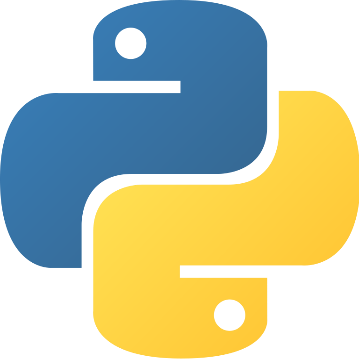
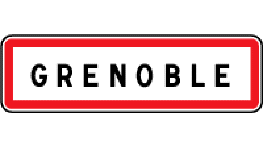
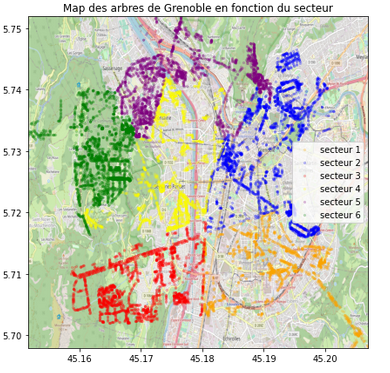
ECOLE DU CONSERVATOIRE NATIONAL DES ARTS ET METIERS (EICNAM – NOUVELLE-AQUITAINE)



**UE Entreposage et Fouille de données Défi EGC’2017**



****

ADRIEN GOLEBIEWSKI – MAXIME CHARRUYER

PROMOTION N°2

Table des matières

[**Introduction - Présentation du contexte** 3](#_Toc68980358)

[**1 - Aperçu des données et Data Préparation** 4](#_Toc68980359)

[1.1 - Aperçu des données 4](#_Toc68980360)

[1.2 - Data Cleaning – Préparation des données 5](#_Toc68980361)

[1.2.1 – Aperçu et gestion des données manquantes 5](#_Toc68980362)

[1.2.2 - Modification du typage et du format des données 8](#_Toc68980363)

[1.2.3 - Suppression et création de nouvelles variables 10](#_Toc68980364)

[**2 – Analyse exploratoire des données** 11](#_Toc68980365)

[2.1 – Analyse statistique des données 11](#_Toc68980366)

[2.2 – Analyse par mapping des données 16](#_Toc68980367)

[**3. Prédiction de l’existence d’un défaut d’un arbre – Classification binaire** 18](#_Toc68980368)

[3.1 – Traitement préliminaires 18](#_Toc68980369)

[3.2 – Définition des classifieurs, choix des hyperparamètres et des métriques d’évaluation 21](#_Toc68980370)

[3.2.1 – Définition des classifieurs de prédiction 21](#_Toc68980371)

[3.2.2 – Choix des hyperparamètres des classifieurs 22](#_Toc68980372)

[3.2.3 - Choix des métriques d’évaluation et gestion du sur-apprentissage 23](#_Toc68980373)

[**4. Prédiction de la localisation d’un défaut d’un arbre – Classification multi-labels** 29](#_Toc68980374)

[**5. Conclusion et mise en perspectives** 34](#_Toc68980375)

**Introduction - Présentation du contexte**

La conférence EGC (Extraction et Gestion des Connaissances) est un événement annuel réunissant des chercheurs et praticiens de disciplines relevant des sciences des données et des connaissances. Ces conférences permettent en autres de démocratiser la Data Science auprès d’industriels et en parallèle de répondre à des problématiques Data concrètes.



En 2017, EGC’2017 a proposé un défi dont le contexte est la gestion des espaces verts pour la ville de Grenoble, et notamment des arbres qui y sont présents.

La 1ère tâche vise à prédire si les arbres présentent des défauts ou non, est un problème de classification supervisée. La seconde tâche est quant à elle axée sur une meilleure connaissance de l’état ainsi que de l’évolution du "parc végétal" de Grenoble.

L’objectif est ainsi de proposer un modèle basé sur des données fournies qui permettrait de prédire au mieux les arbres malades, ainsi que la localisation potentielle de la maladie. Des préconisations pour les futures plantations d’arbre dans la ville de Grenoble sont également à définir. Des modèles de machine learning (IA) sont donc à prévoir pour réussir les différentes tâches.



Le travail présenté dans ce rapport tente de répondre au Défi EGC 2017, dont l’enjeu est, pour la ville de Grenoble, de prédire l’apparition de maladies parmi les arbres du parc urbain. Il a été réalisé entièrement à l’aide du langage informatique Python, spécialisé dans le domaine de la Data Science. Les librairies « Numpy », « Pandas » et « scikit Learn » ont été employées pour l’analyse, la gestion de données et le développement d’algorithmes ML de prédiction. Les librairies « Matplotlib » et « Seaborn » ont été employées pour l’aspect visualisation. Le code a été lancé à travers le notebook Jupyter.

Dans un premier temps, les données mises à disposition par les organisateurs du défi ont été nettoyés, modifiés à travers une phase de Data Cleaning afin d’obtenir des données exploitables pour la suite de nos démarches. Ensuite, les données en résultant ont été analysés par mapping et par analyses statistiques de manière à avoir une vision globale de leurs caractéristiques Dans un second temps, quelques algorithmes de classification sélectionnés ont été testés et évalués sur les données afin de répondre au défi n°1.

Enfin, nous avons tenté de répondre au défi n°2 en proposant des recommandations de la gestion des espaces verts de Grenoble.

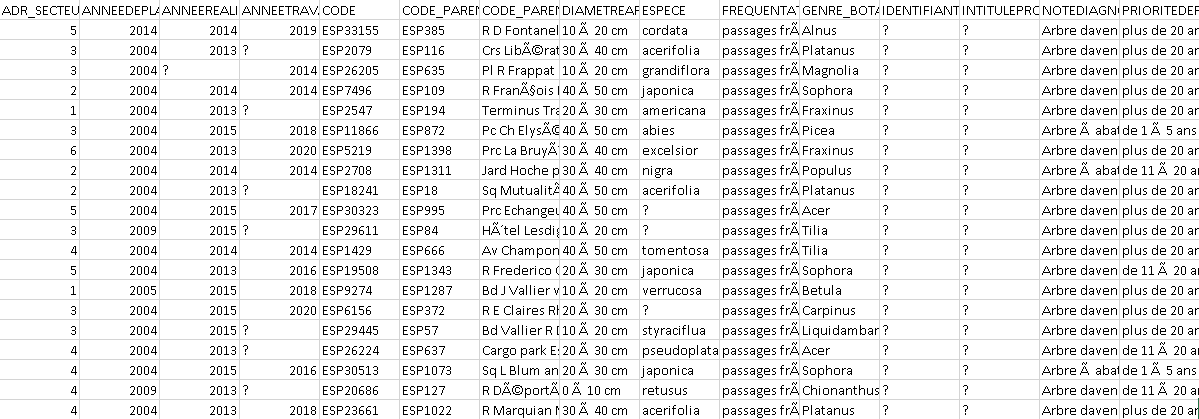
# **1 - Aperçu des données et Data Préparation**

Avant d’entamer des modèles de prédiction sur notre ensemble de données d’arbres, il est nécessaire de réaliser des étapes en amont pour préparer notre ensemble de données et d’en avoir un aperçu rapide.

## 1.1 - Aperçu des données

Les données mises à notre disposition sont sous format Excel dans un fichier XLS.

Le jeu de données est constitué de 15 375 arbres.



Chaque arbre possède 29 variables qui le décrivent, et 5 variables à prédire, dont une qui reflète si l’arbre est sain ou pas (’Default or not’ prenant les valeurs 0 ou 1), et 4 variables

spécifiant les endroits atteints (’Collet’, ’Houppier’, ’Racine’, ’Tronc’ prenant également les

valeurs 0 ou 1).

Certains attributs décrivent l’arbre (Code, DiamètreArbreÀUnMètre, Année- DePlantation, Espèce, Genre\_Bota. . .), son emplacement (Adr\_Secteur, Trottoir, FréquentationCible,) des informations établies à l’occasion de diagnostics (AnnéeRéalisationDiagnostic, NoteDiagnostic, AnnéeTravauxPréconisésDiag, Remarques. . .)

Parmi les données fournies, la géolocalisation de chaque arbre était présente à travers 2 coordonnées, x et y.

## 1.2 - Data Cleaning – Préparation des données

Avant de procéder à la modélisation des données, il est nécessaire de passer par un nombre d’étapes préliminaires qui sont essentielles à un projet de **machine Learning**. Bien que fastidieuses, il ne faut surtout pas les mettre de côté car elles sont garantes de la qualité de notre modélisation. Un modèle dont la qualité des données n’est pas optimale, ne pourra pas réaliser de bonnes prédictions.

