Projet IF23 : Localisation Indoor

Pauline Bergerot

I. Description du projet

L'objectif de ce projet était de pouvoir se localiser dans un bâtiment de l'Université de Technologie de Troyes en se servant de la force de signal reçue des différentes sources de wifi. Afin d'identifier les différentes sources j'ai enregistré les adresses SSID avec la puissance de leur signal.

Le projet s'est découpé en plusieurs parties. Il fallait d'abord récupérer les données dans les différentes salles du dernier étage du bâtiment C. Puis on a trié les données reçues pour ensuite entraîner un modèle d'intelligence artificielle afin de retrouver à l'aide des puissances des signaux wifi, la localisation d'une d'un ordinateur.

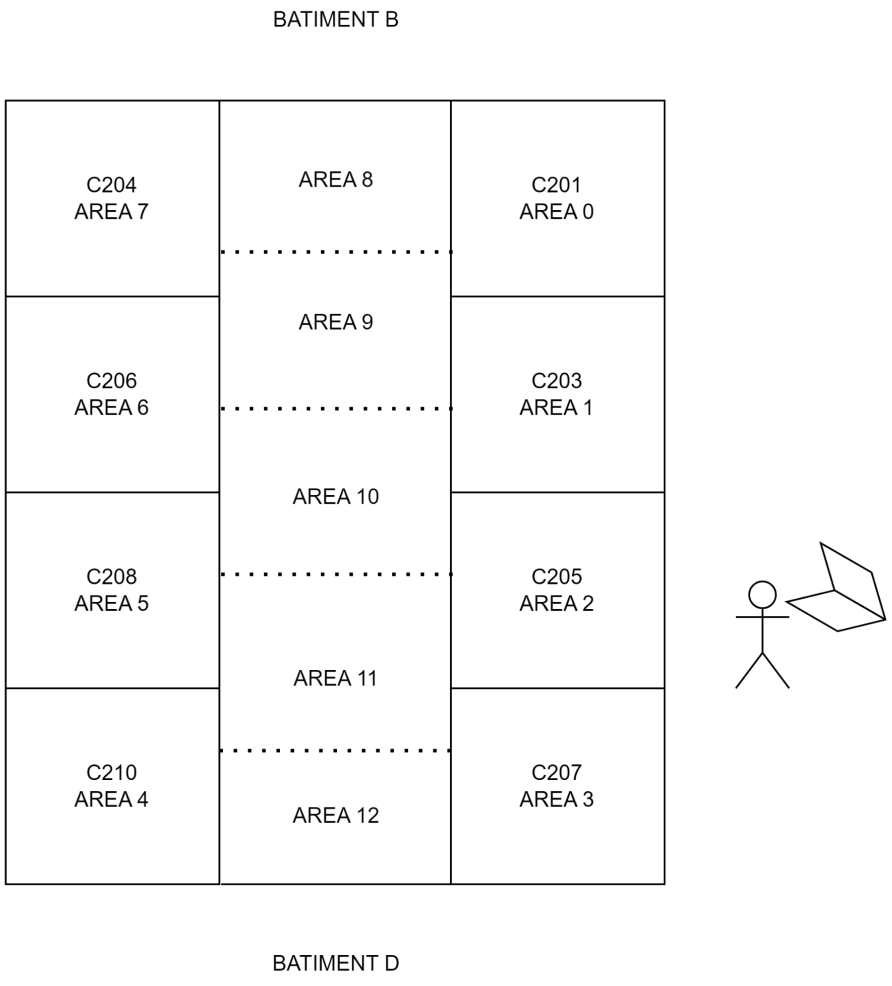

Figure 1 : Répartition des zones par salle / partie de couloir

Afin d'enregistrer plus facilement les données récupérées sur le chemin, j'ai défini chaque salle / partie du couloir par un numéro de zone. On peut donc grâce à cela traiter les données récupérer sans faire de différence entre couloir et salles.

II. Récupération des données

Afin de récupérer les données, j'ai ordonné le code de la manière décrite dans la figure 2.

