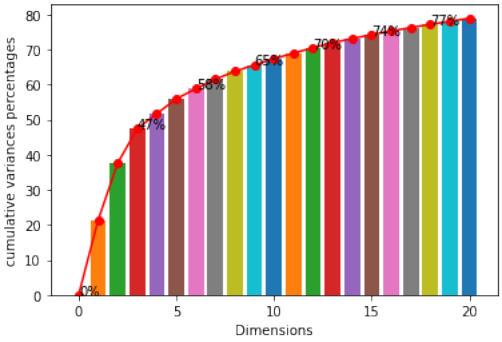
On peut donc constater dans un premier temps que les données ne sont pas réparties de manière totalement équilibré. En effet, presque 75% des données sont réparties dans les catégories Walking et Jogging. Il faudra donc en prendre compte dans la suite du projet.
On remarque de plus que les mesures ont été faites sur un faible nombre d'individus, 36 plus précisément. Ceci peut en réalité biaisé en pratique car les caractéristiques physiques (vitesse de marche, taille, etc...) peuvent être visibles.

1.2 ACP

On procède à une ACP afin d'étudier les variables de nos individus. On remarque très distinctement que trois classes apparaissent, correspondant à nos trois axes : x, y et z.



De plus, les variances cumulatives nous montrent qu'avec 15 composantes principales, on peut arriver à avoir 75% de l'information (sachant qu'on a commencé à 90 variables) :



2 Les premiers modèles

Nous avons commencé par étudié 3 classe: assis, jogging et descente d'escalier. Nous avons effectué un Kmeans pour voir s'il pouvait bien fitter avec notre jeu de donnée:

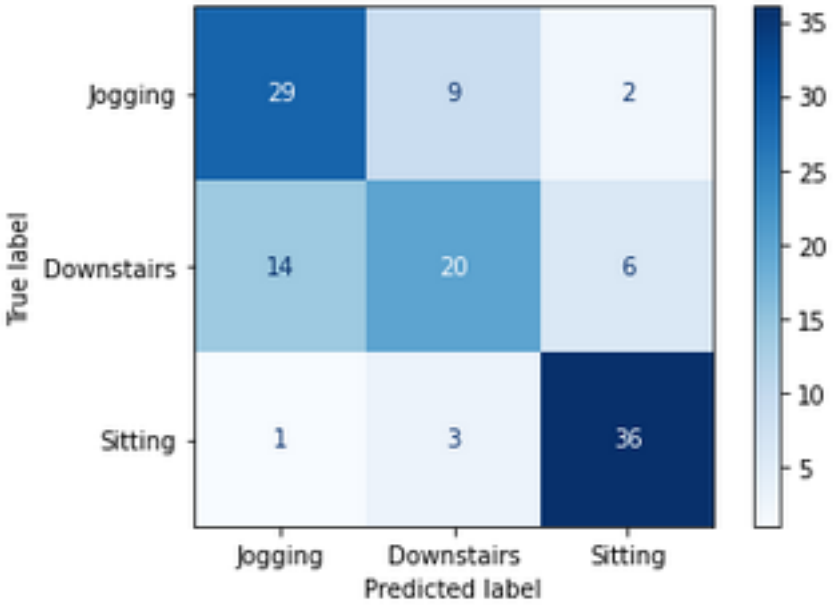
```
[[127  19   4]
 [ 96  35  19]
 [ 12  59  79]]

Out[16]: 0.5355555555555556
```

2.1 Kmeans

Comme on pouvait s'y attendre Kmeans n'est pas très efficace, nous avons donc essayé la régression multiclasse.

```
score :  0.7083333333333334
```

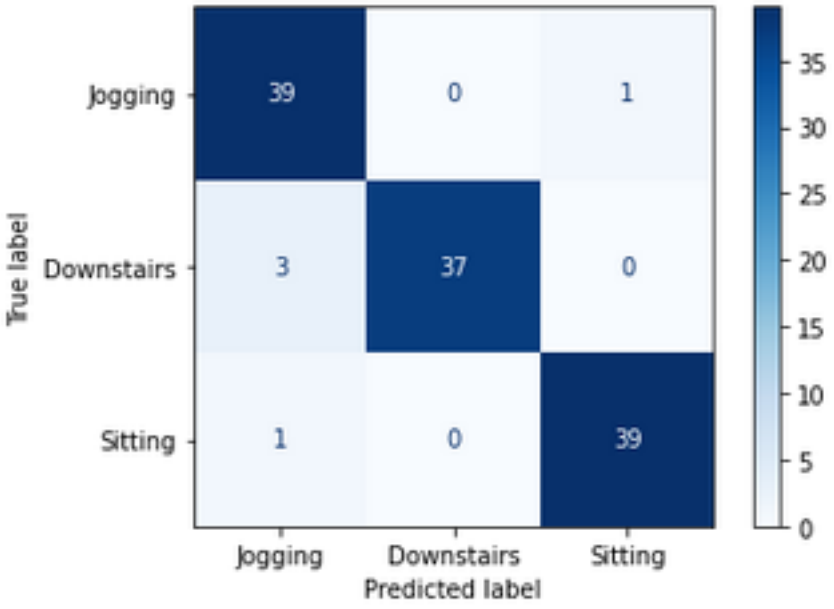


2.2 Regression multi classe

Nos résultats sont mieux mais ce n'est toujours pas incroyable. C'est pourquoi nous allons essayé de faire une classification par SVM à noyau.

