# MLBD Project : Simplified Human Activity Recognition w/Smartphone

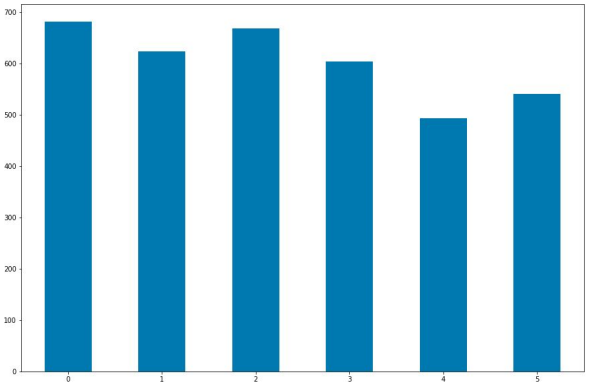
Thibaut Piquerez & Sylvain Renaud
HES-SO MSE - Machine Learning on Big Data
25.06.2019

## 1. General context and objectives of the project

Ce projet est réalisé dans le cadre du cours “Machine Learning on Big Data” du Master HES-SO. Le projet a pour but de mettre en pratique les éléments vus durant le semestre et peut être réalisé par groupe de 2 personne, ce qui est le cas ici. Le choix du sujet est libre et s’est porté, dans notre cas, sur la reconnaissance d’activités sur la base des capteurs de téléphone portable. Le projet est implémenté dans un notebook Jupyter et la bibliothèque scikit-learn est utilisée.
Le but est donc de prédire l’activité en cours grâce aux données des capteurs d’un téléphone. Les activités possible sont *debout*, *assis*, *couché*, *marche*, *monte les escaliers*, *descends les escaliers*.
La reconnaissance d’activité peut avoir plusieurs utilisation. Par exemple, créer une application permettant de faire un suivi des activités quotidiennes d’une personne, où elle peut visualiser le temps qu’elle a passée assise, couchée ou à marcher. Par des calculs il est alors possible de déterminer le nombre de calories brûlée ou les kilomètres parcouru. On peut imaginer lancer des alertes quand la personne est assise depuis trop longtemps pour lui dire d’aller se dégourdir les jambes, ce qui la maintiendra en meilleure forme physique.
Une autre idée serait la surveillance de personnes dans un EMS, un hôpital ou un autre établissement du genre. L’application préviendrait alors le personnel médical quand une personne peu apte à se déplacer seule se lève de son lit.

## 2. Database description

Le dataset utilisé dans le cadre du projet vient du site internet Kaggle et se nomme “Simplified Human Activity Recognition w/Smartphone”. Accessible avec le lien suivant : https://www.kaggle.com/mboaglio/simplifiedhuarus. Deux fichiers au format CSV sont disponible mais un seul a été utilisé (train.csv) car le deuxième (test.csv) contient uniquement les features, sans les labels, ce qui n’est pas utile dans notre cas. Le fichier train contient 3609 exemples avec 561 features, ces features correspondent aux différentes mesures des capteurs du téléphone, ainsi que des statistiques sur ces valeurs telles que la moyenne, la déviation standard et différents coefficients. Les capteurs utilisées sont les 3 axes de l’accéléromètre et les 3 axes du gyroscope. Les labels sont au nombre de 6 : STANDING, SITTING, LAYING, WALKING, WALKING_DOWNSTAIRS et WALKING_UPSTAIRS. Comme le montre le graphique ci-contre, le dataset est assez bien balancé donc pas de problème de ce côté là.




## 3. Data pre-processing & feature extraction

Pour la préparation des données, le fichier *train.csv* est simplement chargé et séparé en deux parties correspondantes aux labels et aux features. La colonne contenant les identifiants est supprimé car elle n’est pas pertinente. Ensuite le dataset est séparé en “train” et “test” set. Le test set comprends environ ⅓ des entrées et le train set en compte environ ⅔ . Aucune normalisation est réalisé car toutes les valeurs sont déjà comprise entre -1 et 1 et aucun nettoyage des données n’a été requis. Les label ont été transformé à l’aide d’un LabelEncoder.

Ce projet n’a pas nécessité une phase de feature extraction. Cependant, comme les features sont nombreuses (561), nous avons essayé deux techniques de feature selection basé sur les statistiques, afin de voir si il était possible de réduire le nombre de feature à traiter dans nos modèles sans perte de précision.

La première méthode a été de retirer les features ayant une faible variance. Cela a été réalisé avec `from sklearn.feature_selection import VarianceThreshold.` Le seuil a été défini à 0.2, donc toutes les features présentant une variance inférieure seront éliminées. Seules 59 features sont gardées avec cette technique, on enregistre leurs indices dans une liste d’indices, en plus de la liste contenant tous les indices.

