ISEN
BREST

Rapport de Projet A3

Partie Intelligence Artificielle

Du 10 juin au 14 juin

Avec Ehsan SEDGH GOOYA

PORCHER Jeanne

ROUILLE Alban

POURCHASSE Lana

Table des matières :

1. Visualisation sur carte

1.1. Préparation des Données :

Pour la préparation des données, nous sommes partis du fichier CSV déjà nettoyé, mis à disposition sur Moodle. Nous avons sélectionné pour la visualisation uniquement les coordonnées longitudinales et latérales, ainsi que la hauteur totale des arbres, afin de créer des clusters en fonction des tailles. Nous avons standardisé la hauteur totale des arbres pour qu'elle ait une moyenne de 0 et un écart-type de 1. La normalisation sert a optimisé le temps de calcule pour la suite des calcule

1.2. Apprentissage Non Supervisé :

Pour la partie apprentissage non supervisé, nous avons choisi deux algorithmes différents : K-Means et Agglomerative Clustering. Nous avons opté pour K-Means car il s'agit d'un algorithme simple et rapide à exécuter pour des petites et moyennes quantités de données. Toutefois, il nécessite de spécifier le nombre de clusters attendu pour fonctionner correctement. Nous avons également choisi l'algorithme Agglomerative Clustering car il fonctionne bien avec les données numériques continues, comme la taille des arbres. Cependant, il peut être lent pour les grandes quantités de données.

En résumé, K-Means est préféré pour sa simplicité et son efficacité sur des volumes de données petits à moyens, tandis qu'Agglomerative Clustering est choisi pour sa capacité à traiter efficacement des données numériques continues malgré sa lenteur sur des ensembles de données volumineux.

Le principe de K-Means repose sur un processus itératif d'assignation et de mise à jour des centroïdes. Pour cela, on commence par choisir k centres de clusters initiaux (centroïdes) de manière aléatoire. Ensuite, on attribue chaque point de données au centroïde le plus proche. Puis, on met à jour les centroïdes en calculant la moyenne des points assignés à chaque cluster. Les étapes d'assignation et de mise à jour sont répétées jusqu'à ce que les centroïdes se stabilisent ou qu'un nombre maximal d'itérations soit atteint.

L'agglomérative clustering, également appelé clustering hiérarchique ascendant, est une méthode de regroupement qui construit une hiérarchie de clusters en fusionnant progressivement les plus similaires. On commence par considérer chaque point de données comme un cluster individuel. Puis on trouve les deux clusters les plus proches (selon une mesure de similarité ou de distance) et les fusionne en un seul cluster. On met à jour la matrice de distance pour refléter les nouvelles distances entre le nouveau cluster formé et les autres clusters existants. On répète les étapes de fusion et de mise à jour jusqu'à ce qu'il ne reste plus qu'un seul cluster ou jusqu'à atteindre un nombre désiré de clusters.

1.3. Métriques pour l'Apprentissage Non Supervisé :

La **Méthode du coude** est la somme des distances entre chaque arbre et le centre de son cluster (centroïde) est utilisée pour évaluer la qualité des clusters. Lorsque le nombre de clusters augmente, ces distances diminuent car les points sont plus proches de leur centre de cluster. Cependant, après un certain point, augmenter encore le nombre de clusters réduit beaucoup moins la somme des distances. Cela signifie que la diminution de la variance intra-cluster n'est plus significative.

Le **score de silhouette** est calculé en prenant la différence entre la distance moyenne d'un point à tous les autres points dans son cluster et la distance moyenne d'un point à tous les points d'un autre cluster voisin. Cette différence est ensuite divisée par la plus grande valeur entre ces deux distances moyennes. En résumé, le score de silhouette mesure à quel point un point est proche de son propre cluster par rapport aux autres clusters voisins, normalisé par la distance maximale entre les clusters.

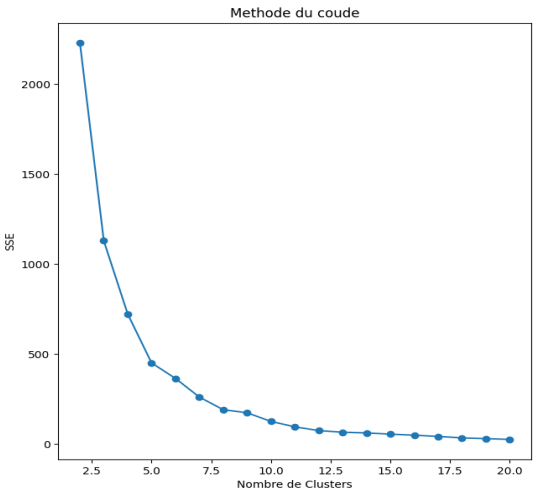


Figure 1 : Application de la méthode du coude (à gauche) | Figure 2 : Application du score de silhouette (à droite)

