TX
Mobilité Dynamique:
Analyse de données, classification et prédiction

Sofiane Lamrous & Jules Vercoustre

6 janvier 2018

Contexte

Le dévelopement des smartphones ces dernières années a permis l'accès à une multitude de possibilités. Aujourd'hui il est simple de suivre les deplacements de chacun grace aux données GPS collectées par les smartphones. Mobilité Dynamique est une application developpée dans ce sens, elle permet à ses utilisateurs d'avoir un suivi de leurs deplacements mais surtout de contribuer à la recolte de données GPS dans un but de recherche et d'exploration pour ameliorer les reseaux de transport locaux. Les données GPS peuvent être exploitées de différentes manières et peuvent contribuer à la réalisation de nombreux projets. Dans le cadre de notre TX nous nous sommes penchés sur la problematique de reconnaissance du mode de transport puis de manière plus specifique à la reconnaissance des lignes de bus pour les déplacement en bus. Dans la dernière partie nous avons traité le sujet de pattern et de périodicité des voyages. En effet nous avons identifié pour chaque utilisateurs leur lieux d'intérêt c'est à dire les lieux les plus fréquentés pour mieux comprendre leurs habitudes de transport.

Table des matières

1 Reconnaissance des ligne des bus

1.1 Analyse statique du jeu de données "Mobilité Dynamique"

De prime abord, nous avons décidé d'analyser un jeu de données au format json issu de l'application. La taille du jeu de données correspond à des enregistrements sur quelques mois en 2017. Etant donné que nous utiliserons par la suite le logiciel R il s'agit d'abord de convertir les données en un data frame tel que ci-dessous :

TABLE 1 – Jeu de données

	id	date	lat	lng	mode	mode_str
1	59046be7c9e77c0001b582e4	2017-07-16 13 :44 :42	49.42239	2.820488	3	Still
2	59046be7c9e77c0001b582e4	2017-07-16 13 :48 :27	49.42220	2.819472	3	Still
3	59046be7c9e77c0001b582e4	2017-07-16 14 :32 :45	49.42217	2.820005	9	Entering Geofence

Sachant que :
- *id*, l'id unique des utilisateurs
- *date*, la date de l'enregistrement
- *lat lng*, les coordonnées géographiques, respectivement latitude et longitude.
- *mode*, le mode de transport utilisé :
  - 0, on a vehicle
  - 1, biking
  - 2, on foot
  - 3, still
  - 4, unclassified
  - 5, tilting
  - 7, fast walk
  - 8, run
  - 9, entering a geofence (that is a location where the user is remaining for at least 20 minutes)
  - 10, exiting from a geofence (that is transport begins)
  - 11, tracking service started
  - 12, tracking service stopped

Afin de visualiser l'exactitude des coordonnées géographiques enregistrées nous avons représenté dans l'espace les différents enregistrements :

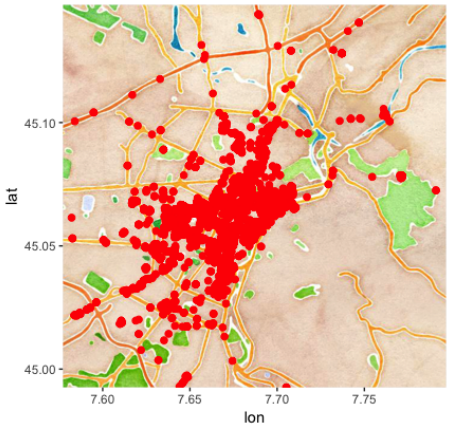

FIGURE 1 – Visualisation des données de Turin

Il vient donc que les coordonnées sont très précises (nombre de décimales très élevés). Cependant cette précision accrue pose un problème majeur. En effet, lorsque qu'à lieu un enregistrement celui-ci n'est pas forcèment la position "réelle" de l'utilisateur. Ainsi, la marge d'erreur entre le point réel et le point enregistré peut varier de 0 à plusieurs dizaines de mètres. C'est une problématique à prendre en compte lors de la classification d'un point sur une ligne de bus ou non.

1.2 Récupération des données des lignes de bus

Afin de déterminer un classifieur permettant de déterminer si un point enregistré est ou non sur une ligne de bus il est nécessaire d'avoir les données GPS des lignes de bus (en loccurence celles de Compiègne). Cette partie a été délicate et fastidieuse. En effet, il nous fallait un nombre d'enregistrements conséquent et très précis pour pouvoir ensuite utiliser cette base de points pour un classifieur. Nous avons réussi à obtenir ces points en nous basant sur le site web Compibus qui scrape les données via une API dédiée. Obtenir les données nous a pris du temps mais nous sommes finalement parvenus à obtenir le tracé exact de toutes les lignes de bus de Compiègne (format json).