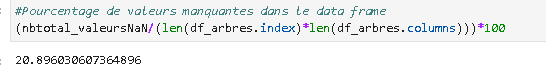
Les étapes préliminaires, aussi appelées étapes de **data pre-processing**, recouvrent ainsi les étapes de vérification de l’état, du nettoyage – **data cleaning** – et de la préparation de données. – **data preparation**.

Ce processus de « data cleaning » et de « data preparation » est donc à appliquer sur notre ensemble de données dans la partie précédente.

Nous avons tout d’abord traité la gestion des données manquantes de notre data set et ensuite corrigé ce dernier en apportant des modifications.

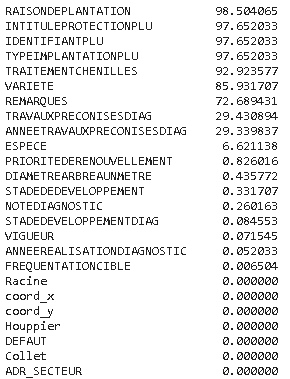
### 1.2.1 – Aperçu et gestion des données manquantes

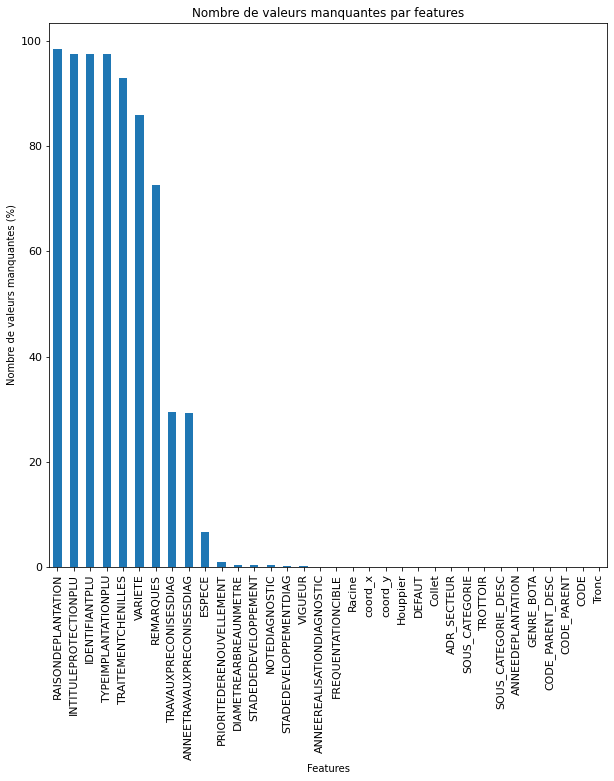
Dans l’ensemble de notre Data Frame, il y a 20,9 % de valeurs manquantes.



On note en particulier que de nombreuses variables comportent beaucoup de valeurs manquantes ou plus communément appelées des « NaN values ». Plus de 99% par exemple pour la variable « RAISONDEPLANTATION », 98% pour « IDENTIFIANTPLU », « TYPEIMPLANTATIONPLU » et 96% pour « VARIETE ».

Ainsi, certaines variables sont constituées majoritairement de valeurs manquantes « NaN » comme on peut le constater à travers le tableau et l’histogramme ci-dessous :





Le remplacement des NaN values a été long et surtout fastidieux car il a été réalisé sur l’ensemble des variables constituées de ce type de valeurs. L’intérêt ici n’est de pas présenter notre technique employée pour chacune des 29 variables mais de vous définir les deux types de gestion des NaN Values que nous avons suivi :

* **Un remplacement simple** des NaN values par une autre valeur en utilisant la fonction « replace » de la librairie « Pandas ».

Par exemple, pour les variables « ANNEEREALISATIONDIAGNOSTIC »,

« ANNEETRAVAUXPRECONISESDIAG », «IDENTIFIANTPLU » : les valeurs manquantes ont été remplacées par 0 afin de garder la présence d’une non-information qui pourrait apparaître comme un signal faible.



Ou encore, en remplacement les valeurs manquantes par la médiane des valeurs des variables impliquées. C’est notamment le cas pour les variables « ANNEETRAVAUXPRECONISEDIAG » et « ANNEEREALISATIONDIAGNOSTIC ».



* **Un remplacement complexe des NaN values.**

La variable DiamètreArbreÀUnMètre a, quant à elle, été transformée en attribut numérique en considérant le minimum de l’intervalle. Un travail de « split » des données de la variable a été effectué afin de transférer en liste celles-ci.

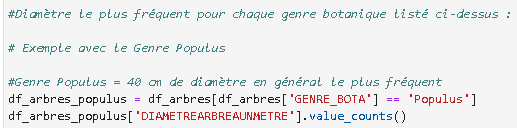


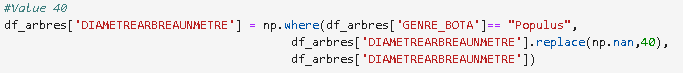
Ensuite une récupération des premières valeurs de chacune des listes a été effectuée.



Pour les variables « DIAMETREARBREAUNMETRE » et « ESPECE » : les valeurs manquantes ont été remplacées par le mode en fonction du genre botanique de l’arbre (GENRE\_BOTA), autrement dit le diamètre le plus fréquent et l’espèce la plus fréquente par rapport au genre botanique de l’arbre en question.

Un travail long et fastidieux a donc été lancé pour pouvoir tester les GENRES\_BOTA pour lesquels les « Espèces » et « DIAMETREARBREAUNMETRE » des arbres ne sont pas définies. Puis pour chaque GENRE\_BOTA, nous avons trouvé l'espèce de l'arbre et DIAMETREARBREAUNMETRE la plus fréquent, valeur que nous avons sélectionnée pour remplacer la NaN Value selon le GENRE\_BOTA de l'arbre.





### 1.2.2 - Modification du typage et du format des données

Un certain nombre de champs (7 au total) possède des valeurs numériques (

ANNEEDEPLANTATION, ANNEEREALISATIONDIAGNOSTIC, ANNEE

TRAVAUXPRECONISESDIAG, IDENTIFIANTPLU, coord\_x, coord\_y) pour lesquels aucun

Formatage n’est nécessaire.

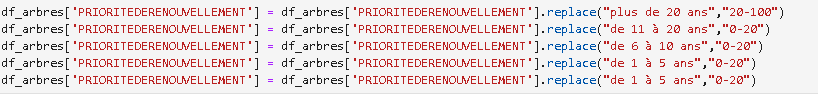
Des conversions ont cependant été effectuées pour améliorer l’exploitabilité du corpus. Ainsi, l’attribut Adr\_Secteur (figure 1), dont les valeurs sont des nombres compris entre 1 et 6, sans relation d’ordre particulière, a été transformé en attribut nominal. De même, les variables de classe Défaut, Collet, Houppier, Racine, Tronc, à valeurs binaires, ont été transformées en attributs nominaux (cela permet d’appliquer des programmes de classification).

La variable DiamètreArbreÀUnMètre a, quant à elle, été transformée en attribut numérique en considérant le minimum de l’intervalle.

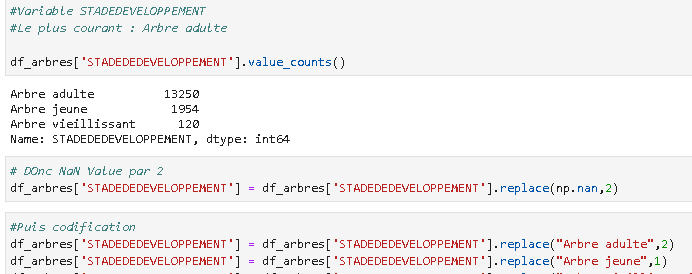
Les valeurs de la variable PrioritéDeRenouvellement, nominales au départ, ont été codifiées

sous forme d’intervalles (exemple : la valeur "moins de 5 ans" est transformée en intervalle

1-5).



Aussi, nous avons codifié les variables « StadeDeDéveloppement » et « StadeDévelopementDiag » par les entiers 1, 2 et 3, correspondant respectivement aux valeurs "arbre jeune", "arbre adulte" et "arbre vieillissant". Le remplacement des NaN value a été réalisé en fonction du nombre le plus élevé de type de stade.



Cette codification sera en autres bénéfique pour la création d’une variable supplémentaire dans notre data set d’origine.

Enfin, la variable « Remarques » nous a posé problématique. L’attribut « Remarques» comporte du texte libre, décrivant des informations notées sur les arbres, par des techniciens ou botanistes, lors des diagnostics.