1.4. Visualisation sur Carte :

Pour faire la visualisation nous avons utilisé la bibliothèque Plotly qui nous a permis d'avoir un beau rendu avec notamment des boutons qui nous permette de changer le graphique que nous voulons afficher. L'utilisateur a 6 choix différent soit 2 – 3 ou le nombre choisis par l'utilisateur de cluster calculer avec Kmean ou alors 2 – 3 ou le nombre choisis par l'utilisateur de cluster calculer avec agglomerative clustering. Ce qui nous permet d'obtenir la figure3.

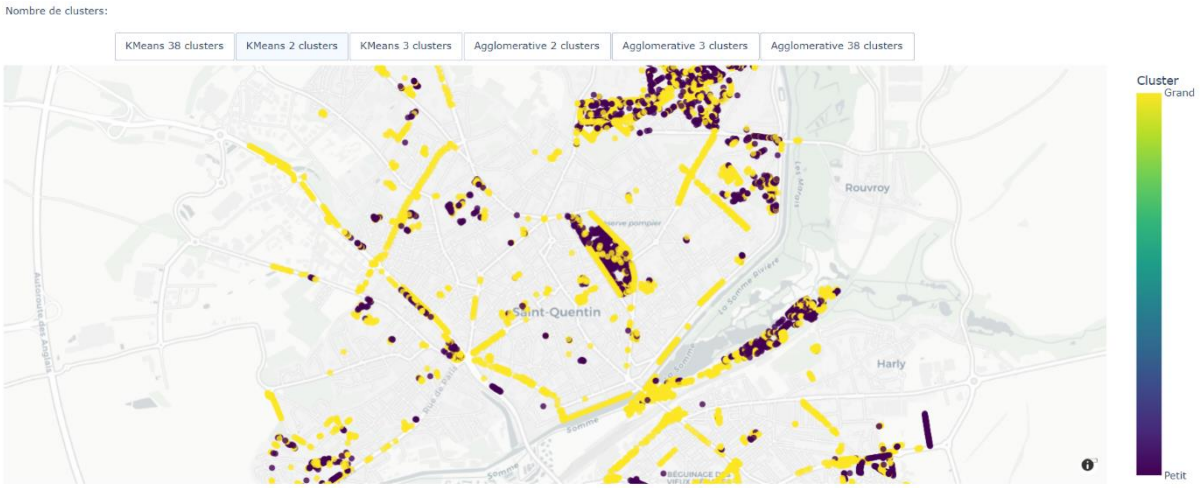
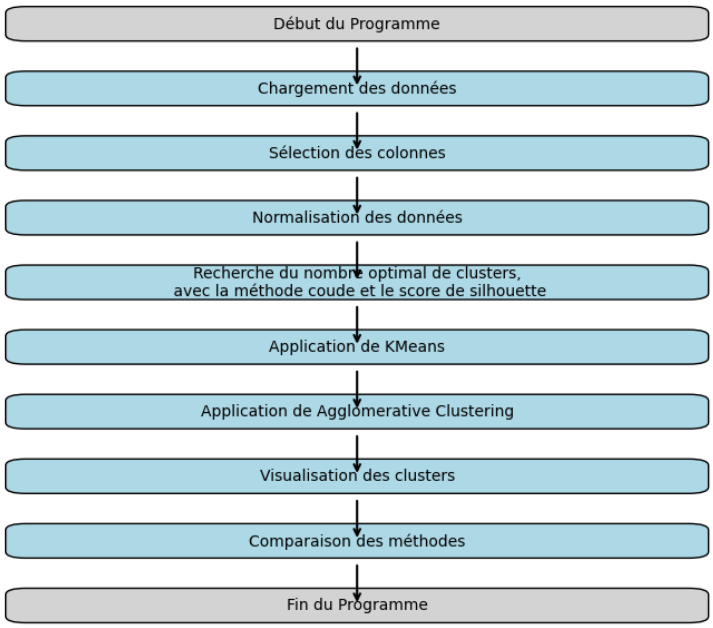


