Défi EGC 2017: Modélisation Cost-Sensitive et Enrichissement de données

Vincent Levorato*,**, Michel Lutz***,****
Matthieu Lagacherie*

*OCTO Technology
vlevorato@octo.com, mlagacherie@octo.com
**LIFO, Université d'Orléans
vincent.levorato@univ-orleans.fr
***TOTAL, Digital Corporate Team
michel.lutz@total.com
****Ecole des mines de Saint-Etienne

Résumé. La conférence EGC'2017 propose un défi dont le contexte est la gestion des espaces verts pour la ville de Grenoble, et notamment des arbres qui y sont présents. L'objectif est 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. Après avoir obtenu quelques résultats intéressants avec des modèles standards, notre approche utilisant un modèle Cost-Sensitive One Against All (CSOAA) nous permet d'obtenir une exactitude de 0,86, une précision de 0,88, et un rappel de 0,91 sur la prédiction unilabel, et une précision/rappel micro de 0,82/0,74 ainsi qu'une précision/rappel macro de 0,66/0,46 pour la prédiction multilabel. L'extraction de connaissances pour la tâche 2 nous a permis de mettre en relief l'intérêt de l'ajout de données sur la nature des maladies et la concentration de la pollution dans la ville.

1 Introduction et Données

Le travail présenté dans cet article répond 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. Une partie de l'article expliquera notre approche sur les 2 types de prédiction proposés dans la tâche 1 (unilabel et multilabel). Une seconde partie proposera un enrichissement des données, suivie d'une conclusion. Avant cela, nous étudierons comment nous avons préparé les données afin de les intégrer à nos modèles.

1.1 Données disponibles

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). Parmi les données fournies, la géolocalisation de chaque arbre était présente

à travers 2 coordonnées, x et y (voir figure 1 pour un aperçu de l'implantation des arbres dans la ville). Un premier jeu de données contenait environ 10 000 arbres et le second environ 5 000 pour un total d'un peu plus de 15 000 arbres. Le jeu final de test contient également 5 000 arbres, ce qui nous donne un jeu d'apprentissage correspondant à trois quarts des données contre un quart pour le test.

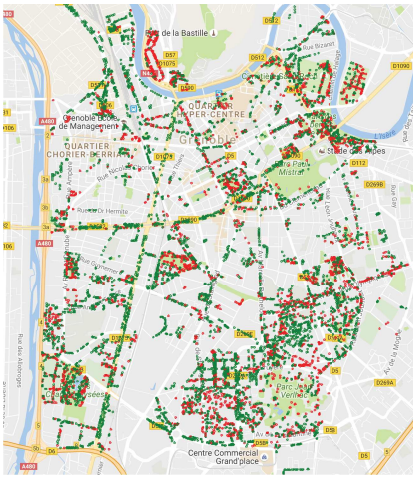

FIG. 1 – *Positionnement des arbres sains (en vert) et malades (en rouge)*

1.2 Préparation des données

1.2.1 Données numériques

Un certain nombre de champs (7 au total) possède des valeurs numériques (ADR_SECTEUR, ANNEEDEPLANTATION, ANNEEREALISATIONDIAGNOSTIC, ANNEE TRAVAUXPRECONISESDIAG, IDENTIFIANTPLU, coord_x, coord_y) pour lesquels aucun formatage n'est nécessaire. Nous rajoutons le champ DIAMETREARBREAUNMETRE dans cette catégorie en ne prenant que la première valeur de l'intervalle. Seules les valeurs manquantes ont été gérées pour certains de ces champs :

— 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.

— DIAMETREARBREAUNMETRE : 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 par rapport au genre botanique de l'arbre en question.

1.2.2 Données catégorielles

A part le champ REMARQUES, tous les autres champs ont été transformés en valeurs numériques, avec un entier pour chaque modalité de la catégorie. Les valeurs manquantes ont

été gérées ainsi :

— ESPECE : idem que pour le diamètre, les valeurs manquantes ont été remplacées par le mode en fonction du genre botanique de l'arbre (GENRE_BOTA).
— Pour toutes les autres variables catégorielles, les valeurs manquantes ont été gérées comme une catégorie en soi.