Malheureusement, avec 1 684 valeurs, dont 1 291 valeurs uniques (ne concernant qu’un arbre), cet attribut est difficilement exploitable en l’état actuel. Nous distinguons trois façons de considérer cet attribut :

* Les valeurs de l’attribut sont laissées telles quelles
* On définit s’il y a une présence ou non d’une remarque
* On conçoit un algorithme de classification qui définit dans un premier temps les différents thèmes abordés dans chacune des chaînes de caractère et qui créée une variable binaire pour chacun de ces thèmes définissant une présence ou une absence pour un arbre donné. Les valeurs de l’attribut seraient ainsi transformées en vecteurs de mots.

Avec nos connaissances limitées en modèle de classification et de vectorisation de données, nous avons abandonné cette option en choisissant d’exploiter la présence ou non d’une remarque lors d’un diagnostic pour un arbre donné.

Cela s’est effectué par la fonction « notna » qui renvoie un objet booléen de même taille indiquant si les valeurs ne sont pas NA.



Puis en « castant » les données, nous obtenons une variable binaire avec la variable « 1 » signifiant la présence d’une remarque dans la fiche de diagnostic et « 0 » si elle est absente.



### 1.2.3 - Suppression et création de nouvelles variables

Le travail de Data Management ne se résume pas par la gestion des données manquantes. Dans le contexte de notre étude, la pertinence de la présence de certaines variables peut être remises en question. A l’inverse, nous avons trouvé pertinent de créer des variables supplémentaires qui nous ont paru intéressantes à ajouter par la suite dans nos modèles.

Les variables Code\_Parent et Code\_Parent\_Desc sont redondantes, et décrivent, pour un arbre, la station parente à laquelle il est rattaché. Ces deux variables ont été supprimées du corpus pour deux raisons. D’une part, le nombre de valeurs distinctes est très important (près de 1 140). D’autre part, l’information de la station d’un arbre, étant de nature géographique, elle est répétitive avec les variables Adr\_Secteur ainsi que coord\_x et coord\_y.

La variable « IntituléProtectionPlus » est également supprimée car près de 90% des valeurs sont absentes donc inexploitables. De plus, cette variable n'apporte rien de plus dans la définition de l'arbre ou de son diagnostic.





Deux attributs ont été créés par agrégation, pour chacun, de deux attributs du corpus. Le premier attribut, appelé EvolutionJsqDiag, est la différence entre le stade de développement initial de l’arbre et son stade de développement lors du diagnostic.

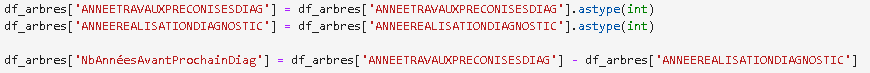
Autrement dit il reflète l’évolution de l’arbre, de sa plantation jusqu’au diagnostic. Il se calcule de la façon suivante :



les valeurs des variables StadeDeDéveloppement et StadeDévelopementDiag étant codifiées par les entiers 1, 2 et 3, correspondant respectivement aux valeurs "arbre jeune", "arbre adulte" et "arbre viellissant". Une différence positive, nulle ou négative, correspond alors à une évolution positive, constante ou négative, du développement de l’arbre.

Le second attribut, appelé NbAnnéesAvantProchainDiag est également une différence, qui

mesure, par un nombre d’années, la nécessité plus ou moins importante d’un prochain diagnostic, conséquence d’éventuels défauts ayant endommagé l’arbre. Il se définit de la façon suivante :



Ces attributs supplémentaires contribueront aux modèles de prédiction conçus dans la suite du projet.

Après l’ensemble de ces démarches de « data cleaning » et de « « data management, nous avons récupéré un data set à zéro nombre de valeurs manquantes et des données au format et type uniformisé prêtes à être analysées et entrainées dans le cadre des deux défi EGC traités.



# **2 – Analyse exploratoire des données**

Analyser le data set de départ et en récupérer des informations, tel était notre objectif avant d’entamer les phases de prédiction. Connaître les données qui seront par la suite exploitées par des puissants algorithmes est primordiale dans tout projet en lien avec les données.

Dans le cadre du projet, nous avons séparé cette partie en deux :

* **L’analyse purement statistique** des données et
* L’analyse des données des arbres par **mapping** sous forme de cartes afin de visualiser les données géographiques des arbres.

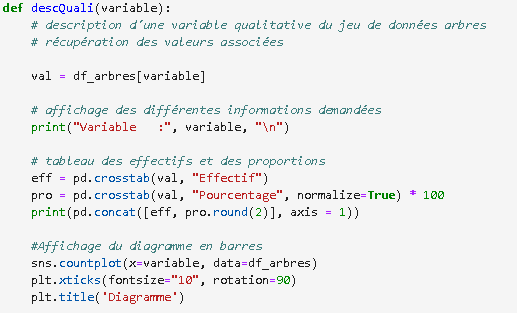
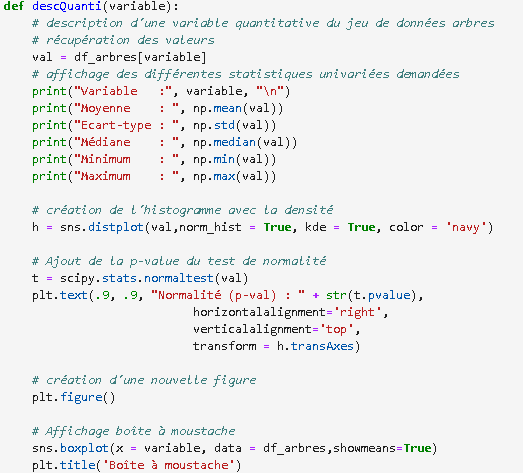
## 2.1 – Analyse statistique des données

Dans un ensemble de données, on peut distinguer 2 types de variables :

* les variables qualitatives et
* les variables quantitatives

Avec pour chacune des moyens de descriptions statistiques différents.

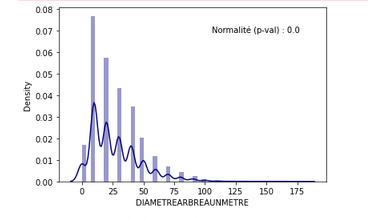
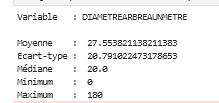
Nous avons ainsi réalisé deux fonction Python décrivant de manière statistique les variables qualitatives pour l’une et les variables quantitatives pour l’autre. Les librairies « MatplotLib » et « Seaborn » du langage Python ont été utilisées.

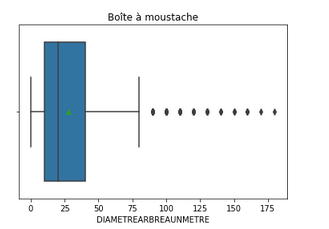


Pour les variables dites «**quantitatives**», un tableau récapitulatif des moyenne, écart-type, médiane, minimum et maximum d’une variable donnée a été réalisée.

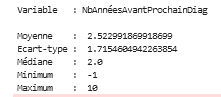
Un histogramme a été construit pour avoir un aperçu de la distribution, et éventuellement comparer à une gaussienne associée à un test de normalité. Une Boîte à moustaches pour estimer la présence ou non d'outliers a également été développée.

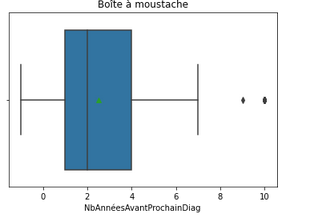
Par exemple, pour la variable sur le diamètre de l’arbre (mètre), nous avons une moyenne de 27,56 mètres environ, le minimum est de 0 mètre (valeur aberrante) et le maximum est de 180 mètres. La médiane est de 20 mètres. Ici, nous pouvons voir que les valeurs extrêmes (les diamètres supérieurs à 80 mètres) impactent grandement la moyenne car elle est supérieure à la médiane de 35%. L’écart type de 20.79 confirme notre impression sur les valeurs extrêmes car ce dernier nous montre que la distribution des données est hétérogène or ici notre distribution est plutôt homogène. Ce sont donc les valeurs extrêmes qui impactent excessivement notre écart- type.

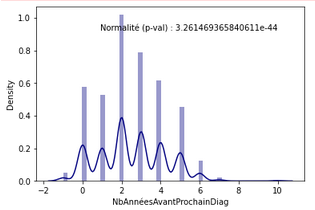




La boite à moustache nous illustre l’analyse faite auparavant. Elle nous permet de confirmer que les diamètres supérieurs à 75 mètres sont à considérer comme valeur extrême.