Figure 3 : Visualisation de la carte par cluster de taille

1.5. Préparation du Script :



2. Modèle de prédiction de l'âge

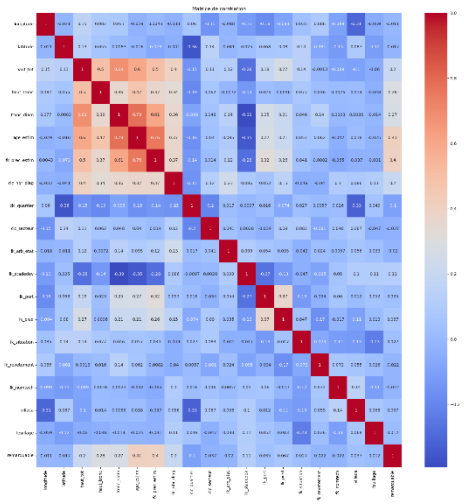
2.1. Préparation des données et choix des variables

Dans un premier temps, nous avons importé les données et effectué une analyse initiale pour comprendre leur structure et leur appliquer les transformations nécessaires telles que l'encodage des variables qualitatives et la normalisation des variables quantitatives.

Choix des variables

Les variables sélectionnées pour le modèle ont été choisies sur la base de la matrice de corrélation de l'ensemble des données ainsi que sur un test ANOVA, permettant de confirmer la p-value entre chacune des valeurs choisies dans un premier temps :

Variable dépendante	Variable indépendante	F	PR(>F)
tronc_diam	remarquable	571.121551	1.175721e-121
tronc_diam	fk_stadedev	1830.78866	0.0
haut_tot	remarquable	310.718357	3.702249e-68
haut_tot	fk_stadedev	1069.599566	0.0
haut_tronc	remarquable	642.546436	5.144828e-136
haut_tronc	fk_stadedev	406.653671	1.542156e-244
fk_prec_estim	remarquable	1439.782401	4.591393e-288
fk_prec_estim	fk_stadedev	1271.139109	0.0
clc_nbr_diag	remarquable	298.482957	1.345018e-65
clc_nbr_diag	fk_stadedev	441.630838	5.738068e-264
age_estim	remarquable	1504.874825	7.294479e-300
age_estim	fk_stadedev	2153.490296	0.0



Sur la base d'une corrélation supérieure à |0.3|, La figure 4 nous indique que les variables utiles à la régression sont :

Variables qualitatives encodées :

- Remarquable
- Stade de développement

Variables quantitatives normalie l'arbre

- Hauteur du tronc
- Alignement de l'arbre
- Diamètre du tronc
- Hauteur totale du tronc

Figure 4 : Matrice de corrélation des données encodées et normalisées

2.2. Modèle de prédiction et fonctionnement

2.2.1. Modèle de prédiction

Après avoir parcouru la documentation, nous avons choisi 3 modèles d'apprentissage supervisé :

- Régression KNN (KNeighborsRegressor)
- Gradient Boosting (GradientBoostingRegressor)
- Forêt aléatoire (RandomForestRegressor)

2.2.2. Principe de fonctionnement

Chaque modèle utilise différentes techniques pour estimer l'âge en fonction des variables d'entrée :

- KNeighborsRegressor est basé sur des instances, c'est-à-dire qu'il prédit la valeur cible pour un nouveau point de données en utilisant les valeurs cibles des k plus proches voisins.
- GradientBoostingRegressor est basé sur le principe de l'addition séquentielle de modèles faibles (typiquement des arbres de décision) pour corriger les erreurs des modèles précédents.
- RandomForestRegressor utilise un ensemble d'arbres de décision construits de manière aléatoire et moyennés pour faire des prédictions, ce qui aide à réduire la variance et à améliorer la généralisation.

2.3. Justification du choix du modèle

Nous avons comparé les performances des différents modèles en utilisant 3 des métriques les plus efficaces dans le cas d'une régression : l'erreur quadratique moyenne (MSE), le score R^2 et l'erreur absolue moyenne (MAE). Le modèle choisi a été sélectionné en termes de performances globales.

Tableau des performances des modèles

On observe dans un premier temps que le GradientBoostingRegressor semble être le plus performant. Les R^2 sont également assez hauts, cela pourrait être lié à un phénomène de sur-apprentissage.

Modèle	MSE	R^2	MAE
KNeighborsRegressor	0.001710	0.8400	0.0239258
GradientBoostingRegressor	0.001459	0.8635	0.0281313
RandomForestRegressor	0.001377	0.8712	0.0234484

On observe sur la figure 2 que le GradientBoostingRegressor est le plus efficace pour le R^2 mais le RandomForestRegressor est meilleur sur le MSE et le MAE.