FIGURE 2 – Ligne 5

1.2.1 Classifieur euclidien de la ligne de bus

Un pan de notre projet consistait à déterminer si un enregistrement se situe ou non sur l'une des lignes de bus de Compiègne. Nous avons donc choisi de constuire un classifieur euclidien qui réalise cela. Le principe du classifieur est le suivant :

- Une droite est tout d'abord tracée entre deux points de la ligne de bus : (x_i, y_i) *et* (x_{i+1}, y_{i+1})
- Ensuite un intervalle de confiance est construit (ou marge de confiance). Celui-ci est composé de deux droites parallèles à la droite précédement tracée et distantes d'une valeur delta $\delta = d$ de cette même droite. Ces droites ont pour équation $d_1 = ax + b + d$ et $d_2 = ax + b - d$. On notera que d est un paramètre réglable.
- Enfin deux dernières droites viennent entrecouper les droites d_1 et d_2 pour former au final un parallélogramme qui constitue la zone au sein de laquelle on considère qu'un point se situe sur la ligne.

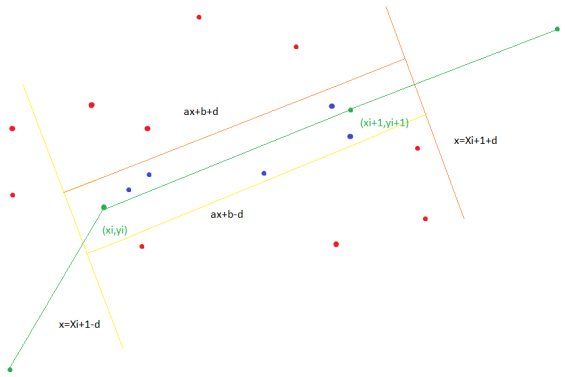

FIGURE 3 – Principe classifieur euclidien

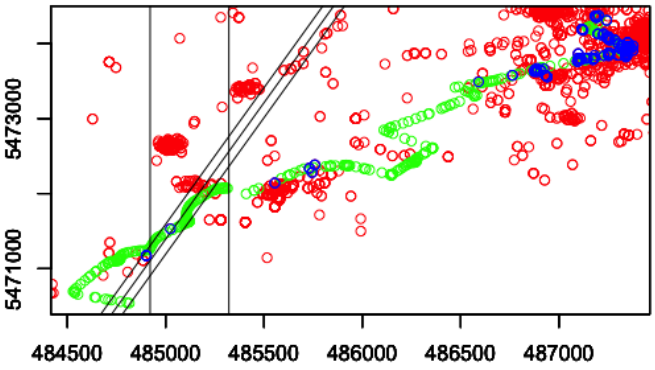

FIGURE 4 – Exemple classifieur euclidien sur la ligne 5

1.3 Conlusion sur la reconnaissance des lignes de bus

Nous avons réussi à déterminer un classifieur euclidien permettant de déterminer les points appartenant ou non à une ligne de bus. Ce classifieur pourra être par le futur être amélioré en retournant les utilisateurs auxquels appartiennent les points. Cela permettra par la suite d'informer les usagers (de l'application) sur leur trajets quotidiens ou récurrents et sur la possibilité d'emprunter une ligne de bus adaptée à leur besoins. Tout ceci dans une démarche éco-responsable qui s'intégre dans une problématique de mobilité intelligente.

2 Reconnaissance des modes de transport

Dans ce rapport, nous proposons une méthode pour identifier au mieux 4 modes de transports incluant la marche, le vélo, la voiture et le bus, le tout en utilisant les données GPS.

Les principales étapes de la méthode pour l'identification des modes de transport sont :
— Nettoyage des données
— Extraction de nouvelles variables explicatives
— Modèle de classification
— Evaluation du modèle

Les données GPS brutes sont tout d'abord regroupées par utilisateurs puis mises en relation pour ensuite être découpées en trajets ("trips"). Ensuite des variables explicatives (prédicteurs) sont extraites des données GPS avant d'utiliser la méthode qui classifiera nos trajets en fonction des modes transport.

De la méthode on obtient ensuite des variables plus importantes et consistantes que certaines. On procède donc à l'élimination des variables non pertinentes.

Chacunes des étapes est décrite dans cette section.

2.1 Nettoyage des données

De prime abord, pour de meilleur performances, 2 techniques de nettoyage des données ont été utilisé dans ce projet. D'abord nous avons enlevé les duplicats dans le jeu de données. En effet, des points GPS sont parfois enregistrés plus qu'une fois du fait d'erreurs d'enregistrement sur l'appareil GPS. Aussi, nous avons supprimé les trajectoires de type "outlier" qui dévient anormalement. Par exemple si la vitesse moyenne d'une trajectoire marqué "Marche" excède 10 m/s alors elle est marqué comme aberrante et retiré du jeu de données.