Nous travaillerons, après préparation des données, sur *un ensemble de 15 213 arbres* exactement. Ce qui suit ne modifiera pas ce nombre.

1.2.3 Données textuelles

Seul le champ REMARQUES a retenu notre attention en tant que donnée textuelle. Remplacer tout texte par 1 et toute valeur manquante nulle ou manquante par 0 permet de garder le champ pour tester un premier modèle, mais n'utilise pas les informations du texte. Dans un deuxième temps, nous avons utilisé une technique connue pour la comparaison de grands volumes de textes : le LSH (Local Sensitive Hashing) (Andoni et Razenshteyn, 2015) couplé aux *n-grams* (Kondrak, 2005). Pour $n = 5$, nous avons haché chaque n-gram du champ REMARQUES, puis n'avons gardé que la valeur minimale (minHash). L'idée est que plus deux textes sont identiques, plus la probabilité d'avoir un certain nombre de codes de hachage identiques est grande. En utilisant 50 fonctions de hachage différentes, cela ajoute autant de variables numériques par arbre. Cela permettra d'améliorer légèrement le modèle comme nous le verrons dans les résultats.

1.2.4 Données de géolocalisation

Les coordonnées des arbres sont, dans notre cas, gérées comme de simples données numériques. Néanmoins, nous avons voulu vérifier si la densité d'arbres à un endroit donné pouvait ou non influer sur les prédictions. Nous avons donc effectué un clustering sur les points correspondant aux coordonnées des arbres. De prime d'abord, on pense à l'algorithme k-Means, cependant, celui-ci impose de choisir la valeur de k, contrainte qui peut être traitée, mais surtout a comme hypothèse forte d'avoir des clusters linéairement séparables. L'algorithme de clustering DBSCAN résout ces deux contraintes, mais ne permet pas de construire des clusters de densités différentes, ce qui est ce que nous voulons précisément obtenir. Nous nous sommes donc penchés sur l'algorithme OPTICS qui généralise DBSCAN et permet d'obtenir des clusters de densités différentes (Ankerst et al., 1999). Pour les paramètres ϵ et $minPts$ de la méthode qui sont respectivement le rayon maximum de recherche et le nombre minimum de voisins à trouver dans ce rayon, nous les avons traités comme suit : nous avons ignoré ϵ en prenant la plus grande valeur possible pour laquelle le résultat du clustering n'évolue plus, et avons choisi $minPts = 1$. Ces valeurs s'expliquent par la faible densité au global des arbres, et pour ainsi avoir tout arbre classifié dans un cluster, au lieu d'avoir des arbres non-classifiés (ce qui l'algorithme renverra si on augmente la valeur de $minPts$). L'algorithme produit ainsi une sorte de dendrogramme pour lequel une autre valeur ϵ_{cl} est nécessaire pour effectuer la coupe. Un certain nombre de recherche d'optimum existent, mais nous avons préféré choisir cette valeur de manière à obtenir la meilleure prédiction sur le jeu de test tout en maximisant l'homogénéité des clusters au sens de la V-mesure (Rosenberg et Hirschberg, 2007). Nous avons choisi la valeur $\epsilon_{cl} = 8$ ce qui donne 8227 clusters avec un score d'homogénéité d'environ 70%. Une nouvelle variable numérique catégorielle a été ajoutée au jeu de données en

prenant comme valeur l'identifiant du cluster. Le nombre de clusters est dû à ceux de taille 1 qui n'amènent aucune information : les clusters les plus gros sont plus cohérents (fig. 2).