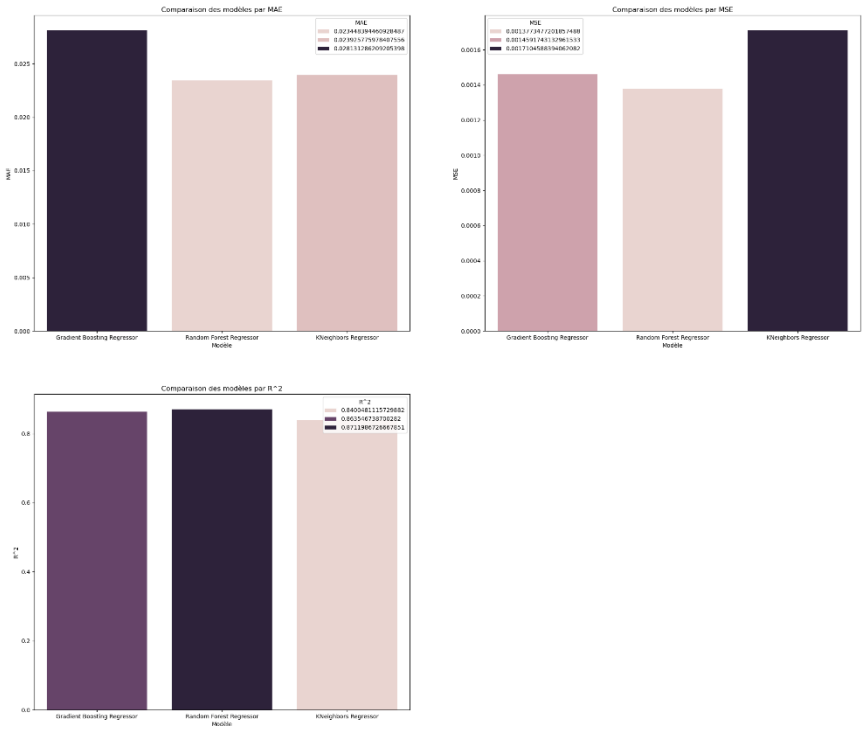


Figure 5 : Comparaison de l'efficacité de chaque modèle avant GridSearch

2.4. Ajustements et fine-tuning

Nous avons affiné les hyperparamètres des modèles en utilisant GridSearchCV pour optimiser les performances en utilisant une technique de cross-validation 3-folds. Cette étape, cruciale pour éviter le sur-apprentissage nous conforte dans le choix du modèle.

Tableau des performances des modèles

Modèle	MSE	R^2	MAE
KNeighborsRegressor	0.002029	0.7993	0.025842
GradientBoostingRegressor	0.001396	0.8620	0.024520
RandomForestRegressor	0.001419	0.8596	0.024506

L'ensemble des modèles affinés ont des résultats similaires (figure 3). On choisit tout de même le GradientBoostingRegressor qui obtient les meilleurs scores au global.

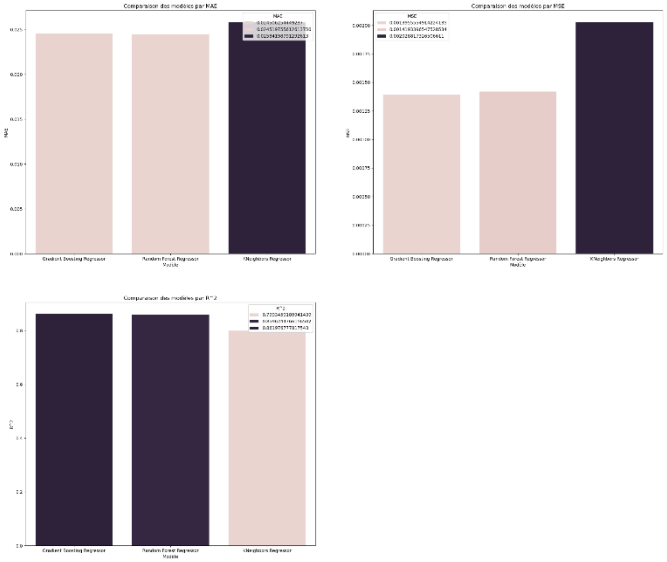
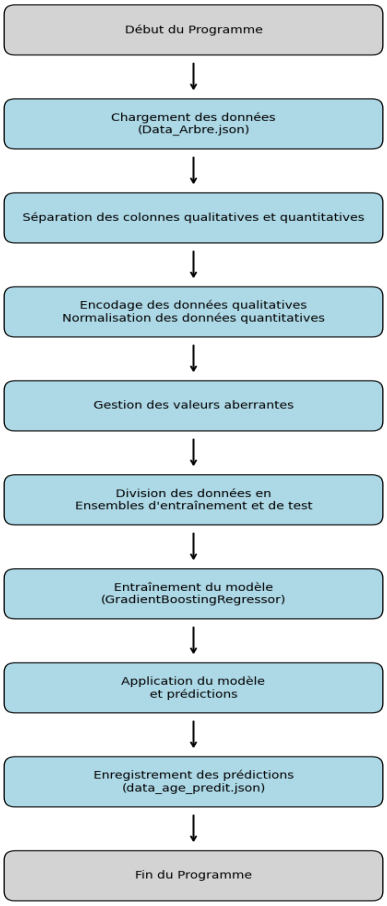