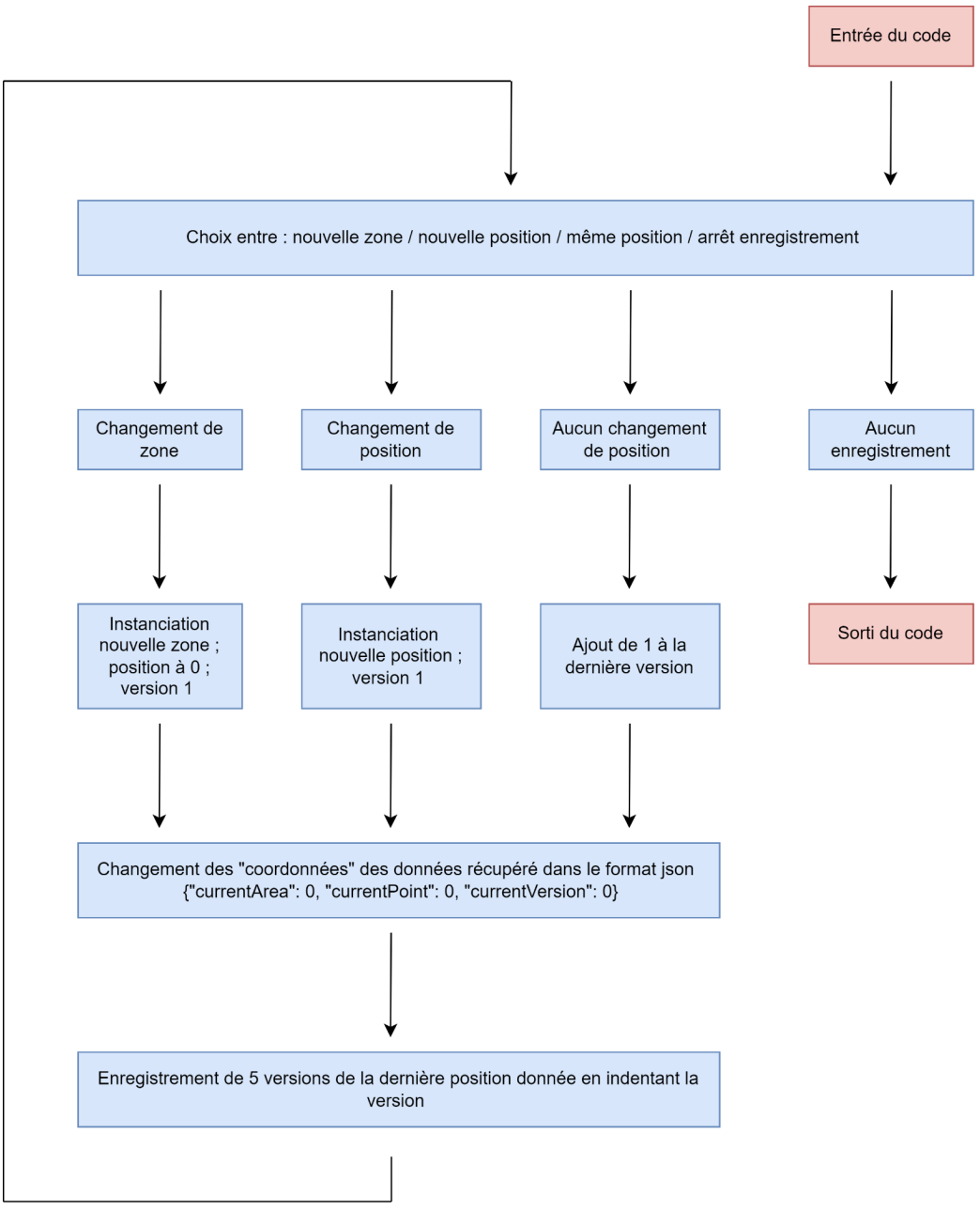

Figure 2 : Logique du code de récupération des données par position.

L'objectif était de récupérer les données de chaque position par tranche de 5 versions afin de pouvoir rapidement prendre le nombre souhaité de mesures mais en permettant de choisir à quel moment on voulait récupérer plus de données. Pour se repérer et reprendre les acquisitions à tout moment, les "coordonnées" de la position sont stockées dans un fichier à part (on y retrouve la zone, la position et la version).

Pour stocker les données, j'ai enregistré les données en créant un dossier pour chaque zone, un dossier pour chaque position de la zone et un fichier json de chaque version de la position actuelle. Les dossiers et fichiers se sont donc créés au fur et à mesure de l'acquisition en ajoutant un nouveau dossier pour toute nouvelle instanciation de zone, un nouveau dossier pour toute nouvelle instanciation de position dans une zone et un nouveau fichier pour chaque nouvelle version de position.