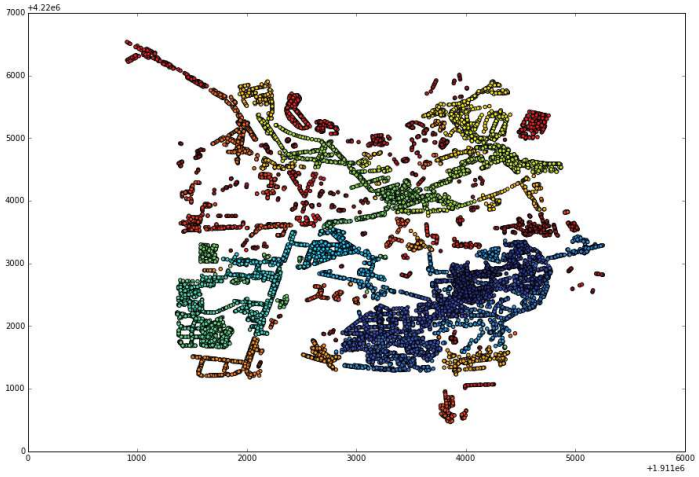

FIG. 2 – *Clustering obtenu par OPTICS*

2 Tâche 1 : prédiction des défauts de l'arbre

2.1 Modèles pour la prédiction unilabel

Dans un contexte supervisé, plusieurs algorithmes peuvent être utilisés pour l'apprentissage et la prédiction. Nous avons utilisé tout d'abord 2 modèles de classifieurs connus en Ensemble learning : les Random Forests (Breiman, 2001), et les Extremely Randomized Trees (appelés également Extra Trees) (Geurts et al., 2006). Pour faire simple, ces modèles sont composés d'un certain nombre d'arbres de décision qui ont chacun une vision partielle du problème. Le retour de la forêt correspond à l'ensemble des votes des arbres. Ces deux modèles sont implémentés dans la librairie Python Scikit-learn (Pedregosa et al., 2011) que nous avons utilisée.

2.2 Résultats pour la prédiction unilabel

La baseline du défi est de 86% pour l'exactitude, 82% de précision et 72% de rappel (valeurs que nous rappellerons entre parenthèses pour la comparaison). En utilisant que les données de base, nous obtenons les résultats du tableau 1 pour des forêts de 100 arbres sans optimisation particulière des paramètres. Les résultats sont similaires en terme d'exactitude. Nous avons ensuite ajouté la variable issue du clustering expliqué section 1.2.4, dont les résultats apparaissent dans le tableau 2, où l'on constate que cela a un impact positif sur le modèle ET. Dernier test avec ces modèles en ajoutant 50 variables numériques tirées du hachage du champ

REMARQUES expliqué section 1.2.3, dont les résultats sont donnés dans le tableau 3, et qui finalement ne fait gagner qu'un 1% d'amélioration sur la précision du modèle Extra Trees.

	RF	ET
Exactitude	**0,87** (0,86)	**0,87** (0,86)
Précision	0,81 (0,82)	0,78 (0,82)
Rappel	**0,79** (0,72)	**0,81** (0,72)

TAB. 1 – *Performances des modèles RF et ET (5-fold cross-validation) - Uni-label*

	RF	ET
Exactitude	**0,87** (0,86)	**0,87** (0,86)
Précision	0,81 (0,82)	0,79 (0,82)
Rappel	**0,79** (0,72)	**0,82** (0,72)

TAB. 2 – *Performances des modèles RF et ET (5-fold cross-validation) avec ajout de la variable issue du clustering OPTICS - Uni-label*

	RF	ET
Exactitude	**0,87** (0,86)	**0,87** (0,86)
Précision	0,81 (0,82)	0,79 (0,82)
Rappel	**0,79** (0,72)	**0,81** (0,72)

TAB. 3 – *Performances des modèles RF et ET (5-fold cross-validation) avec ajout de codes de hachage (champ REMARQUES) - Uni-label*

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, ainsi que la géolocalisation dans une moindre mesure, puis en dernier des caractéristiques propres à l'arbre [1].