Figure 6 : Comparaison de l'efficacité des modèles après GirdSearch

2.5. Préparation du script

Avant de commencer le script, nous avons réalisé un schéma bloc afin de mettre en ordre les fonctionnalités et leur ordre d'exécution.



3. Système d’alerte pour les tempêtes

3.1. Etudes des distributions

La colonne cible pour le besoin 3 est fk_arb_etat. Cette colonne est composée de plusieurs variables distinctes dont la répartition est visible sur la figure 7.

On y observe que seulement les valeurs ‘essouché’ et ‘non essouché’ correspondent à l'état des arbres à la suite d'une tempête. Cela implique que notre modèle ne serait pas équilibré dans la distribution comme le montre la figure 8, 165 / 27 données. Une telle quantité de données est largement insuffisante, ce qui le rendrait biaisé envers la classe majoritaire du fait de la recherche d'équilibre du modèle.

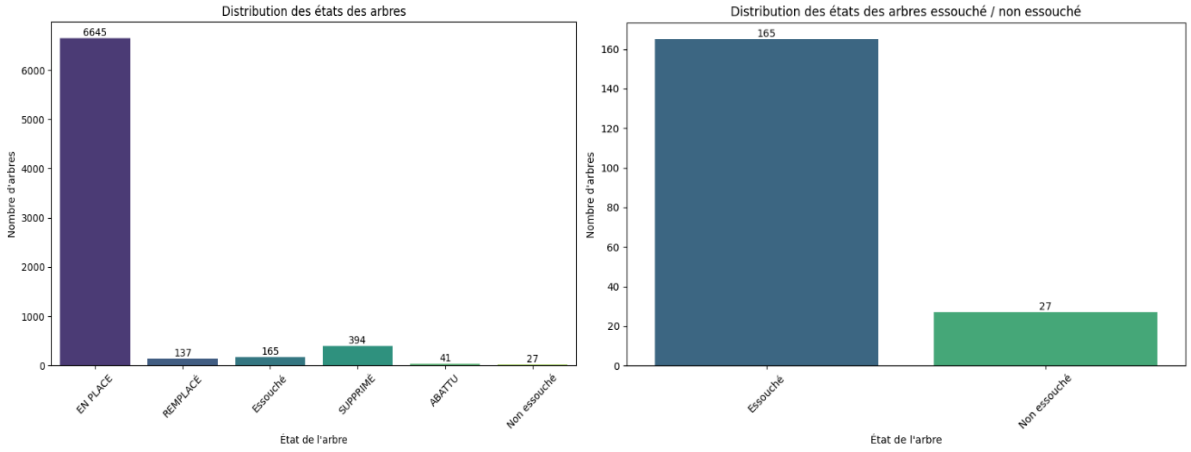


Figure 7 : Distribution de l'état des arbres des données de Data_arbre.csv (à gauche) |

Figure 8 : Distribution de l'état des arbres essouchés et non essouchés (à droite)

Afin de réparer ce déséquilibre, nous avons décidé d'ajouter les arbres abattus en tant qu'arbres non essouchés. En effet, il est logique de penser que si un arbre est abattu cela peut être dû à une chute ou un mauvais état de l'arbre à la suite d'une tempête ou d’un coup de vent violent. De plus, lors de l'abattage il est rare qu'en pratique la souche de l'arbre soit retirée, dans le but de gagner du temps. Comme le montre la figure 9, l’équilibre de notre répartition s'est amélioré mais n'est toujours pas parfait, dans le but d'améliorer cet équilibre nous avons utilisé une technique de rééquilibrage. La technique d'undersampling ne peut pas s'appliquer dans notre cas car notre jeu de donnée est déjà petit et nous ne pouvons pas nous permettre de perdre en information.

C'est pour cette raison que la technique que nous avons décidé d'utiliser est celle de l'oversampling qui consiste à augmenter la taille de la classe minoritaire par création de nouvelles données.