III. Gestion des données

Pour traiter les données j'ai d'abord enregistrer les fichiers sous forme de dossier et de fichier, puis sous forme JSON et enfin en tant que data frame pour visualiser et traiter les données sur python et en csv pour stocker de manière permanente les données.

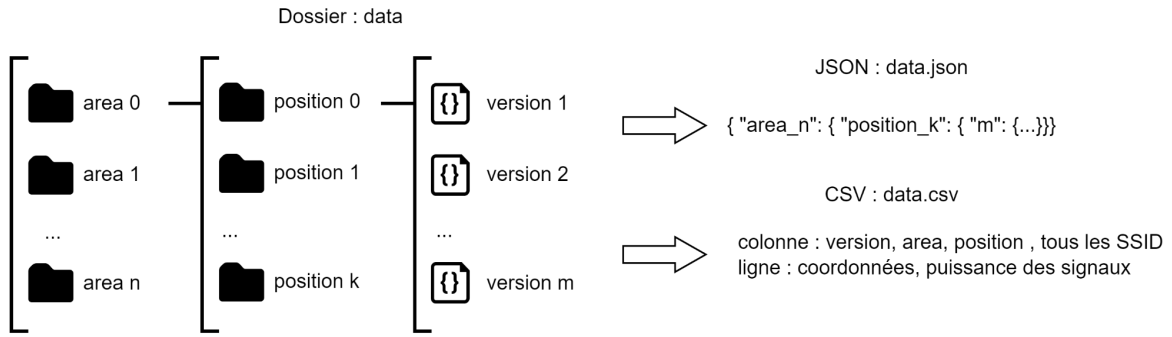

Figure 3 : Dossier / Fichier contenant les données enregistrée

La prochaine étape est de gérer les données manquantes. En effet, en fonction de la salle on peut capter différentes sources de wifi. On doit donc toutes les transformer en NaN ce qui nous permettra ensuite de les retirer ou de les utiliser différemment.

Pour tester d'autres analyses, j'ai remplacé toutes les valeurs de NaN par d'autres valeurs. Dans un premier cas j'ai tout remplacé par la valeur - 95. Dans un second j'ai remplacé pour les mêmes positions la moyenne des autres valeurs à cette position. Ce deuxième cas va me permettre de pousser les analyses mais elles ne pourra pas être utilisées pour analyser en temps réel la position de l'ordinateur.

IV. Traitement des données

On va d'abord chercher la meilleure manière d'exploiter les données récupérées. On peut faire une étude globale afin de définir au mieux les données. Les jeux de valeurs que nous possédons sont :

- avec NaN : les données en gardant les NaN
- avec -XX : les données où on remplace les NaN par -XX
- avec moyenne et -XX : les données où on remplace les NaN par la moyenne pour une même position ou par -XX

XX représente une valeur entre 0 et -130 tel qu'expliqué dans la partie qui suit.

a. Etude par classe ascendante hiérarchique

Une première façon d'analyser les données est de passer par une classification ascendante hiérarchique. Ce type d'étude permet de visualiser assez simplement les classes qui ressortent.

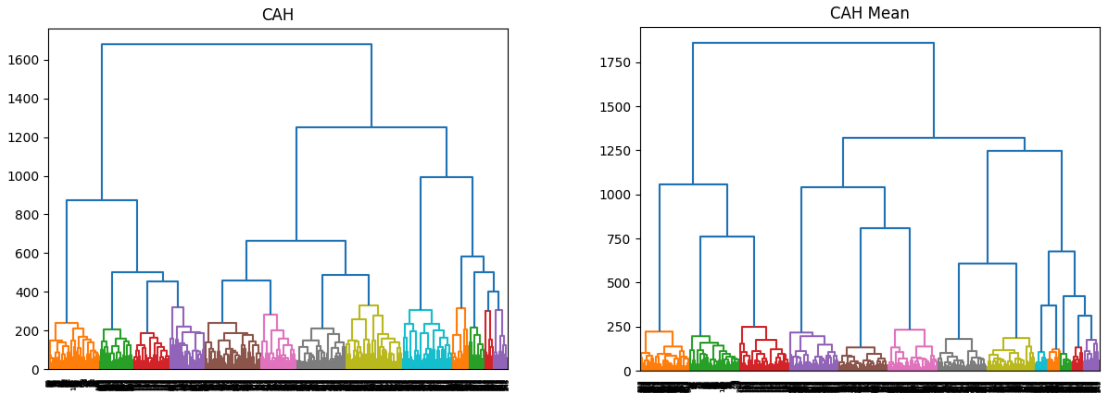

Figure 4 : Comparaison des dendrogrammes des données utilisant -95 et la moyenne à la place des NaN