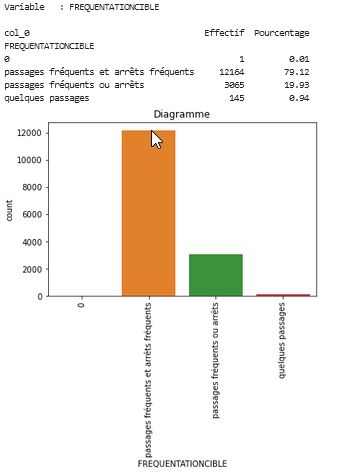
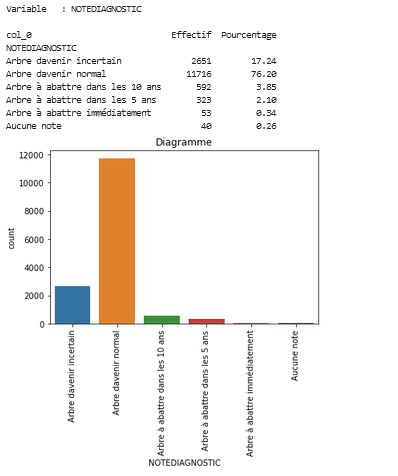
La variable nombre d’année avant le prochain diagnostic a une moyenne de 2.5 ans et une médiane de 2 ans. Le minimum ici n’est pas à prendre ne compte car la valeur est négative. Le maximum lui est de 10 ans pour le prochain diagnostic. Ici, les valeurs dites extrêmes ont moins d’impact (médiane et moyenne plus proche). Les valeurs extrêmes étirent un petit peu la moyenne vers la gauche. L’écart type de 1.7 confirme que la plupart des valeurs sont autour de la moyenne et que la distribution est quasi homogène. La boite à moustache confirme nos analyses.

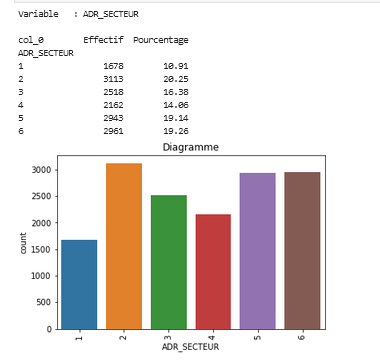




Pour les variables dites « qualitatives », une table d'effectifs et de pourcentages a été réalisée ainsi qu’un diagramme en barres pour identifier les différentes modalités. Les diagrammes circulaires sont ici à éviter ici, car trop de modalités pour certaines variables.

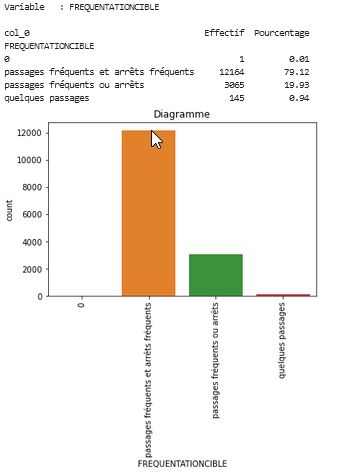
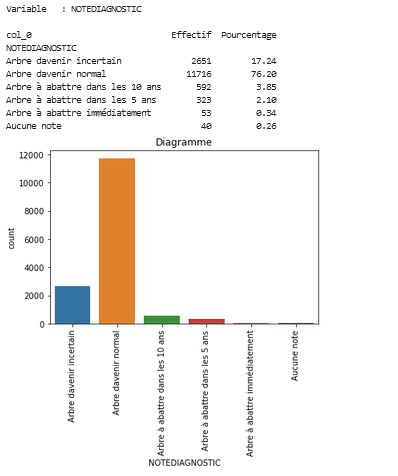
Par exemple, la distribution de la variable fréquentation cible nous montre que la plupart des arbres (79%) ont un passage et un arrêt fréquent d’un passant. La variable qui permet de connaitre le diagnostic de l’arbre nous montre qu’une grande partie des arbres ont un avenir dit normal (76%) des arbres. Nous pouvons observer que 17% des arbres ont un avenir incertain et que 6% des arbres vont devoir être abattre dans les 10 ans.



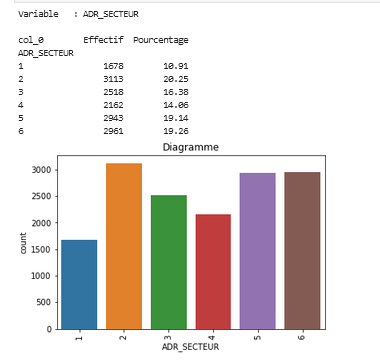
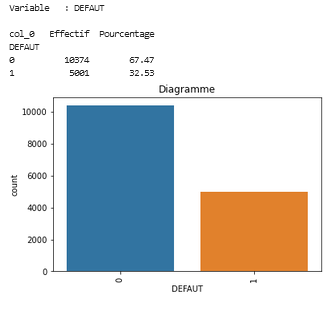
La distribution de la variable du secteur où se trouve l’arbre est plutôt homogène entre les six secteurs. Nous pouvons noter un nombre moins important dans le secteur 1 (11% de la population d’arbre). Grâce à la variable défaut nous pouvons voir que 32% des arbres ont un défaut.

Nous allons maintenant analyser les statistiques descriptives de quelques variables qualitatives.

Pour commencer, la distribution de la variable fréquentation cible nous montre que la plupart des arbres (79%) ont un passage et un arrêt fréquent. La variable qui permet de connaitre le diagnostic de l’arbre nous montre qu’une grande partie des arbres ont un avenir dit normal (76%) des arbres. Nous pouvons observer que 17% des arbres ont un avenir incertain et que 6% des arbres vont devoir être abattre dans les 10 ans.



La distribution de la variable du secteur où se trouve l’arbre est plutôt homogène entre les six secteurs. Nous pouvons noter un nombre moins important dans le secteur 1 (11% de la population d’arbre). Grâce à la variable défaut nous pouvons voir que 32% des arbres ont un défaut.

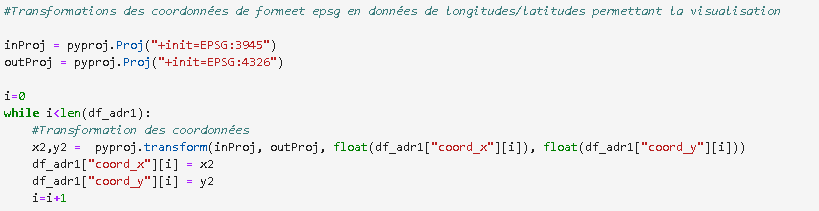


## 2.2 – Analyse par mapping des données

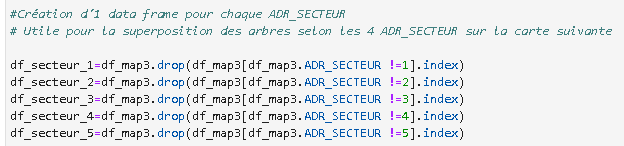
En analysant notre Data Frame de départ, nous nous sommes rendus-compte de la présence de données géographiques de type longitude (coord\_x) et latitude (coord\_y) qui sont pertinentes dans la représentation visuelle des arbres.

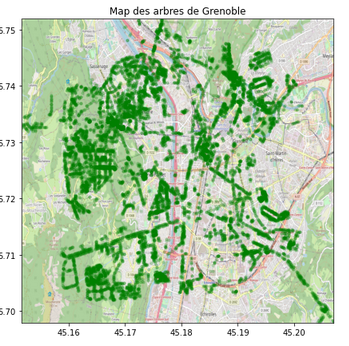
Nous avons représenté ces données sous forme de carte avec la librairie « pyroj » de Python, librairie de projections cartographiques et de transformations de coordonnées.

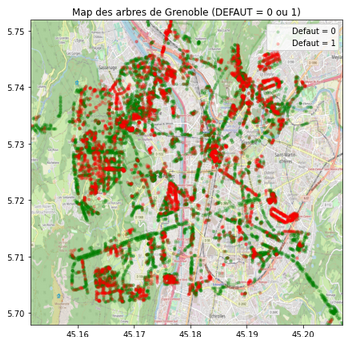
Une transformation des coordonnées de format « epsg » en données de longitudes/latitudes permettant la visualisation a été réalisée avant d’importer une carte de fond de la ville de Grenoble et d’afficher nos arbres de différentes manières.