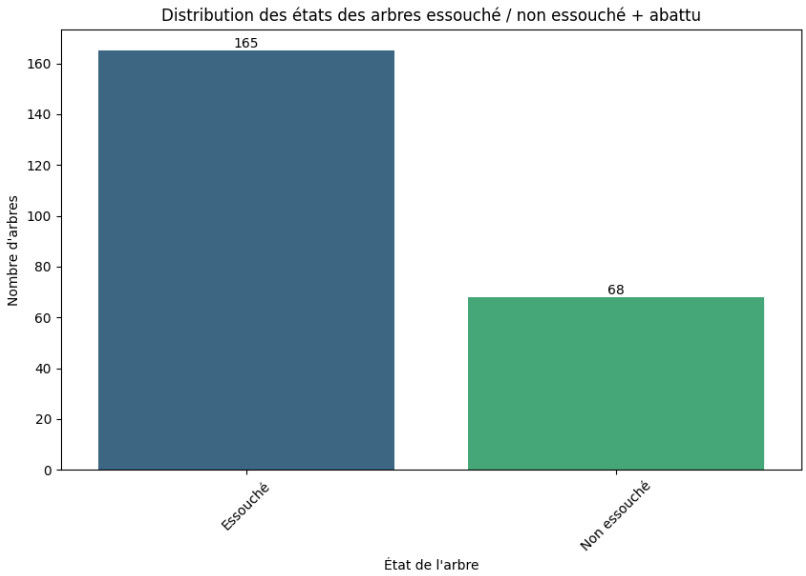


Figure 9 : Distribution de l'état des arbres essouchés et non essouchés abattus et en place

3.2. Choix des variables d'intérêts

Afin de choisir correctement nos variables d'intérêts pour ce besoin, nous avons encodé nos données catégorielles puis réaliser une heatmap de corrélation (figure 4)**.** D'après celle-ci, nous pouvons voir que les colonnes en forte corrélation avec fk_arb_etat sont :

- tronc_diam avec une corrélation à -0.24
- age_estim avec une corrélation à -0.23
- clc_nbr_diag avec une corrélation à -0.24
- latitude avec une corrélation à 0.38
- haut_tot avec une corrélation à -0.24
- haut_tronc avec une corrélation à -0.23

De plus, nous avons réalisé un test de khi-deux afin de vérifier que ces variables soient explicatives de fk_arb_etat (figure 10).

	Feature	Khi2	P-value
6	tronc_diam	4.759295e+02	1.642417e-105
12	age_estim	1.827038e+02	1.244805e-41
3	clc_secteur	8.857534e+01	4.893526e-21
4	haut_tot	5.195793e+01	5.670262e-13
15	fk_nomtech	3.954746e+01	3.201822e-10
5	haut_tronc	2.794441e+01	1.248514e-07
14	clc_nbr_diag	2.062236e+01	5.593884e-06
13	fk_prec_estim	1.595636e+01	6.481975e-05
11	fk_revetement	1.780829e+00	1.820476e-01
18	remarquable	1.648485e+00	1.991653e-01
10	fk_situation	1.607501e+00	2.048434e-01
17	feuillage	1.345603e+00	2.460483e-01
7	fk_stadedev	1.037957e+00	3.082971e-01
9	fk_pied	7.161536e-01	3.974085e-01
2	clc_quartier	4.098115e-01	5.220652e-01
8	fk_port	2.457714e-01	6.200685e-01
16	villeca	8.595807e-03	9.261311e-01
1	latitude	6.750566e-05	9.934445e-01
0	longitude	8.077111e-07	9.992829e-01

Figure 10 : Résultats du test du khi-deux

3.3. Encodage des données catégorielles

Dans le but d'utiliser nos différents algorithmes de machines Learning nos données doivent être encodés, après analyse de notre jeu de données nous sommes partis sur l'idée d'encodé toutes nos données grâce au Label Encoder. Le Label Encoder est le plus adapté du fait du grand nombre de catégories dans certaines colonnes comme clc_secteur.

3.4. Normalisation des données numériques

Les différentes variables n'étant pas à la même échelle il convient de les normaliser afin de les mettre sur la même échelle et de faciliter l'apprentissage des modèles de machines Learning.

Pour la normalisation nous avons utilisé la méthode des min-max, il s'agit de la méthode la plus commune et permet de ramener les valeurs entre 0 et 1.

3.5. Jeu de données et entrainement de modèles pour la classification

Afin de réaliser notre classification, nous avons divisé notre jeu de donnée entre deux parties, une pour l'entraînement du modèle, qui représente 80% du jeu de donnée d'origine et une autre de test pour vérifier que notre modèle fonctionne correctement.