En comparant ces deux schémas on note certaine parité mais on voit que ce ne sont pas exactement les mêmes zones dans les deux cas.

On va donc se servir des matrices de confusion afin de voir comment sont représentées chacune des zones à l'aide du traitement par CAH.

Area	1	2	3	4	5	6	7	8	9	10	11	12	13
area_0	100	0	0	0	0	0	0	0	0	0	0	0	0
area_1	6	71	23	0	0	0	0	0	0	0	0	0	0
area_10	0	0	2	0	0	0	0	0	0	0	23	0	0
area_11	0	0	0	0	0	0	0	0	0	0	9	6	15
area_12	0	0	0	0	0	0	0	0	0	0	0	12	13
area_2	0	0	2	0	10	0	12	76	0	0	0	0	0
area_3	0	0	0	0	0	12	87	1	0	0	0	0	0
area_4	0	0	0	0	0	4	0	0	96	0	0	0	0
area_5	0	0	0	0	0	96	0	0	4	0	0	0	0
area_6	0	0	0	0	97	2	0	1	0	0	0	0	0
area_7	0	0	33	65	2	0	0	0	0	0	0	0	0
area_8	0	0	4	0	0	0	0	0	0	21	0	0	0
area_9	0	0	11	0	0	0	0	0	0	14	0	0	0

Figure 5 : Matrice de confusion du traitement par CAH en remplaçant les NaN par -95

Area	1	2	3	4	5	6	7	8	9	10	11	12	13
area_0	0	100	0	0	0	0	0	0	0	0	0	0	0
area_1	0	2	98	0	0	0	0	0	0	0	0	0	0
area_10	0	0	0	0	0	0	0	0	0	0	0	22	3
area_11	0	0	0	0	0	0	0	0	0	0	0	0	30
area_12	0	0	0	0	0	0	0	0	0	0	25	0	0
area_2	0	0	2	0	0	0	0	98	0	0	0	0	0
area_3	0	0	0	0	0	0	100	0	0	0	0	0	0
area_4	0	0	0	100	0	0	0	0	0	0	0	0	0
area_5	0	0	0	0	0	100	0	0	0	0	0	0	0
area_6	0	0	0	0	100	0	0	0	0	0	0	0	0
area_7	100	0	0	0	0	0	0	0	0	0	0	0	0
area_8	0	0	0	0	0	0	0	0	0	25	0	0	0
area_9	0	0	0	0	0	0	0	0	23	2	0	0	0

Figure 5 : Matrice de confusion du traitement par CAH en remplaçant les NaN par -95 et les moyennes

Si on compare ces deux matrices on voit que la matrice de confusion en utilisant la moyenne est beaucoup plus fidèle à la réalité. On peut tester différentes

valeurs pour remplacer les données manquantes afin de trouver le meilleur cas possible.

J'ai donc fait des tests sur des valeurs plus ou moins aberrantes. En cherchant à remplacer les données manquantes par des valeurs plus extrêmes comme -110 ou -130 et par les moyennes permet d'obtenir des matrices de confusion parfaites avec chaque zone parfaitement représentée. Cependant elle ne permet pas d'être plus précise dans le cas où on remplace toutes les valeurs par -110 ou -130. J'ai étendu les différents calculs en utilisant 0 et -70 et on voit que dans les extrêmes (0,-130) on ne va pas avoir d'amélioration sur le remplacement de toutes les valeurs par ces valeurs mais en utilisant les moyennes en plus on obtient des représentations parfaites.

b. Etudes de K-means

En passant par les K-means on obtient dans le cas du remplacement des valeurs par -95 les résultats suivants.