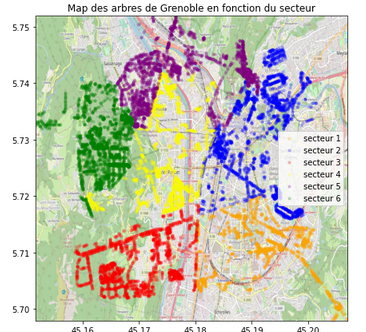


A l’aide des techniques de manipulations de Data frame avec la librairie « Pandas », 3 cartes ont ainsi été réalisées. Les techniques de « merge » et « drop » ont été employées pour superposer les différents défauts d’un arbre à partir de leurs localisations. Les différents points affichés à travers les cartes représentent les arbres et leurs positions. L’unicité de celles-ci ont été notamment récupérées à l’aide de l’attribut « CODE » des arbres mis à notre disposition.



La map N°1 représente la localisation de la totalité des arbres de la métropole de Grenoble. Grâce à la seconde map nous pouvons observer la localisation des arbres ayant un défaut (point rouge) par rapport à ceux sans défaut (point vert). Pour finir, avec la dernière map nous pouvons observer la localisation de chacun des arbres en fonction de leur secteur de localisation. On peut remarquer que la majorité des arbres possédant un défaut sont localisés à l’ouest de Niort dans les secteurs 3,4 et 5.





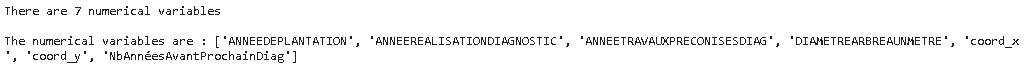
# **3. Prédiction de l’existence d’un défaut d’un arbre – Classification binaire**

Identifier les arbres ayant un défaut et prédire la localisation de celui-ci constitue le premier défi du sujet d’étude. Avant d’appliquer nos modèles de prédiction, il est nécessaire de réaliser des traitements en amont comme par exemple la définition de notre base d’apprentissage et de test.

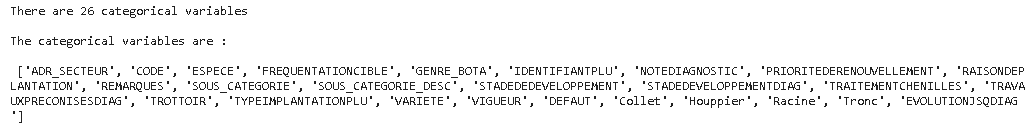
## 3.1 – Traitement préliminaires

Tout d’abord, il est important de signaler que notre Data Frame est constitué de 2 types de données :

* Des données «numerical» de type entier



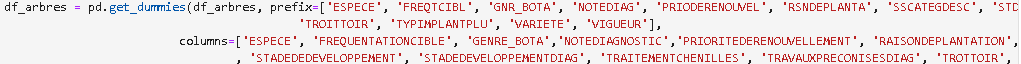
* Des données « object » de type catégoriel



Or lorsque l’on est en présence de données **catégorielles**, il est nécessaire d’encoder nos variables. On parle d’**encodage binaire** lorsque l’on encode une variable en deux catégories, en général en 0, 1. Par exemple, une variable décrivant le sexe d’une personne pourra être encodé en 0, 1. Une **variable catégorielle** (avec plus que deux catégories) pourra également être encodé, selon le nombre de catégories qui la compose.

La stratégie que nous avons adoptée est celle de la création de **dummies**. A partir d’une variable, pour les n-catégories qui la compose, on créera n-1 colonnes encodées de façon binaire (0-1) correspondant, chacune correspondant à une des catégories comprises dans la variable.

Nous avons ainsi appliqué cette technique à notre Data Frame pour l’ensemble de nos variables catégorielles.



Nous passons ainsi de 31 features à 496 features à la suite de l’encodage binaire.

La phase d’apprentissage de nos différents modèles peut donc s’appliquer aux données en résultant. Pour cela, Scikit-learn est une bibliothèque libre Python qui est destinée à l'apprentissage automatique et à la construction de modèles de classification et de prédiction.

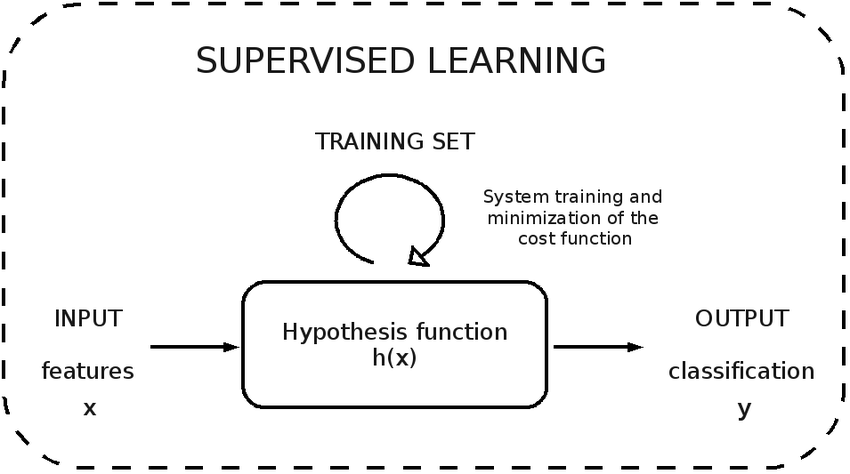


Dans notre cas d’étude, notre variable à prédire est la variable « **Défaut** » de notre Data Frame. Nous souhaitons la prédire grâce aux autres variables « explicatives » ou « prédictives » de notre ensemble de données.



Distinguons deux types de problèmes de prédiction : la présence ou non d’une variable à expliquer Y ou d’une forme à reconnaître qui a été, conjointement avec X,observée sur les mêmes objets.

Dans notre cas d’étude, il s’agit bien d’un problème de modélisation ou **apprentissage supervisé** : trouver une fonction h susceptible, au mieux selon un critère à définir, de reproduire Y ayant observé X.



Comme nous l’avons vu en cours, l’entraînement d’un modèle de prédiction revient à mesurer l’erreur de la sortie de l’algorithme avec les données d’exemple et chercher à la minimiser.

Un premier piège à éviter est d'évaluer la qualité de votre modèle final à l'aide des mêmes données qui ont servi pour l'entraînement. En effet, le modèle est complètement optimisé pour les données à l'aide desquelles il a été créé. L'erreur sera précisément minimum sur ces données. Alors que l'erreur sera toujours plus élevée sur des données que le modèle n'aura jamais vues.

Pour minimiser ce problème de sur-apprentissage, la meilleure approche est de séparer **dès le départ** notre jeu de données en deux parties distinctes :

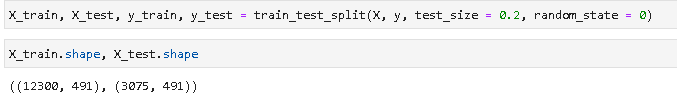
* Le **training set**, qui va nous permettre d’entraîner notre modèle et sera utilisé par l’algorithme d’apprentissage. C'est celui dont on a parlé depuis le début.
* Le **testing set**, qui permet de mesurer l’erreur du modèle final sur des données qu’il n’a jamais vues. On va simplement passer ces données comme s'il s'agissait de données que l’on n'a encore jamais rencontrées (comme cela va se passer ensuite en pratique pour prédire de nouvelles données) et mesurer la performance de notre modèle sur ces données.

C'est à nous de définir la proportion du dataset que vous souhaitez allouer à chaque partie.

En général, les données sont séparées avec les proportions suivantes : **80 % pour le training set et 20 % pour le testing set.**



On peut cette fois utiliser la fonction de scikit-learn  « train\_test\_split » qui prend en paramètre la proportion des données tests désirée :



Notre ensemble d’apprentissage est ainsi constitué de 12 300 arbres. Notre ensemble de test est constitué de 3075 arbres. On retrouve ainsi les proportions 80%/20% définie auparavant à travers notre ligne de code.

## 3.2 – Définition des classifieurs, choix des hyperparamètres et des métriques d’évaluation

### 3.2.1 – Définition des classifieurs de prédiction

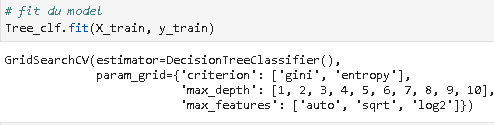
Grâce à ces phases d’encodage et de définition du processus d’apprentissage, nos modèles de prédiction sont prêts à être entrainés.

Nous avons décidé de choisir 3 classificateurs, vus dans la cadre de notre cursus.

Nous avons choisi de tester un algorithme de trois catégories différentes :

* Méthodes produisant un modèle de classification explicite. Les méthodes de classification produisant un modèle explicite, tel qu’un arbre de décision a été utilisé de façon complémentaire.