Afin d'équilibrer la distribution de notre jeu d'entrainement nous avons utilisé de l'oversampling avec la méthode du Synthetic Minority Oversampling Technique (SMOTE). Cette méthode permet de créer de nouvelles données en réalisant une interpolation entre les 5 échantillons voisins d'un point et ainsi obtenir un nouveau point.

Sur ce jeu de données à la distribution équilibrer nous avons entrainé 5 modèles de machines Learning qui correspondait à notre objectif et jeu de données. Ces modèles sont le LogisticRegression, le DecisionTree, les KNeighbors, et le GradientBoosting.

Nous avons réalisé plusieurs métriques sur ces modèles afin de vérifier quel modèle est le plus performant. Dans un premier temps nous utilisions les matrices de confusion afin de vérifier le taux de faux positif, faux négatif de notre modèle.

Puis nous avons décidé d'utiliser les divers scores comme l'accuracy, précision, rappel, f1-score afin de pouvoir comparer nos modèles. De plus, nous utilisons de l'AUC pour mesurer les performances des différents modèles, plus on est proche de 1 plus le modèle est capable de réaliser une séparation des classes.

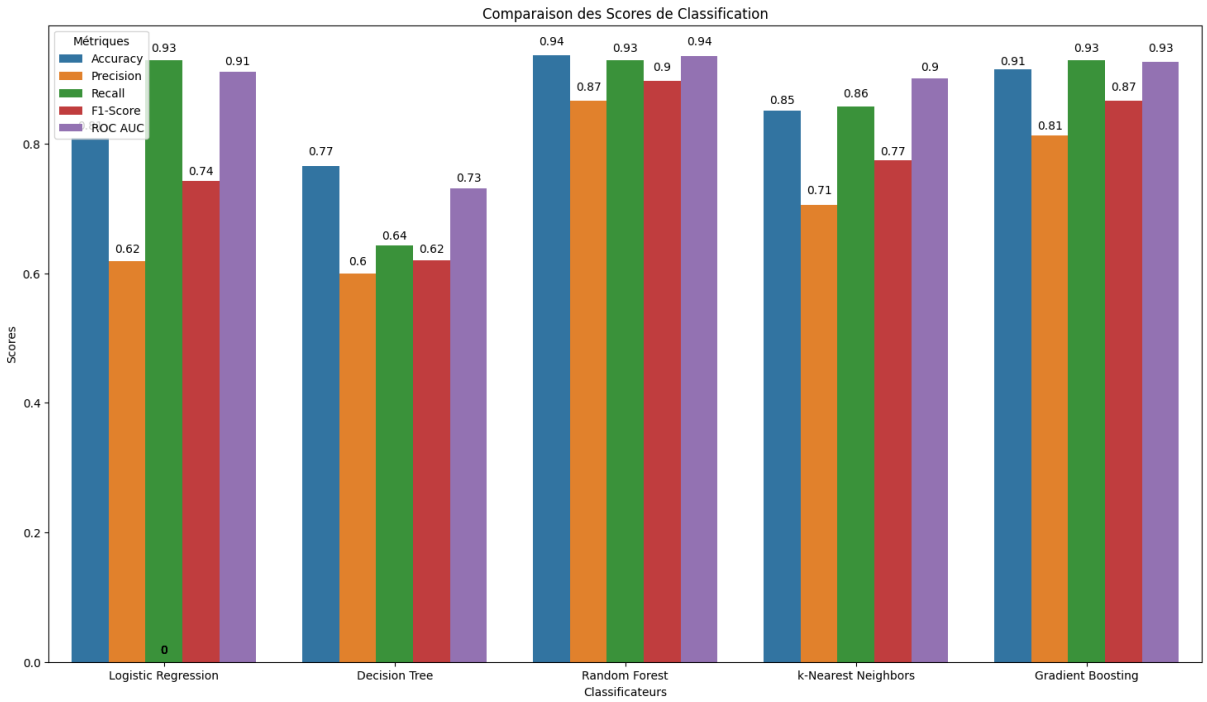


Figure 11 : Comparaison de la précision de chaque

3.6 Grid Search des modèles les plus performants

Afin de trouver les meilleurs paramètres pour les 3 modèles les plus performants et ainsi trouver notre modèle le plus performant à utiliser pour notre système d'alerte, nous avons réalisé un Grid Search sur le RandomForest, le KNearest Neighbors, et le Gradient Boosting.

Malheureusement les résultats de ce GridSearch ne furent pas concluants, comme le montrent les matrices de confusion des figures 12 à 14. Ces matirices de confusion montrent que les meilleurs modèles ont un fort taux de faux positif et de faux négatif ce qui n'est pas bon pour avoir un modèle performant et fonctionnel.