Ensuite, nous avons décidé que les prédictions se ferait sur des "voyages" (aussi appelé "trips" dans ce rapport). Ainsi nous avons splité le jeu de données initial en n voyages. Chaque utilisateur x_i a un total de m voyages. La somme des x_i^m (i variant de 1 à U, U étant le nombre d'utilisateurs) nous donne donc n.

Chaque voyage/trip est composé de e enregistrements de la façon suivante :
— Un enregistrement de type "entering a geofence" (mode = 9) qui initie le voyage (x_i^1)
— Un nombre variable *dots* ($dots > 2$ en raison des calculs de vitesse, d'accélération, etc) d'enregistrements avec des modes de transports différents ou non (mais différents de 9 et de 10).
— Un enregistrement type "exiting a geofence" (mode = 10) qui indique la fin du voyage (x_i^e).

Avec cette nouvelle structure de données nous nous sommes retrouvés avec des voyages où le mode utilisé est "Still" (mode=3). C'est à dire que l'usager n'est pas en mouvement. Or cette classe ne sera pas utilisée par notre futur classifieur. En effet on cherche à prédire des modes de transport et non l'absence de mouvement. On supprime donc tous les voyages de ce type.
Enfin par la suite nous avons décidé de limiter chaque voyage à un seul type de mode de transport. Ainsi nous avons déterminé des "sous-voyages" parmis les "voyages".

Exemple de voyage :

Table 2 – Jeu de données

	id	date	lat	lng	mode	mode_str
0	59046be7c9e77c0001b582e4	2017-07-16 13 :44 :42	49.42239	2.820488	9	Entering Geofence
1	59046be7c9e77c0001b582e4	2017-07-16 13 :44 :42	49.42239	2.820488	2	On foot
2	59046be7c9e77c0001b582e4	2017-07-16 13 :48 :27	49.42220	2.819472	2	On foot
3	59046be7c9e77c0001b582e4	2017-07-16 14 :32 :45	49.42217	2.820005	2	On foot
4	59046be7c9e77c0001b582e4	2017-07-16 13 :48 :27	49.98620	2.564743	0	Vehicule
5	59046be7c9e77c0001b582e4	2017-07-16 14 :32 :45	49.97867	2.874832	0	Vehicule
6	59046be7c9e77c0001b582e4	2017-07-16 14 :32 :45	49.97233	2.876232	10	Exiting geofence

Exemple de sous-voyage :

Table 3 – Jeu de données

	id	date	lat	lng	mode	mode_str
0	59046be7c9e77c0001b582e4	2017-07-16 13 :44 :42	49.42239	2.820488	9	Entering Geofence
1	59046be7c9e77c0001b582e4	2017-07-16 13 :44 :42	49.42239	2.820488	2	On foot
2	59046be7c9e77c0001b582e4	2017-07-16 13 :48 :27	49.42220	2.819472	2	On foot
3	59046be7c9e77c0001b582e4	2017-07-16 14 :32 :45	49.42217	2.820005	2	On foot
4	59046be7c9e77c0001b582e4	2017-07-16 14 :32 :45	49.97233	2.876232	10	Exiting geofence

2.2 Extraction de nouvelles variables explicatives

L'extraction de variables explicatives joue un role important dans la reconnaissance des modes de transports. Nous utilisons donc nos connaissance dans le domaine des transports pour créer des variables consistantes dans un contexte d'apprentissage automatique. L'apprentissage automatique doit fonctionner le mieux possible et ainsi présenter des résultats robustes avec un faible taux d'erreur pour la reconnaissances des différents modes de transport. On décide donc de créer le plus de variables possible pour un choix final de variables plus travaillé.

Les variables explicatives font référence aux statistiques descriptives pour la trajectoire complète. Ce qui rend chaque trajectoire davantage comparables avec les autres. Aussi parmi les variables explicatives, certaines sont obtenues par la décomposition du profil de chaque trajectoire. On a ainsi davantage de détail en ce qui concerne le mouvement et le comportement de la trajectoire.

Tout d'abord nous avons calculé 4 variables de "mouvements" à partir des trajectoires : la vitesse, l'accélération, l'angle entre deux points (direction de deux points consécutifs) et la sinuosité (le chemin effectif divisé par la distance la plus courte). Nous avons ensuite utilisé les méthodes statistiques pour extraire d'autres variables (basées sur les variables de mouvements définis précèdement) :

- Moyenne
- Deviation standard : mesure la dispersion.
- Mode : represente la valeur qui apparait le plus fréquement dans le jeu de données.
- top 3 min
- top 3 max
- Etendue des valeurs
- 1er et 3eme quartiles
- Skewness (coefficient d'asymétrie) : mesure de l'asymétrie de la distribution d'une variable aléatoire réelle
- Kurtosis (coefficient d'applatissement) : mesure de l'aplatissement, ou a contrario de la pointicité, de la distribution d'une variable aléatoire réelle
- ...