Area	0	1	2	3	4	5	6	7	8	9	10	11	12
area_0	0	0	100	0	0	0	0	0	0	0	0	0	0
area_1	0	0	2	0	0	0	0	73	0	0	0	25	0
area_10	0	0	0	0	0	0	23	0	0	0	0	2	0
area_11	0	0	0	0	0	0	9	0	0	21	0	0	0
area_12	0	0	0	10	0	0	0	0	0	15	0	0	0
area_2	15	0	0	0	0	0	0	0	0	0	73	2	10
area_3	75	0	0	0	0	0	0	0	24	0	1	0	0
area_4	0	96	0	0	0	2	0	0	2	0	0	0	0
area_5	0	2	0	0	0	62	0	0	34	2	0	0	0
area_6	0	0	0	0	0	2	0	0	0	0	12	0	86
area_7	0	0	0	0	65	0	0	0	0	0	0	33	2
area_8	0	0	0	21	0	0	0	0	0	0	0	4	0
area_9	0	0	0	5	0	0	12	0	0	0	0	8	0

Figure 5 : Matrice de confusion du traitement par K-Means en remplaçant les NaN par -95

A partir des données récoltées du K-Means on obtient majoritairement les mêmes résultats que pour le traitement par CAH.

Grâce aux deux dernières études menées on peut déjà mieux comprendre les données qui ont été récoltées en menant une analyse sur un maximum de valeurs. On se rend compte ici qu'on ne peut pas vraiment être correct dans la détection à l'aide des traitements que nous avons effectués. Cependant on peut déjà noter l'importance qu'ont le remplacement des données manquantes car en choisissant des valeurs plus aberrantes on arrive à détecter la zone grâce aux zones qui n'ont pas été détectées et moins à la variation entre les données qu'on a récupéré. On en conclut donc qu'il est intéressant d'utiliser ces valeurs manquantes.

Tous les fichiers ayant servis à cette conclusion sont stockés dans le dossier "files". On y retrouve en plus de toutes les matrices de confusion dans ce rapport, les matrices qui ont servi en plus mais qui n'ont pas été montrées.

V. Utilisation des données

Afin de pouvoir détecter la zone dans laquelle on se trouve, on va entraîner plusieurs modèles et comparer leur résultat pour utiliser le meilleur des systèmes.

a. Entrainements des modèles

J'ai testé plusieurs modèles pour choisir le meilleur. Les trois modèles utilisés sont la forêt d'arbres décisionnelles (RFC), les k voisins les plus proches (KNN) et la machine à vecteur de support (SVM). Pour toutes les valeurs qu'on a sélectionné plus tôt, on a entraîné le modèle sur 80% des valeurs et on a laissé 20% des valeurs pour le test. On obtient donc le tableau suivant.

Strength	0	-70	-95	-110	-130
RFC	0.97311827	0.98387096	0.989247311	0.98924731	0.9892473
RFC with Mean	1.0	1.0	1.0	1.0	1.0
KNN	0.90860215	0.93010752	0.924731182	0.93010752	0.9247311
KNN with Mean	1.0	1.0	0.99462365	1.0	1.0
SVM	0.98924731	0.98924731	0.98387096	0.98924731	0.9892473
SVM with Mean	1.0	1.0	1.0	1.0	1.0

Figure 6 : Résultat précision des modèles

Grâce à ces données on se rend compte qu'on a toujours une très bonne précision pour tous les modèles. Malgré tout, on peut noter que l'algorithme de détection des K voisins les plus proches est moins précis que les autres.

b. Détection de zone

Pour la détection de zone j'ai décidé d'utiliser les trois modèles en les comparant. On pourra donc écarter les erreurs de certains algorithmes en servant les deux autres pour appuyer.

Pour travailler dans des conditions similaires aux précédentes analyses, on récupère tous les SSID disponibles dans le bâtiment et on utilise ces clés pour effectuer l'analyse.

Le code donne ainsi la possibilité de détecter la zone dans laquelle on se trouve en fonctionnant en temps réel ou données par données.

VI. Conclusion

Dans ce projet on a réussi à montrer qu'il était possible de détecter la zone dans laquelle on se trouve à l'aide de différents algorithmes tels que les k voisins les plus proches ou des forêts d'arbres décisionnelles. Dans la théorie, on arrive à avoir une exactitude aux alentours des 99%. La prochaine étape serait de montrer qu'il est aussi possible de le faire dans la pratique.