L’objectif est de créer un modèle qui prédit les valeurs de la variable cible, en se basant sur un ensemble de séquences de règles de décision déduites à partir des données d’apprentissage. L’arbre approxime donc la cible par une succession de règles logiques très simples (typiquement des séparations de données par « hyperplan », généralisation d’un plan à plus de 2 dimensions), chaque règle étant choisie en fonction du résultat de la règle précédente.



* Méthodes d’ensemble. Les méthodes d’ensemble pour la classification nous semblent pertinentes en raison du nombre important d’instances dans le jeu de données et de la difficulté de la tâche de prédiction de la présence d’un défaut. Le programme Random Forest a été testé.

Un apprentissage est réalisé en parallèle sur de multiples arbres de décision construits aléatoirement et entraînés sur des sous-ensembles de données différents. Le nombre idéal d’arbres, qui peut aller jusqu’à plusieurs centaines voire plus, est un paramètre important : il est très variable et dépend du problème. Concrètement, chaque arbre de la forêt aléatoire est entrainé sur un sous ensemble aléatoire de données selon le principe du bagging, avec un sous ensemble aléatoire de « features » (caractéristiques variables des données) selon le principe des « projections aléatoires ». Les prédictions sont ensuite moyennées lorsque les données sont quantitatives ou utilisés pour un vote pour des données qualitatives, dans le cas des arbres de classification.

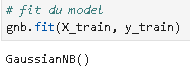
* Méthodes bayésiennes. Les méthodes bayésiennes étant généralement performantes, le programme de classification GaussianNB() a également été testé.

Le classificateur Naive Bayes suppose que l'effet d'une caractéristique particulière dans une classe est indépendant des autres caractéristiques. Même si ces caractéristiques sont interdépendantes, ces caractéristiques sont toujours considérées indépendamment. Cette hypothèse simplifie le calcul, et c'est pourquoi elle est considérée comme naïve. Cette hypothèse est appelée indépendance conditionnelle de classe.

Nous utilisons l’un des classifieurs Bayes qui est le classificateur GaussianNb().



L'instance « gnb » de l'estimateur (for classifier) ​​est d'abord ajustée au modèle ; c'est-à-dire qu'il doit apprendre du modèle.



### 3.2.2 – Choix des hyperparamètres des classifieurs

### 3.2.3 - Choix des métriques d’évaluation et gestion du sur-apprentissage

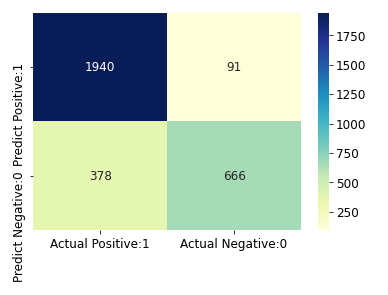
Une fois lancé les algorithmes, il est nécessaire de pouvoir les évaluer grâce à des métriques d’évaluation. On mesure la performance du modèle sur la base de test. Il existe certaines façons standard de le faire en fonction des types de problèmes. Dans le cadre de notre étude, il s’agit d’une classification avec une variable cible y qualitative. Ainsi l’ensemble de nos métriques d’évaluation vont se résumer autour de :

* **Une matrice de confusion**

La matrice **indique le nombre de prédictions correctes pour chaque classe** et le nombre de prédictions incorrectes pour chaque classe organisée en fonction de la classe prédite. Les valeurs situées en dehors de la diagonale constituent des erreurs.

Chaque ligne du tableau correspond à une classe prédite, et chaque colonne correspond à une classe réelle.

Dans notre cas d’étude, on peut prendre l’exemple de la matrice de confusion correspondant au classificateur Random Forest. La matrice a été conçue à l’aide de la librairie Seaborn de Python.



Nous pouvons ainsi compter 469 erreurs, 469 arbres mal classés :

* 91 arbres ont été identifiés comme étant des arbres possédant un défaut (à tort) .
* 378 arbres ont été identifiés comme étant des arbres ne possédant pas de défaut (à tort).
* **Précision et rappel du modèle**

À partir de la matrice de confusion on peut dériver tout un tas de critères de performance. De manière générale, on préfère donner une fraction d'erreurs à un nombre total d'erreurs (5% d'erreurs, ça parle plus que 10 erreurs, quand on ne connaît pas le nombre de points dans le jeu de test).

Par exemple, le **rappel** ("recall"  en anglais) qui est le **taux de vrais positifs**, c’est à dire la proportion d’arbres ayant un défaut que l’on a correctement identifiés. C’est la capacité de notre modèle à détecter la présence des défauts chez les arbres.

On s’intéressera aussi à l’**accuracy**, indique le pourcentage de bonnes prédictions, mesurant la proportion d’arbres présentant un défaut, correctement prédis.

En reprenant l’exemple du classificateur Random Forest, nous avons obtenu les résultats suivants :



Cela signifie donc que :

* La prédiction qu’un arbre a un défaut est juste dans 84 % des cas.
* Le classificateur prédit correctement 64% des arbres possédant un défaut.
* **Courbe ROC et AUC.**

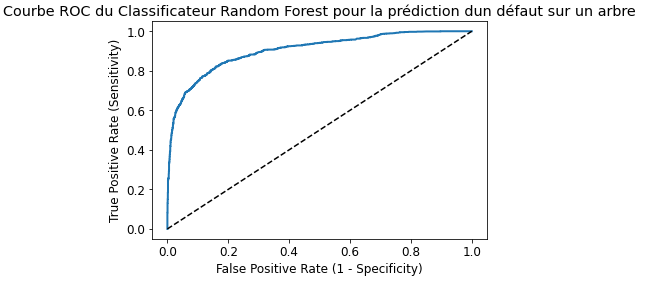
Une **courbe ROC** (**receiver operating characteristic**) est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification. Graphiquement, il [représente](https://fr.wikipedia.org/wiki/Repr%C3%A9sentation_graphique) la [mesure](https://fr.wikipedia.org/wiki/Mesure_physique) de la [performance](https://fr.wikipedia.org/wiki/Performance) d'un [classificateur](https://fr.wikipedia.org/wiki/Classification_automatique) binaire sous la forme d'une [courbe](https://fr.wikipedia.org/wiki/Courbe) qui donne le taux de [vrais positifs](https://fr.wikipedia.org/wiki/Vrai_positif) (fraction des positifs qui sont effectivement détectés) en fonction du taux de [faux positifs](https://fr.wikipedia.org/wiki/Faux_positif) (fraction des négatifs qui sont incorrectement détectés).

L’aire sous la courbe ROC noté AUC est un indicateur statistique quantifiant la capacité du classificateur à discriminer, prédire les arbres possédant un défaut ou non.

Dans le cas du classificateur Random Forest, nous avons obtenu un score AUC de 0,90.



Et la courbe ROC suivante :



Cette valeur d’AUC de 0,90 signifie que, dans 90% des cas, un arbre tiré au sort de la population des arbres atteints d’un défaut a une valeur du test du classifieur supérieure à celle d’un arbre tiré au sort de la population des arbres sans défauts.

Quand on développe un modèle d’apprentissage, nous essayons de lui apprendre comment atteindre un objectif : prédiction de l’existence d’un défaut pour les arbres de Grenoble. Pour ce faire, nous partons d’une base de données d’arbres qui va servir à l’entrainement du modèle, c’est-à-dire à son apprentissage pour arriver au but voulu. Or, si nous ne faisons pas bien les choses, il se peut que le modèle considère comme validées, uniquement les données qui ont servi à faire l’apprentissage, ne reconnaissant aucune autre donnée qui soit un peu différente de la base initiale. C’est ce phénomène que l’on appelle « surapprentissage ».

Pour éviter ce phénomène, nous avons réalisé la technique de Kfolds cross validation. Contrairement à la validation classique, ou l’on divise les données en deux, en cross validation on divise les données d’entraînements en plusieurs groupes. L’idée est en suite d’entraîner le modèle sur tous les groupes sauf un. Si on a k groupes, on entraînera le modèle k fois avec à chaque fois un nouveau groupe de test.

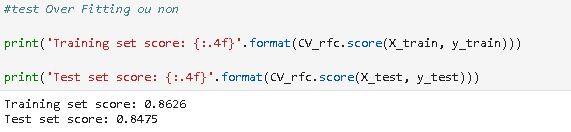
Cette technique de validation croisée est appelée k-fold et nous l’avons appliqué pour nos 3 classifieurs limitant ainsi le risque de surapprentissage.