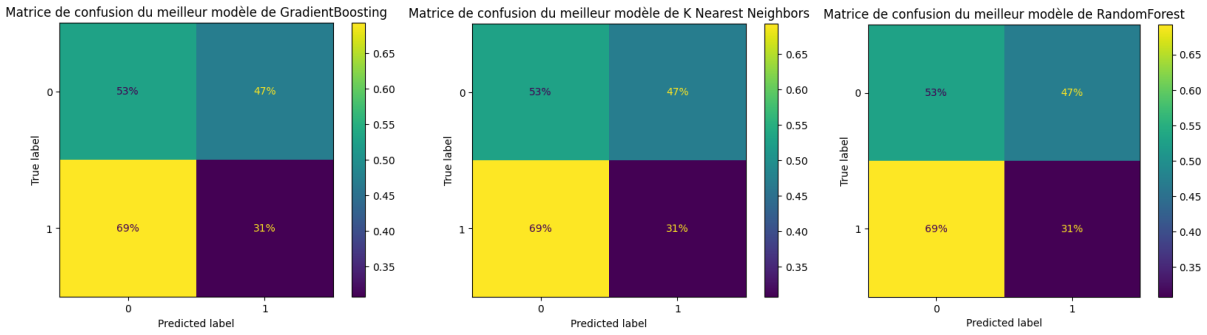


Figure 12, 13, 14: Matrice de confusion de Gradient Boosting, KNearest Neighbors et RandomForest

3.7 Carte avec les arbres essouchés ou non par la tempête

Du fait de nos problèmes de faux positif et faux négatifs sur nos modèles, nos prédictions sont incorrectes et cela se remarque fortement sur la carte de la figure 15.

4101 arbres sont considérés comme essouché par la tempête et 2544 non sont pas essouché par la tempête, ce qui est totalement aberrant.

Ce problème de modèle s'explique par la faible quantité de données pour entrainer notre modèle, malgré l'utilisation de techniques comme l'oversampling, notre modèle n'est pas capable de prédire correctement l'état d'un arbre à la suite de la tempête.

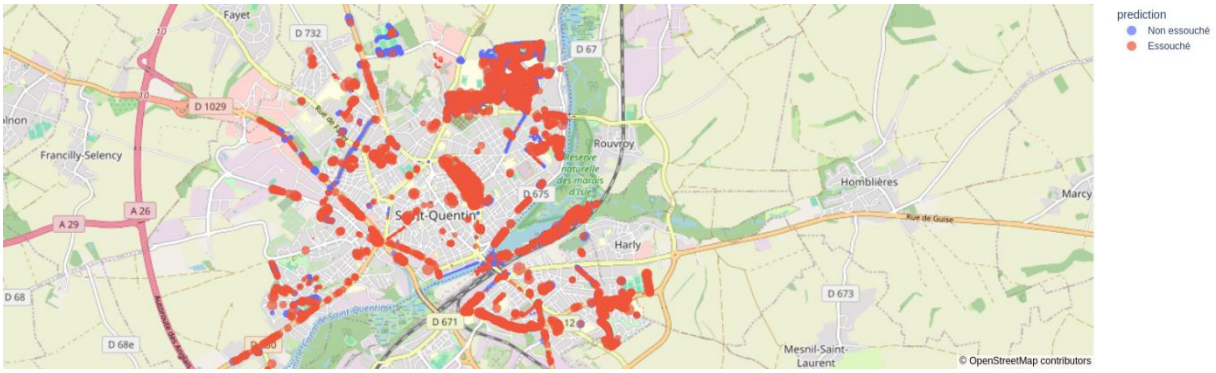
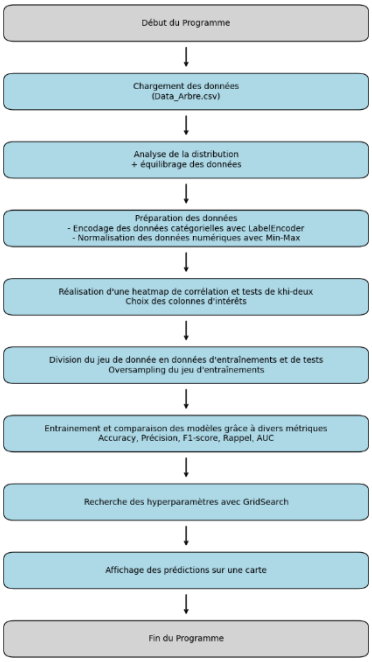


Figure 15: carte de prédiction de l'état des arbres à la suite d'une tempête

3.8 Préparation du script



4. Annexe – Diagramme de Gantt