2.3 Classification et évaluation du modèle

Pour cette partie nous avons besoin de données d'apprentissages robustes. Or nous n'en avons pas à l'instant présent. Nous avons donc décider de modéliser un classifieur qui pourra s'appliquer à de futures données robustes.

Le modèle de classifieur, afin de prédire de future données, a besoin d'être "entrainé" sur un jeu de données dit d'apprentissage. Nous avons donc constitué un dataframe qui contient n "voyages". A chacun des n voyages est associé p variables explicatives ainsi qu'une réponse y (mode) qui correspond au mode de transport utilisé. Le mode de transport, la réponse donc, est en effet la variable que l'on va chercher à prédire. Le but *in fine* est, lorsque que l'on dispose des données GPS d'un voyage d'un utilisateur, de prédire de manière précise et certaine son mode de transport.

Analysons maintenant en détail le jeu de données dont nous disposons.

2.3.1 Description

Jeu de données composé d'exemples issus de 4 classes correspondants à 4 modes de transport 0,1,2 et 4 (respectivement vehicule, biking, on foot et unclassified).

2.3.2 Dimensions

190 observations avec 49 variables prédictives et une étiquette (réponse) dont les labels sont 0,1,2 et 4.

- Ici $n > p$ cependant n reste très petit par rapport à p. En effet on a seulement 3 fois plus d'observations que de prédicteurs. On aura donc un fléau de dimension aussi appelé problématique de type "high-dimensional" dans la littérature anglaise.
- On fera donc attention au sur-apprentissage, qui peut être plus important sur un jeu de données possédant une dimension légérement plus petite que le nombre d'observation.

2.3.3 Méthodologie

Analyse numérique des données

- Analyse des distributions des données afin de savoir si elle sont déjà centrée et réduites ;
- Analyse du taux de "représentation" des différentes classes au sein du jeu de données.
  - Dans le cas où une classe est sur-représentée, par exemples à 90%, la mesure de la performance d'un classifieur devrait alors être plus fine que simplement mesurer le pourcentage de données correctement classifiées ;
  - En effet, un classifieur qui classe simplement toutes les données comme appartenant à la classe sur-représentée aurait, dans l'exemple précédent, un pourcentage d'erreur de 10% seulement, alors qu'il est fondamentalement très mauvais.

Familles ou types de modèles testés

- Random Forest
- Support vector machine avec Kernel linéaire
- Support vector machine avec Kernel polynomial
- Support vector machine avec Kernel gaussien

Validation croisée K-fold répétée

Pour chaque type de modèle dont on veut tester les performances, on répètera 3 fois une validation croisée de type K-Fold avec $K = 5$. Nous nous référerons à ce type de validation croisée en l'appelant *validation croisée 5x3*.

Cela veut dire qu'on entrainera $5 * 3 = 15$ classifieurs pour chaque modèle testé (plusieurs par famille de modèles afin de sélectionner les hyper-paramètres).

Recherche et sélection des hyper-paramètres par validation croisée

Les modèles testés possèdent tous un ou plusieurs hyper-paramètres pouvant grandement influencer leurs performances. La sélection de ces hyper-paramètres se fait à l'aide de *grilles* ou *grid* contenant des combinaisons générales et larges d'hyper-paramètres. Une fois la meilleure combinaison choisie (plus faible pourcentage d'erreur par validation croisée), on affine la grille de recherche autour de ces hyper-paramètres.

Dans le cas de modèles à 3 (ou plus) hyper-paramètres, les combinaisons sont trop nombreuses pour nos capacités de calculs. On a donc opté pour une méthode de séparation : on sélectionne deux hyper-paramètres par validation croisée en gardant le troisième constant à sa valeur théorique par défaut, puis ce sont les deux premiers hyper-paramètres qui deviennent constant (avec les valeurs trouvées précédemment) et le troisième qui est sélectionné par validation croisée.

C'est par exemple ce que nous avons fait pour le modèle de SVM à kernel polynomial, dans lequel il y a trois hyper-paramètres : le coût (C), le degré du polynôme (d) et le "scale" (mesure de la dispersion des observations). Nous avons d'abord effectué le tuning du coût et du degré en gardant le scale à sa valeur par défaut (1 / nombre de prédicteurs). Puis une fois les meilleures valeurs du coût et du degré déterminées par validation croisée, nous relançons une dernière passe de tuning pour "scale" autour de sa valeur par défaut.