2.3 SVM

```
score de svm optimisé :  0.9583333333333334
```



On obtient ici un super résultat avec très peu de données. C'est très puissant et c'est notre meilleur algorithme pour le moment. Nous allons continuer d'essayer d'autres classifieurs.

2.4 Boosting

L'utilisation d'Adaboost combiné a un Arbre de décision nous permet d'obtenir un score de 91 % et 94 % pour une forêt aléatoire.

Une méthode de gradient boosting nous donne elle un score de 94 % et la cross validation suivante [0.91,.94,0.93,0.91,0.9]. Cette méthode semble donner des résultats similaires à Adaboost. Enfin nous avons tester xgboost qui a donné un score de 89 % ce qui est moins bon que les 2 autres méthodes de boosting.

2.5 Classification par Forêts aléatoires

Une foret alétoire donne, une fois optimisé, un score de 94 % et un score out of bag de 92.5 %

2.6 Réseaux de neurones

Un RN de sk learn donne un score de 0.875 alors que celui de keras donne 0.87 ce qui est a peu près équivalent.

2.7 Résume

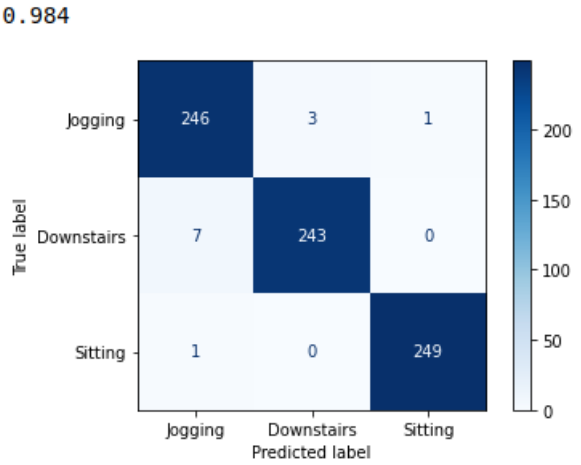
En résumé notre meilleur classifieur reste SVM et les méthodes de boosting donnent aussi de bons résultats.

3 Augmentation des données

Nous avons décidé pour cette partie de laisser tomber les algorithmes qui n'étaient pas efficaces sur la partie précédente, c'est-à-dire ceux qui donnait des résultats de moins de 80%. Ici, nous allons analyser les conséquences d'une augmentation du nombre de données considéré, du nombre de classes considéré et du temps de chaque segment temporel.

3.1 Augmentation du nombre de données

Dans cette partie, nous passons de 400 individus pour l'ensemble d'apprentissage, à 1000 individus. La proportion pour l'ensemble de test reste la même.
SVM : La SVM nous donne avec l'augmentation de données, les résultats suivants :

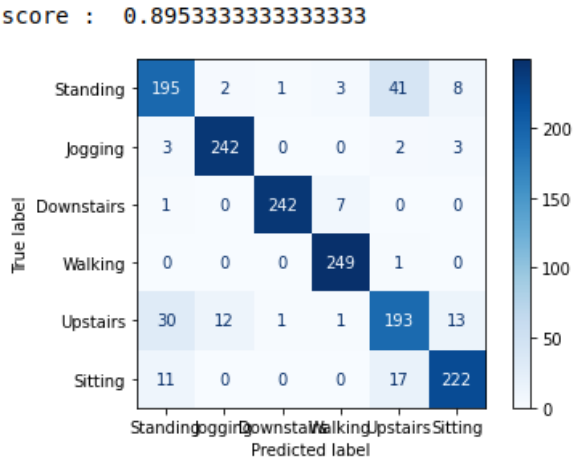


Ce résultat est alors notre meilleur score, pour un temps d'execution très faible en plus.
Classification spectrale : trop long.
Adaboost : Avec un temps d'execution moyen, Adaboost optimisé (composé d'une forêt aléatoire) donne un score de 98%. Ce score concurrence la SVM testée précedemment.
Gradient boosting : En ce qu'il s'agit de cette méthode de boosting, une cross-validation nous donne des résultats compris entre 97% et 98%. Cette méthode vient donc également concurrencer Adaboost et SVM.
XGBoost : XGBoost nous donne un résultat de 97%. Il s'inscrit également dans les algorithmes très performants. D'autant plus qu'il met peu de temps à tourner également.
Forêt aléatoire : Lorsqu'elle est optimisée, la forêt aléatoire donne un score de 97%. Encore un algorithme très efficace avec tant de données.
Réseaux de neurones : Nous avons repris le même réseau de neurones que précedemment. Celui-ci nous offre un score de 95%, ce qui est légerement moins bien que les algorithmes précédemment cités.