De plus, nous avons réalisé en amont de la prédiction une phase de Data Cleaning permettant de réduire le nombre de features et de supprimer ainsi les possible corrélations entre les variables. De cette façon on simplifie au maximum nos données, on améliore les performances du modèle et on réduit au passage les risques d’overfitting.

Identifier la présence d’un surapprentissage est possible en regardant les score de prédiction pour l’ensemble test et l’ensemble d’apprentissage. Plus ces scores sont éloignés, plus le risque d’overfitting est grand.

Nous avons ainsi fait la démarche pour chacun de nos 3 classificateurs. Et comme le montre la capture ci-dessous, les scores sont restés proches à chaque fois notamment pour le classifieur Random Forest.



Nous n’avons pas eu de surapprentissage dans la conception de nos modèles.

3.3 – Présentation et interprétation des résultats

Les résultats des métriques précédemment définies pour chacun des classificateurs sont résumés dans le tableau ci-dessous :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Prédiction Défaut** | **Matrice de Confusion** | **Accuracy** | **Rappel** | **Courbe ROC** | **AUC** | **Importance\_Features** |
| **Bayes GNB CLF** |  | 0,40 | 0,62 |  | 0,5426 | Non défini |
| **Decision Tree CLF** |  | 0,83 | 0,64 |  | 0,85 |  |
| **Random Forest CLF** |  | 0,85 | 0,64 |  | 0,90 |  |

A travers ce tableau de résultat, on remarque que le classificateur **Random Forest** est le plus performant en regard des différentes métriques : 0,85 en « accuracy » à quasi-égalité avec le classifier Tree et un rappel de 0,64.

Le classifier Random Forest se démarque cependant du Tree Classifier par un AUC élevé de 0,90.

La méthode d’ensemble de classifieurs RandomForest semble donc celle qui fournit le meilleur modèle de prédiction pour les problèmes posés.

Grâce à sa fonction associée :



Nous avons pu définir les attributs qui sont les plus prédictifs d’un défaut :



Une première analyse des 10 variables contribuant le plus sur l’ensemble des arbres nous, indique que **l’avis de l’expert a une grande influence** (variables centrées sur le **diagnostic encadrées noires**), ainsi que **la géolocalisation** dans une moindre mesure (encadrées **bleu**), puis en dernier des **caractéristiques physiques propres à l’arbre** (diamètre, vigueur, NoteDiag encadrées **vert**).

Nous avons voulu vérifier si seul l’expert pouvait fournir de bonnes prédictions, et si nous ne gardons que les variables de type diagnostic, nous nous approchons des résultats utilisant l’ensemble des variables. Par exemple, sur un modèle Random Forest paramétré de la même manière que précédemment, nous obtenons une exactitude de 84% et un rappel de 58%. Sachant que nous atteignons 64% de rappel dans le meilleur des cas, cela montre qu’il y a bien d’autres informations permettant de couvrir un ensemble plus complet, et de réduire ainsi le nombre de faux négatifs.

L’avis de l’expert constitue donc un bon moyen de prédiction mais ne constitue pas à lui seul le seul prédicteur.

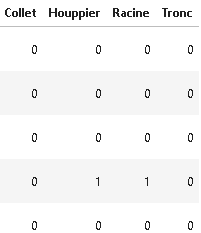
# **4. Prédiction de la localisation d’un défaut d’un arbre – Classification multi-labels**

Prédire la présence de défaut sur un arbre grenoblois correspondait à la première étape du Défi 1. La deuxième étape va plus loin et consiste à prédire la localisation du défaut de l’arbre.

3.1 – Définition de notre méthodologie

La classification dans le domaine de l’apprentissage automatique est une tâche prédictive qui permet d’apprendre sur des observations labellisées (c’est-à-dire « étiquetées ») afin de prédire les labels des nouvelles observations. Il existe plusieurs types de classifications en fonction du sujet à prédire

Dans notre Data Set de départ, les localisations des défauts, nos variables cibles sont identifiées ou non à travers les colonnes suivantes :

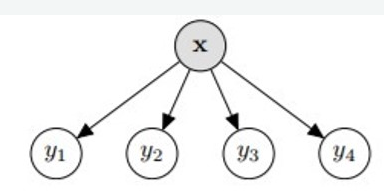


On parle alors de **classification multi-labels** : plusieurs labels (localisations du défaut) à prédire et chaque label n’a que deux valeurs possibles (0 : absence de défaut et 1 : présence d’un défaut).

Pour pouvoir résoudre cette problématique de classification multi-classes, nous avons testé 3 méthodes :

* Une transformation en problèmes de classification binaire appelée **méthode de pertinence binaire**. Cela revient à former indépendamment un classificateur binaire pour chaque étiquette. Le modèle combiné prédit ensuite toutes les étiquettes de cet échantillon pour lesquelles les classificateurs respectifs prédisent un résultat positif.

Données Arbres



Tronc

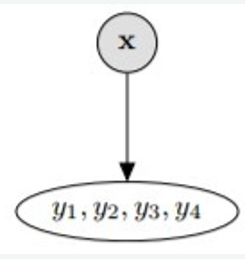
Racine

Collet

Houppier

* Une transformation en problème de classification multi-classes : la transformation LP (**Label powerset**) crée un classificateur binaire pour chaque combinaison d'étiquettes de localisation présente dans l'ensemble d'apprentissage. Les variables cibles (y1, y2,...,yn) sont combinées et chaque combinaison est traitée comme une classe unique.

Données Arbres



Combinaison des défauts

* **L’adaptation d’algorithmes de classification**. Certains algorithmes / modèles de classification ont été adaptés à la tâche multi-étiquettes, sans nécessiter de transformations de problèmes. Exemples pour l’algorithme des k-plus proches voisins : les algorithmes ML-kNN et BrKNN étendent le classifieur k-NN aux données multi-étiquettes.

Nous avons ainsi adopté ces 3 types de méthodes pour les quelle plusieurs classifieurs ont été employés.

Contrairement à un classificateur traditionnel, un classificateur multi-labels peut faire une prédiction totalement correcte, partiellement correcte ou totalement incorrecte. C’est la raison pour laquelle on ne peut pas utiliser les mêmes métriques de performance que celles utilisées dans la classification traditionnelle mise en œuvre dans notre partie précédente.

Nous allons nous appuyer sur les Métriques basées sur les observations : elles sont calculées séparément pour chaque observation. Deux métriques ont été choisies :

-Hamming Loss: la différence symétrique entre les labels faussement prédits et les labels réels. Plus la métrique est faible, plus l’erreur de prédiction est limitée.

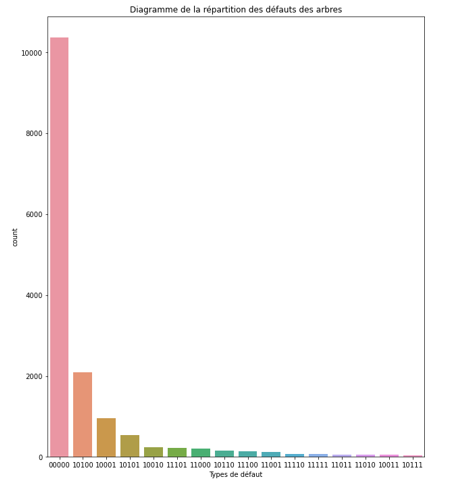
- le Rappel : la proportion entre les labels correctement prédits et les labels réels.

3.2 – Présentation des résultats et interprétation

Avant de pouvoir appliquer nos algorithmes pour les différents cas définis précédemment, nous avons voulu obtenir la répartition des défauts parmi tous les arbres de notre base.

Nous avons ainsi concaténé les variables de localisation pour définir une seule et unique variable de localisation dont nous avons obtenu sa distribution grâce à un histogramme.

L’histogramme quantifie les occurrences des différentes combinaisons de classes de réponse. La combinaison la plus fréquente est évidemment ’00000’, qui correspond un arbre sain (le premier 0 correspond à la présence d’un défaut ou non, les suivants à la maladie présente : Collet, Houppier, Racine ou Tronc). La seconde combinaison la plus fréquente représente les arbres malades uniquement au niveau du houppier (’10100’) avec 2000 occurrences.



On remarque donc bien que houppier + Présence défaut est la relation la plus fréquente qui dénote qu’un arbre atteint d’un défaut est très souvent atteint au niveau du houppier.

Après cette pré-étude, nous avons construit nos classifieurs pour chacune des 3 méthodes définies précédemment.

Ces 3 méthodes ont été conçues à travers des pipelines permettant de créer un objet unique qui inclut toutes les étapes du prétraitement et de la classification des données. Comme pour la technique du GridSearch utilisée précédemment, nous utilisons cette technique dans le cadre de la classification multi-étiquettes. Ci-dessous un exemple de l’instance de la méthode Binary Relevance avec le classifier Random Forest.