Construction de statistiques mesurant les performances des modèles

Dans chacun des jeux de données, les classes sont représentées de manière équitable. On utilisera donc la moyenne des pourcentages d'erreur des validations croisées répétées comme statistique de mesure de performance des modèles, ainsi que son intervalle de confiance.

Pour le modèle sélectionné, nous détaillerons les étapes de tuning ayant conduit au choix des valeurs des hyper-paramètres.

2.3.4 Analyse statistique des données

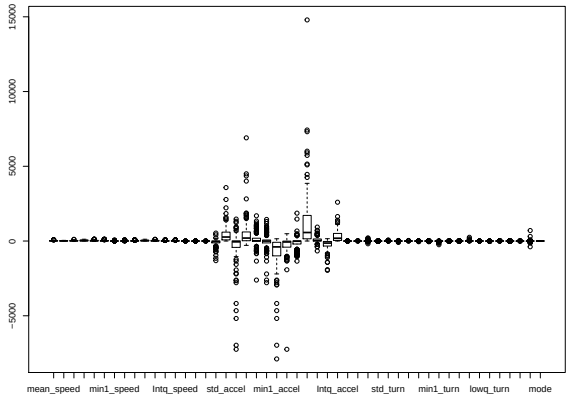

FIGURE 5 – Distribution des variables

On remarque bien ici que les variables prédictives ne sont pas centrées-réduites. Ce que l'on confirme à l'aide d'une analyse numérique, leur moyenne ne sont pas proche de zéro et leur écarts-type pas proche de 1). On procède donc à un centrage-réduction. Alors on obtient le graphique suivant :

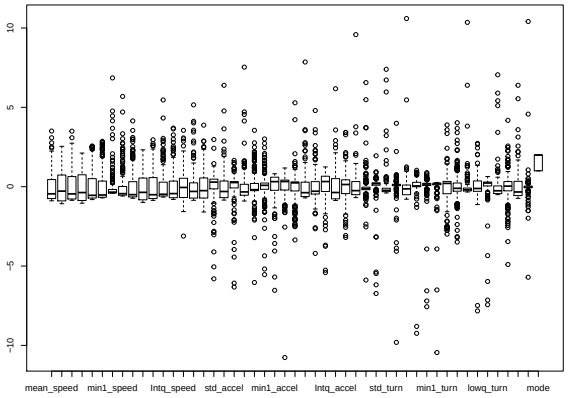

FIGURE 6 – Distribution des variables après centrage-réduction

Ensuite, on se penche sur la corrélation des variables. Comme on peut s'y attendre, certaines variables sont très corrélées entre elles. En effet, certains prédicteurs sont calculés en fonction d'autres prédicteurs.

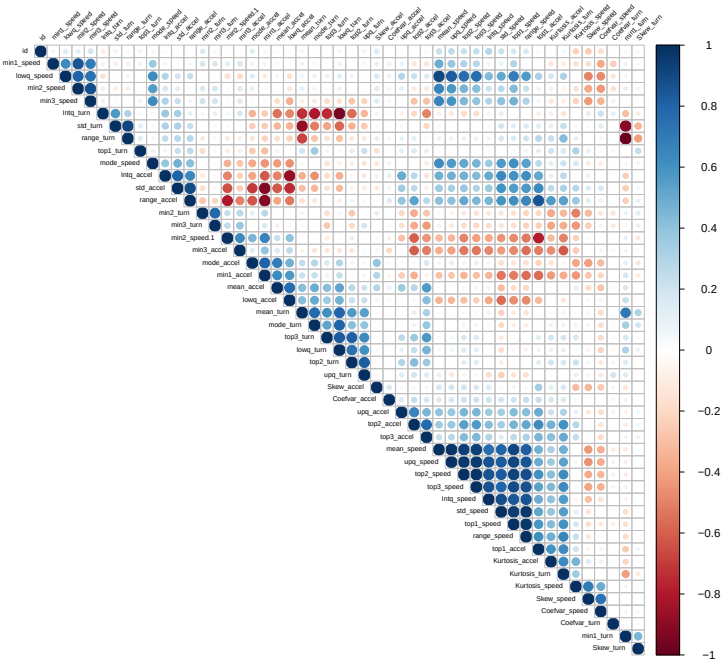

FIGURE 7 – Corrélation des variables entre elles

On peut penser à décorréler les variables à l'aide d'une ACP. Cependant, après calculs, une ACP n'améliore pas le modèle de classifieur final. On choisi donc de poursuivre avec des variables corrélées.

Enfin, les classes, au nombres de 4, sont distribuées de manière très disproportionnée.

On sait donc par avance que notre modèle tendra a être efficace pour prédire les classes 0 et 2 (similairement réparties). Mais totalement inefficace pour les classes 1 et 4 qui sont extrêmement

Table 4 – Proportion des classes

0	1	2	4
46.315789	1.578947	49.473684	1.052632

peu présentes dans notre jeu de données.