Le seconde méthode est basé sur le test statistiques du $\chi^2$. On choisit de garder uniquement les 150 meilleures features. Les données sont dans l’intervalle [-1;1] mais le test du $\chi^2$ nécessite des valeurs positives. Nous avons alors ajouter 1 à toutes les données de X et ainsi transformer l’intervalle en [0;2]. Les indices des 150 meilleures features sont ajoutés à la liste des indices globales, qui contient maintenant tous les indices, les indices de la première feature selection et les indices de la deuxième feature selection.

## 4. ML technics

Plusieures techniques de machine learning sont utilisées pour prédire les labels, aussi bien des techniques supervisées que non-supervisées. Pour chaques algorithmes supervisés, de la cross-validation ainsi qu'un "grid search" ont été utilisé à l'aide de la fonction "GridSearchCV" de scikit-learn. Pour la partie non-supervisé, nous avons juste voulu voir si il est possible de faire du clustering sur ces données là. Les techniques utilisés sont des algorithmes classiques de machine learning, que l'on a vu au cours.

Le premier algorithme supervisé utilisé est KNN (K nearest neighbors) de l'implémentation de scikit-learn (KNeighborsClassifier). Afin de tester plusieurs paramètres du modèle, nous avons effectués un "grid search" sur les paramètres suivant : Le premier paramètre est **n_neighbors**, qui représente le nombre de voisin jusqu'aux quels il faut calculer la distance, ensuite on affecte le point au cluster représenté le plus souvent par les voisins. Les valeurs sont 3, 5 ou 7. Le second paramètre est **p** qui définit la puissance de la métrique Minkowski, donc si p=1, c'est la distance Manhattan, si p=2, c'est la distance euclidienne. Les valeurs sont : 1 et 2.
Le modèle est entraîné et testé sur toutes les features, puis seulement les features obtenues par la méthode 1 de features selection, puis seulement les features obtenues par la méthode 2.

Le deuxième algorithme supervisé est basé sur les réseaux de neurones, c'est le MLP (pour multi-layer perceptron). Un grid search a également été effectué, avec les paramètres suivant: **solver** qui sera toujours lbfgs, **alpha** qui prend les valeurs 1e-3, 1e-4 ou 1e-6, **hidden_layer_sizes** qui prend les valeurs (5,10), (7,14) ou (3,6), **max_iter** qui prend les valeurs 250, 300 ou 400, et finalement **activation**, la fonction d'activation à utiliser qui prends les valeurs tanh, relu ou logistic.

Le troisième algorithme supervisé implémenté est le "Support vector machines" (SVM) offert par scikit-learn. Comme pour les autres, un "grid search" a été appliqué à l'algorithme pour 3 les paramètres suivants : la valeur de **C**, le type de **kernel** et la **fonction de décision**. Pour la fonction de décision, il y a 2 valeur : "ovo" pour one-vs-one et "ovr" pour one-vs-rest. L'algorithme a été testé avec les 3 kernels suivants : linear, rbf et sigmoid. Pour la valeur de C, voici les valeurs appliquées : 0.1, 0.5, 1.0, 2.0, 10.0, 50.0, 100.0, 200.0, 500.0, 1000.0. Cette liste de valeurs a été sélectionné après avoir lancé quelques fois l'algorithme, ce qui a permis de faire un choix, principalement pour les valeurs maximums et minimums. L'algorithme a été entraîné, avec le "grid search", pour chaque feature selection ainsi qu'avec toutes les features.

Un algorithme non-supervisé de clustering a également été implémenté pour voir s'il était possible de regrouper les différentes activitées. L'algorithme utilisé pour réaliser cette tâche se trouve être KMeans offert par scikit-learn. Trois différents nombres de clusters ont été testés qui sont les suivants : 2, 3 et 6. Le but étant de voir pour quels nombres l'algorithme arrive à faire des clusters "propre". Une fois l'algorithme déroulé, nous avons dessiné les histogrammes de chaques clusters pour voir quels labels sont contenu dans ceux-ci. L'algorithme a aussi été entraîné avec tous les types de feature selection.

## 5. Experiments and results

Les résultats des différents algorithmes implémentés sont décrits dans les paragraphes suivants.
Pour **KNN**, peut importe le set de feature, le meilleur p est 1. Apparement, la distance Manhattan produit les meilleurs résultats. Avec toutes les features et n=7, on obtient une accuracy de 0.95. Pour feature set de la méthode 1 et n=5, on obtient une accuracy 0.90 et finalement pour le feature set de la méthode 2 et n=3 on obtient une accuracy de 0.93.
On peut voir que l'accuracy la plus élevée s'obtient avec toutes les features. Cependant, les méthodes de features selection sont assez efficace et l'on voit que l'accuracy ne descends pas en dessous de 0.9 malgré qu'il manque une majeure partie des features.