Les résultats obtenus pour les 3 méthodes appliquées à différents classificateurs sont résumés ci-dessous :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prédiction localisation**  **Défauts - Arbres** | | **F1 – Score** | **Hamming Loss** | **Accuracy** |
| **Binary Pertinence** | **Ramdom Forest** | **0,601** | **0,082** | **0,76** |
| Bayes Classifier | 0,23 | 0,76 | 0,43 |
| **Label Power Set** | Bayes Classifier | 0,23 | 0,55 | / |
| Regression Logistic | 0,49 | 0,089 | / |
| **Random Forest** | **0,60** | **0,083** | / |
| **Adaptative Classifier** | **BrKNN Classifier** | **0,726** | / | / |
| **MlKNN** | **0,719** | / | / |

A première vue, on remarque que ce sont les classificateurs adaptés à la problématique multi labelle qui obtiennent les meilleurs résultats : F1 score le plus élevé et la mesure Hamming Loss la plus faible. En effet, les classificateurs BrKNN et MlKNN qui sont les classificateurs adaptés du classifieur KNN ont obtenu les meilleurs résultats.

Concernant les méthodes de transformation du problème de classification multi-label, on remarque que c’est l’algorithme Random Forest qui est le plus performant que ce soit pour la technique de Binary Pertinence ou celle du Label PowerSet.

Ces différents modèles pourraient encore être améliorés (paramétrage de l’entraînement, affinement du seuil, online learning...). La technique de tuning des paramètres à l’aide du GridSearch auraient pu notamment être appliqués pour les différents algorithmes par exemple.

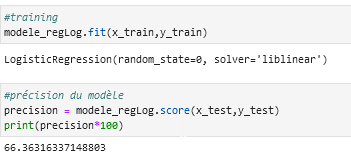
Mais les différents résultats donnent une tendance : la méthode d’ensemble de classifieurs RandomForest est celle qui fournit le meilleur modèle de prédiction dans le cadre de l’étude des arbres pour les problèmes unilabel et multilabels.

**4. Recherche de préconisations pour faciliter l’entretien des arbres de Grenoble**

Nous avons précédemment analysé la base de données des arbres et tenter de prédire l’existence ou non d’un défaut sur un arbre et leur localisation.

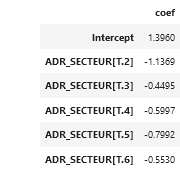
Poursuivons notre fouille de données afin de mieux connaître l’état du parc végétal de Grenoble. L’objectif de cette partie est de poursuivre nos analyses afin de définir des préconisations d’entretien du parc grenoblois.

Dans un premier temps, nous avons utilisé la régression logistique pour essayer de prédire la présence d’un défaut grâce au secteur d’implantation. Nous avons généré un population test et une population d’apprentissage pour tester cette technique. Le score obtenu est de 66%. Ce score est correct il nous est donc envisageable de prédire la présence d’un défaut en fonction du secteur.

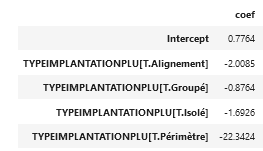


Dans un second temps, grâce a la GLM nous avons étudié la présence d’un défaut en fonction de la variété de l’arbre, le type d’implantation et le secteur d’implantation.

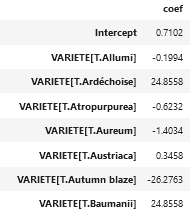
L’étude sur le secteur d’implantation nous a permis d’observer que les secteurs 2 et 5 sont les plus susceptibles d’avoir un défaut sur leurs arbres. La carte évoquée précédemment sur la localisation des arbres qui ont un défaut a des résultats dans ce sens. Ce résultat est cependant est à relativiser car nous avons seulement utilisé une seule méthode.



L’étude sur le type d’implantation nous a permis de remarquer que l’implantation de type « périmètre » est le plus susceptible d’avoir un défaut sur l’arbre. Ce résultat est cependant est à relativiser pour les mêmes raisons que le secteur d’implantation.



L’étude sur le type d’implantation nous a permis de remarquer que l’implantation de type « périmètre » est le plus susceptible d’avoir un défaut sur l’arbre. Ce résultat est cependant est à relativiser pour les mêmes raisons que le secteur d’implantation.



Nous avons effectué la même analyse sur la variété de l’arbre. Cette étude nous a permis de remarquer ce certaines variétés sont plus susceptibles d’avoir un défaut (*Autumn blaze* et *scope* notamment). A l’inverse certaines espèces ne sont très rarement voire jamais malade (*Baumanii*). Cette analyse est à prendre avec du recul car dans le parc végétal de Grenoble certaines espèces ne sont que très peu représentées (moins de 20 individus) le cas de chaque individu a lors une importance démesurée pouvant prêter à confusion quelques fois. Il est également à relativiser pour les mêmes raisons que le secteur d’implantation et le type d’implantation.

Avec ces analyses nous pouvons donner quelques préconisations à la ville de Grenoble afin de faciliter l’entretien de son parc végétal et de limiter l’apparition de nouvelles maladies. Tout d’abord, deux solutions peuvent être préconisées concernant le secteur d’implantation. La première est de limiter la plantation d’arbres dans les secteurs les plus contaminés ou de déplacer certains arbres dans des zones plus saines. La seconde serait solution serait de diagnostiquer plus souvent les arbres de ces secteurs. Pour le type d’implantation la principale préconisation serait de stopper l’implantation de type « périmètre » pour les nouveaux arbres et quand c’est possible de changer le type d’implantation pour les arbres déjà plantés. Pour finir, concernant la variété de l’arbres, les préconisations à prendre serait de diagnostiquer plus régulièrement les variétés les plus touchées par les maladies la seconde préconisation serait de choisir des arbres peu touchés par les maladies lorsque de nouveaux arbres seront plantés.

# **5. Conclusion et mise en perspectives**

Pour cette seconde édition du défi EGC, Big Datext, entreprise Grenobloise spécialisée dans l’analyse prédictive, et la mairie de Grenoble se sont toutes deux impliquées dans la mise en place et la diffusion de la base de données du challenge.

Dans le cadre de l’enseignement « Entreposage et Fouille de données », nous avons tenté à notre niveau de répondre aux différents défis proposés par le challenge.

**Le premier défi** consiste en une tâche de prédiction visant à déterminer, à partir des données disponibles, si l’arbre a ou non un défaut (Variable Default or not) et dans l’affirmative lequel (Variables Collet, Houppier, Racine, Tronc), sachant qu’un arbre peut présenter plusieurs défauts. Nous avons ainsi découpé le problème en deux problématiques : la prédiction de l’existence du défaut et la prédiction de ce même défaut.

Sur une tâche de prédiction unilabel (Variable Default or not) une baseline permet d’obtenir 85% pour l’accuracy et 64% de rappel tandis que sur la tâche multilabel (Variables Collet, Houppier, Racine, Tronc) les taux sont respectivement de 76% et 60 % pour la précision et le F1 Score.

L’étude du premier défi nous a permis de comprendre que les diagnostics des experts et les caractéristiques propres à l’arbre influencent, prédisent la présence ou non d’un défaut sur un arbre donné. Concernant, la prédiction de leur localisation sur l’arbre, les scores obtenus pourraient être améliorés pour définir ainsi les facteurs qui puissent influencer sur la présence des défauts.

**Le deuxième défi**, plus ouvert, vise à appliquer des techniques d'extraction et de gestion de connaissances afin de mieux connaitre l’état du « parc végétal » de Grenoble, de mieux comprendre son évolution et de fournir des préconisations pour faciliter son entretien.

Le second défi nous a permis de comprendre que la localisation géographique des arbres, ainsi que la variété et le type d’implantation peuvent permettre d’anticiper l’entretien des arbres. Certain type d’implantations et certain secteur ont un lien étroit avec la présence d’un défaut. Il est alors possible de prioriser l’entretien en fonction de ces variables.

Cette étude a été extrêmement enrichissant à réaliser. Il nous a permis de consolider nos connaissances en apprentissage supervisé et les techniques de machines learning associées. A travers ce projet, nous avons découvert un autre type de classification : la classification multi-label, problématique que nous n’avons pas traité avant le projet des arbres grenoblois.

De manière plus générale, ce projet a été l’occasion de traiter une problématique orientée data-science dans sa globalité partant du data cleaning et du data management jusqu’aux algorithmes de machine learning afin de faire des prédictions.