Etant donné les répartitions des classes 1 et 4 on choisi de travailler sans les observations correspondantes. En effet il est inutile d'essayer de prédire des classes représentant chacune seulement 1% des classes du data set.

On a donc les nouvelles proportions suivantes :

Table 5 – Nouvelles proportion des classes

0	2
47.56757	50.81081

Le paramètre n vaut maintenant 182. La taille du jeu de donnée a peu changé.

2.3.5 Performance des modèles

Table 6 – Performance des modèles

id	Modèle	Erreur de Validation Croisée (en %)	Intervalle de confiance (en %)
C_1	Random forest	9.32	[4.66 , 13.99]
C_2	SVM Kernel linear	10.59	[5.75 , 15.44]
C_3	SVM Kernel polynomial	8.95	[2.72 , 15.18]
C_4	SVM Kernel Gaussian	29.86	[23.20 , 36.53]

2.3.6 Choix du modèle

C'est le classifieur C_3 (SVM Kernel polynomial) qui minimise l'erreur de validation croisée. Nous choisirons ce classifieur pour de futurs prédictions et allons en expliciter la méthode de construction.

2.3.7 Processus de tuning du modèle choisi

Afin de sélectionner les meilleurs hyper-paramètres C (coût) et σ (l'écart type de la distribution gaussienne), nous avons sélectionné une grille de combinaisons de C et σ avec $C = \{0.1, 1, 10, 100\}$ et $\sigma = \{1, 2, 3, 4\}$.

La figure ci-dessous représente les taux de classification correcte (avec validation croisée 5x3) pour la grille de tuning.

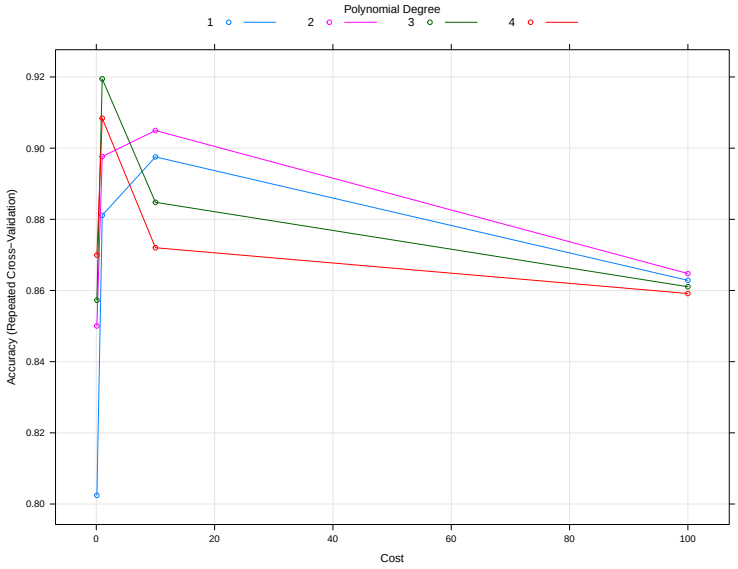

FIGURE 8 – Tuning du SVM Kernel polynomial

Conclusions de la première grille de tuning : Les meilleures performances sont obtenues pour un coût $C = 1$ et pour un degré de $d = 3$. Cependant, pour coût proche de 0 ($C>0$) on voit que la courbe est vite décroissante. On va donc affiner la grille de tuning pour C entre 0 et 1.

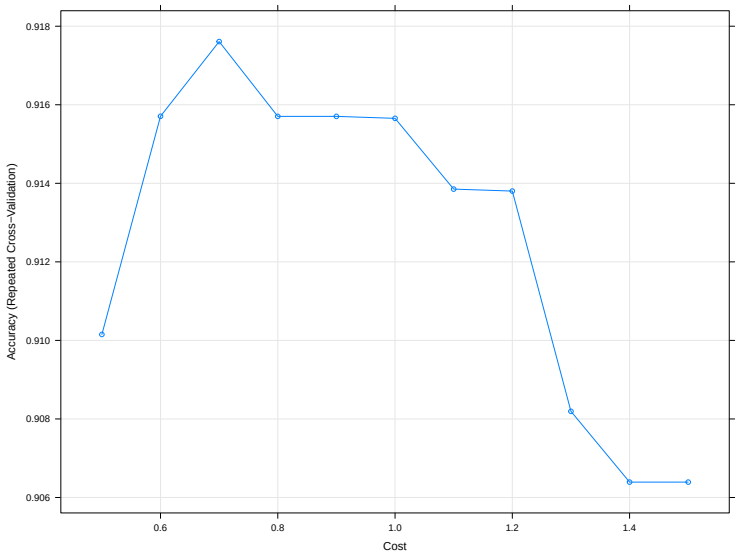

FIGURE 9 – Tuning du SVM Kernel polynomial autour de Cost (C)

On trouve alors finalement un modèle optimal avec $C = 0.7$.