Pour **MLP**, il y a très peu de différence entre les features set. On obtient une accuracy de 0.93 avec toutes les features, de 0.93 avec la méthode 1 et 0.92 avec la méthode 2. Dans tous les cas, la meilleure valeur pour le paramètre hidden_layer_sizes est (7,14).

| Feature selection | Meilleur activation | Meilleur alpha | Meilleur hidden_layer_sizes | Meilleur max_iter | Accuracy |
|---|---|---|---|---|---|
| Toutes les features | tanh | 0.001 | (7, 14) | 300 | 0.93 |
| Feature selection 1 | relu | 0.0001 | (7, 14) | 300 | 0.93 |
| Feature selection 2 | tanh | 0.001 | (7, 14) | 400 | 0.92 |

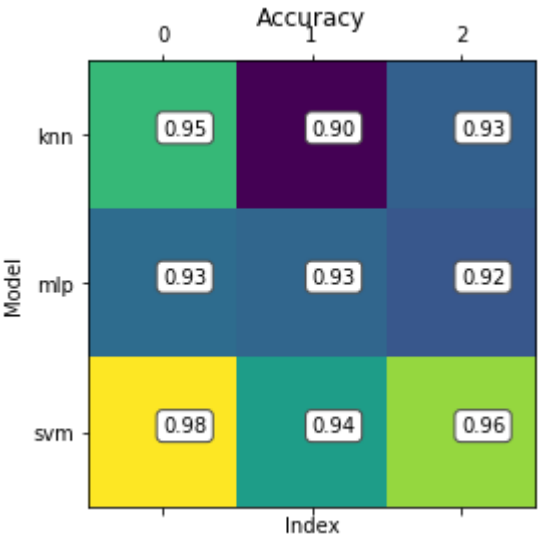
Avec **SVM**, pour les 3 features set, la meilleure fonction de décision est one-vs-one (ovo). Pour le reste, voici un tableau présentant les résultats :

| Feature selection | Meilleur C | Meilleur kernel | Accuracy |
|---|---|---|---|
| Toutes les features | 200 | rbf | 0.98 |
| Feature selection 1 | 200 | rbf | 0.94 |
| Feature selection 2 | 1000 | rbf | 0.96 |

Les performances obtenus sont très bonne d'une manière globale mais l'entraînement avec toutes les features est supérieur aux 2 autres.

Pour **KMeans**, nous avons calculé les histogrammes pour les trois valeurs de clusters testés ainsi que pour les 3 types de feature selection. Avec 2 clusters, l'algorithme arrive très bien à différencier les activités de marche, des activités statiques. Avec 3 clusters, il arrive à faire 1 cluster de plus contenant l'activité "LAYING". Pour 6 clusters, l'algorithme n'arrive pas à bien séparer les activités, on remarque tout de même que la séparation entre les activités de marche et statiques est toujours présente.

## 6. Analysis and conclusions




Dans le but de déterminer quel algorithme donne les meilleurs résultats, nous avons fait un récapitulatif de tous les algorithmes supervisés implémentés ainsi que pour les 3 types de feature selection. Pour ce faire voici un tableau contenant toutes les valeurs de performances obtenues, avec les lignes correspondants à l'algorithme et les colonnes correspondant a la feature selection, a noter que le 0 correspond à toutes les features.

On remarque tout de suite que le meilleur algorithme est SVM que ce soit pour les 3 types de feature selection. Le fait de prendre toutes les features permet de faire augmenter la performance, ce qui veut dire que la plupart des features sont utiles à la prédiction. A noter également une chute de performance avec MLP et la deuxième feature selection. Sinon tous les résultats sont entre 0.90 et 0.98, ce qui est de manière général très bon.

Bien que l'algorithme non-supervisé KMeans n'arrive pas a faire 6 clusters bien propre, les résultats restent tout de même intéressant. D'une part, les activités de mouvements sont très bien différenciés des activités statiques et d'autre part, la séparation de "LAYING" des 2 autres activités statiques ("SITTING" et "STANDING") se fait très bien. Mais l'algorithme n'arrive pas à différencier les 3 types de marche ainsi que "SITTING" et "STANDING". La différenciation entre "SITTING" et "STANDING" est compliqué car ce sont des activités très similaires. En effet, les 2 sont des activités statiques et le haut du corps reste vertical, ce qui est donc difficile pour un algorithme de clustering car il aura tendance à les mettre dans le même cluster ou de faire 2 clusters contenant ces 2 activités. La différenciation des 3 types de marches doit être de même nature car sûrement très similaire.

Pour conclure, le meilleur modèle se trouve être l'algorithme SVM avec l'utilisation de toutes les features. Ce qui nous donne une performance égale à 98% de réussite sur les données de test, ce qui est un très bon résultat. A noter également les résultats intéressants du test avec KMeans bien que inutilisable pour faire de la prédiction sur les 6 labels.