3.2 Augmentation du nombre de classes

Dans cette partie, nous allons ajouter 3 classes (activités) à notre problème. On se retrouve alors avec une prédiction de 6 classes.

SVM : La SVM est de nouveau assez rapide a s'exécuter. Elle offre les résultats suivants :



Nous avons ici presque 90% de réussite sur autant de données que précemment, mais avec 6 classes.

Adaboost : Après optimisation, cet algorithme nous donne un résultat de 90% de réussite sur l'ensemble test. Il est comme précédemment environ équivalent a la SVM. Cependant il reste plus long en temps d'execution.

Gradient Boosting : une cross-validation nous donne des résultats compris entre 88% et 91%. Cependant, le temps d'execution était assez long.

XGBoost : XGBoost donne un score de 90% également. D'autant plus qu'il tourne très rapidemment.

Forêt aléatoire : Après optimisation, la forêt aléatoire donne un score de 89%. Il est alors équivalent aux précédents.

Réseau de neurones : Le meilleur réseau de neurones donne un résultat très rapidemment de 85%. Il est de nouveau un peu moins bon que les précédents classifieurs.

3.3 Augmentation du temps des segments

: Dans cette partie nous avons essayé d'augmenter le temps des sequences temporelles afin d'envisager une meilleure classification. Après avoir fait tourner les mêmes modèles que précédemment, on s'est rendus compte que les résultats étaient quasiment tous inférieurs aux précédents sans augmentation temporelle. Et de plus, ce choix a engendré une augmentation du nombre de variables, et donc des temps d'execution des algorithmes.

3.4 Résumé

: Pour conclure cette partie, on a vu qu'augmenter le nombre de données permettait d'avoir des résultats excellents, avec quelques algorithmes de manière très rapide. De plus,

l'augmentation du nombre de classes à bien évidemment réduit les performances des algorithmes. Mais nous avons quand même réussi à atteindre un score de 90%. Enfin, nous avons réalisé qu'augmenter le temps des signaux n'apportait rien aux performances de nos algorithmes.

4 Analyse fréquentielle

Dans cette partie, nous rabaissons les données comme lors de notre première étude. L'objectif de cette partie est de comparer toute l'étude faite précédemment avec des segments temporels, par une étude fréquentielle. Pour ce faire, nous avons remplacé le dataset précédent par un dataset composé des transformées de fourier des signaux.
SVM : Avec seulement 400 données d'apprentissage, la SVM optimisée atteint le score de 99%, ce qui prouve que la transformée de Fourier a eu de l'effet sur l'apprentissage.
Adaboost : Un score de 98% mais un temps encore long.
Gradient Boosting : Algorithme lent également, mais score de 98% comme Adaboost.
XGBoost : Tourne rapidemment et donne un résultat de 97%.
Forêt aléatoire : Met un temps moyen à tourner et donne un score de 98%.
Réseau de neurones : 97% pour celui de sklearn.

On se rend compte que l'application de la transformée de Fourier au dataset a permis d'augmenter les performances de notre algorithme. Cela s'explique surement par le fait que plus d'information sur l'activité se trouvait dans la fréquence.

5 Conclusion

Tableau récapitulatif :

sizes \ models	SVM	Classification spectrale	Adaboost	Gradient boosting	XGBoost	Forêt aléatoire	MLP sklearn	MLP keras
Data_App = 150 Label_App = 40 N_CLASSES = 3 SEGMENT_TIME_SIZE = 30	0.95	trop long	0.94	0.93	0.89	0.94	0.875	0.87
Data_App = 1000 Label_App = 250 N_CLASSES = 3 SEGMENT_TIME_SIZE = 30	0.98	trop long	0.98	0.975	0.97	0.97	0.95	0.94
Data_App = 1000 Label_App = 250 N_CLASSES = 6 SEGMENT_TIME_SIZE = 30	0.895	trop long	0.9	0.89	0.9	0.89	0.85	0.68
Data_App = 700 Label_App = 150 N_CLASSES = 3 SEGMENT_TIME_SIZE = 60	0.89	trop long	0.87	0.9	0.87	0.89	0.87	0.67
Analyse fréquentielle avec : Data_App = 400 Label_App = 150 N_CLASSES = 3 SEGMENT_TIME_SIZE = 30	0.99	trop long	0.98	0.98	0.97	0.98	0.97	0.64

Pour conclure,