2.3.8 Conclusion du classifieur

Le problème posé par le jeu de données est, comme nous l'avons vu précédement, "high-dimensional". En effet nous avons beaucoup de prédicteurs pour peu d'observations. Cependant, avec un taux de bonne classification de 91.05% notre classifieur est performant. Avec encore davantage de données il serait encore plus précis.

Aussi nous avons pu entrainer notre classifieur sur des observations appartenant à 2 classes. Si on ajoutait des observations correspondant à d'autre classes alors notre classifieur serait plus complet car capable de prédire davanatge de classes. Mais alors peut être qu'un autre modèle SVM Gaussien, Random forest ...) serait plus précis. Si un tel jeu de données se présente (actuellement nous n'en avons pas) alors on pourra utiliser les classifieurs tout prêts fournis en annexe.

2.3.9 Pistes d'améliorations

Idéalement il nous faudrait un jeu de données robuste afin de prédire efficacement toutes les classes de transport possible. Mais, on le sait, concevoir un tel jeu de données est très compliqué et très chronophage (activer l'application à chaque début/fin de voyage pour des dizaines d'utilisateurs). Cependant ce n'est pas impossible et si on pouvait avoir environs 100 observations pour tous les modes de transports alors on pourrait construire un classifieur davantage fiable et exhaustif (tous les modes de transport : biking, foot, car, bus ...).

3 Pattern

3.1 Problématique

Il nous a semblé intéressant d'analyser les habitudes des utilisateurs de l'application pour comprendre quel moyen de transport est utilisé pour tel ou tel deplacement. Le but est de pouvoir identifier le moyen de transport utilisé pour effectuer des déplacement "quotidient" comme par exemple le trajet "maison->bureau" ou bien un trajet inhabituel. Pour ce faire il faut identifier les lieux habituels des utilisateurs c'est à dire les lieux qu'il fréquente de maniere régulière dans le temps. L'idée globale est de pouvoir par la suite, grâce au profil de chaque utilisateur, comprendre la corrélation entre les moyens de transports les plus utilisés et par exemple la distance entre le lieu d'habitation et le lieu de travail. Voire même comprendre une corrélation entre la tranche d'âge de la personne et son moyen de locomotion favori.

Dans cette partie nous avons reussi à extraire les lieux d'intérêt des utilisateurs, malheureusement nous n'avons pas su identifier les types de lieu : "maison" "bureau" "courses" etc. En effet c'est une étape plus délicate.

Pour la réalisation nous nous sommes inspiré de l'article *Periodic Pattern Mining Based on GPS Trajectories* (cf références) qui donne une méthode pour trouver des pattern à partir de données GPS. Nous allons donc utiliser le *Periode Pattern Mining* afin de pouvoir obtenir des activités périodiques. Voici un schèma pris de l'article qui explique l'idée generale.

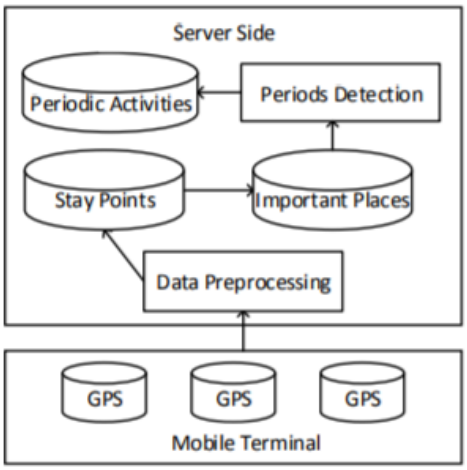

Figure 10 – Periode Pattern Mining

3.2 Filtrage des données

Avant de commencer l'analyse il a fallu procéder à l'étape de filtrage des données et de calcul des vitesses pour tous les points. Le filtrage consiste à retirer tous les utilisateurs ayant uniquement un seul enregistrement dans notre jeu de données sur 144 users. Il y a 11 users avec un enregistrement unique, nous les avons donc retirés. Le calcul des vitesses est le même que dans la partie precedente. Nous avons par la suite fixé deux limites :

— Une limite de temps
— Une limite de vitesse

Dans le document que nous avons pris comme support, la méthode utilise un temps minimal d'arrêt et une distance maximale. Dans notre code nous avons choisi d'utiliser les limites précédement citées.

3.3 Les points d'arrêts

Un point d'arrêt est une zone géographique où l'utilisateur s'arrête pendant une période donnée. Pour trouver ces zones il faut donc définir une limite de temps d'arrêt et une limite de vitesse car les points GPS ne sont pas d'une très grande précision. Dans notre cas nous avons choisi ces paramètres de manière empirique. Nous avons fixé le temps à au moins 1/4 d'heure et la vitesse max à 0.8 km/h.

3.4 Les points d'intérêts

Après avoir trouvé les points d'arrêts il nous a fallu utiliser un algorithme de clustering pour regrouper les points d'arrêts en clusters. Nous avons utilisé le `dbscan clustering` c'est à dire un regroupement par densité. Les centroïdes de chacun des clusters correspondent aux points d'interêt.

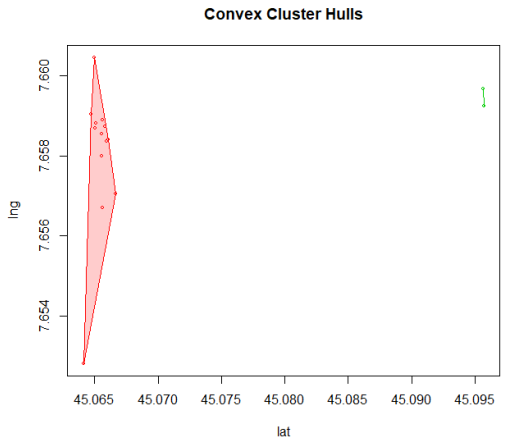

FIGURE 11 – Résultat obtenu pour un utilisateur

3.5 Fonction

La fonction pattern(user,Tlim,Vlim) présente à la fin du fichier `Step_4_Pattern.R` prend en paramétre le numero de l'utilisateur dans le dataframe userpattern. Le temps limite et la vitesse limite sont deux paramètres importants pour définir les conditions d'immobilité. La fonction retourne en sortie les differents points d'arrêts et les clusters.

3.6 Piste d'amelioration Pattern

— Calculer la periodicité des points d'intérêt. Cela permet de voir si un lieu est visité d'une maniere frequente ou de maniere occasionel.
— Interoger une API pour pouvoir connaitre la nature des lieux et donc nommer les points d'intérêts que nous avons extrait.
— A l'aide à un ensemble d'apprentissage choisir les meilleurs parametre de seuil de temps et de vitesse.

4 Conclusion générale

Ces travaux que nous avons réalisé nous ont permis de nous plonger dans la problématique très actuelle des transports intelligents. Tout d'abord nous avons pu étudier et construire un classifieur euclidien permettant de prédire si un utilisateur est ou non sur une ligne de bus. Aussi nous avons développé un classifieur qui détermine avec précision le mode de transport utilisé par tout utilisateur. Enfin nous nous sommes intéressés à la découverte de patterns sous-jacents liés aux trajets récurrents des utilisateurs. Pour finir tout ces travaux ont pour but final d'être améliorés afin d'être finalement intégrés à l'application *Mobilité Dynamique*.

5 Références

— Website : *http* : *//www.ign.fr/sites/all/files/geodesie*$_p$*rojections.pdf* - Autors : IGN
— Website : *https* : *//www.maptools.com/tutorials/utm/why*$_u$*se*$_u$*tm* - Autors : Maptools
— Article : Identifying Different Transportation Modes from Trajectory Data Using Tree-Based Ensemble Classifiers - Autors : Zhibin Xiao, Yang Wang , Kun Fu , and Fan Wu
— Article : glmulti : An R Package for Easy Automated Model Selection with (Generalized) Linear Models - Autors : Vincent Calcagno, Claire de Mazancourt
— Book : An Introduction to Statistical Learning - Autors : Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani
— Book : The Elements of Statistical Learning Data Mining, Inference, and Prediction - Autors : Trevor Hastie Robert Tibshirani Jerome Friedman
— Website : *http* : *//topepo.github.io/caret/index.html* - Autor : Max Kuhn
— Article : Periodic Pattern Mining Based on GPS Trajectories - Autors : Xiaopeng Chen, Dianxi Shi, Banghui Zhao and Fan Liu
— Classes : Machine Learning Strategies for Time Series Prediction - Autors : Gianluca Bontempi
— Article : The Euclidean distance classifier : an alternative to the linear discriminant function : Communications in Statistics - Simulation and Computation : Vol 16, No 2 - Autors : Virgil R. Marco, Dean M. Young, Danny W. Turner
— Article : Methods for Real-Time Prediction of the Mode of Travel Using Smartphone-Based GPS and Accelerometer Data - Autors : Bryan D. Martin, Vittorio Addona, Julian Wolfson, Gediminas Adomavicius and Yingling Fan
— Article : Predicting Travel Mode of Individuals by Machine Learning - Autors : Hichem Omrani
— Thesis : Inferring the Transportation Mode from Sparse GPS Data - Autors : Adel Bolbol