Dans nos différents tests, nous avons voulu vérifier si seul l'expert pourrait fournir de bonnes prédictions, et si nous ne gardons que les variables propres à celui-ci (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, nous obtenons une exactitude de 86%, une précision de 83%, et un rappel de 71%. Sachant que nous atteignons 81% 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.

2.3 Modèles pour la prédiction multi-label

En classification multi-label, qui consiste en un problème d'apprentissage où plusieurs classes peuvent être affectées simultanément à un exemple, on trouve usuellement deux grandes catégories d'approches (Tsoumakas et Katakis, 2007) :

1. Participation des variables : NOTEDIAGNOSTIC : 0,41 - coord_x : 0,08 - coord_y : 0,07 - PRIORITEDERENOUVELLEMENT : 0,06 - DIAMETREARBREAUNMETRE : 0,05 - CODE_PARENT_DESC : 0,05 - CODE_PARENT : 0,04 - TRAVAUXPRECONISESDIAG : 0,03 - ESPECE : 0,02 - GENRE_BOTA : 0,02

— Transformer le problème pour le convertir en un problème pouvant être résolu par un algorithme de classification binaire ou multiclasses usuel ;
— Utiliser des algorithmes multiclasses transformés, qui peuvent résoudre directement le problème multi-label.

Nous avons cherché à exploiter des solutions multi-labels déjà existantes et rapidement implémentables, c'est pourquoi nous nous sommes tournés vers la seconde classe d'approche. Les modèles en Ensemble learning que nous avons utilisés pour la prédiction uni-label ont été adaptés pour traiter des problèmes de prédiction en multi-label, et sont également implémentés dans la librairie Python Scikit-learn. Dans les implémentations R, on trouve la méthode *Random Forest multivarié* de la librairie `randomForestSRC` qui est un algorithme directement dérivé des travaux de Breiman (Breiman, 2001) et la méthode *Random Ferns multi-label* de la librairie `rFerns`, inspirée par les classifieurs bayésiens (Ozuysal et al., 2010). Ces classifieurs peuvent être longs à entraîner sur le jeu de données du défi EGC, notamment en R. Nous nous sommes alors tournés vers une solution reconnue pour ses performances : le système d'apprentissage *out of core* (c'est-à-dire que les données d'apprentissage n'ont pas besoin d'être chargées entièrement en mémoire) Vowpal Wabbit (VW), développé par John Langford [2] au sein du laboratoire Microsoft Research et précédemment Yahoo ! Research. VW propose plusieurs implémentations d'algorithmes multi-classe, dont l'un qui peut être facilement "détourné" pour faire du multi-label : *Cost-sensitive one against all* (CSOAA). Cet algorithme combine trois éléments clés :

— *Classification multi-classe*, qui consiste à entraîner un classifieur par classe de réponse possible, où chaque classifieur sera spécialisé pour prédire l'une des classes. Le problème multi-classe est ainsi décomposé en plusieurs problèmes binaires (Biernat et Lutz, 2015) ;
— *Cost-senstitive learning*, une approche un peu différente d'une classification classique. On ne cherche pas à prédire une réponse positive ou négative, mais le coût associé à une réponse (Elkan, 2001). Dans les données d'apprentissage, une bonne réponse a un coût nul. Si on associe un coût égal à 1 à une mauvaise réponse, le *cost-senstitive learning* ressemble à "l'inverse" d'une classification classique. Mais on peut aussi pénaliser plus ou moins fortement certaines réponses avec des coûts différents de 1 ;
— *Réduction*, qui vise à transformer un problème d'apprentissage en un problème plus simple (Beygelzimer et al., 2015). Pour le CSOAA, VW transforme le problème initial en plusieurs problèmes de régression, chacun visant à prédire le coût associé à l'une des classes de réponse.

L'intérêt de cet algorithme VW est qu'il peut directement intégrer des exemples d'apprentissage avec une cible multi-label. En prédiction, CSOAA VW se comporte par défaut comme une prédiction multi-classe, mais on peut l'utiliser comme un classifieur multi-label en récupérant des fichiers intermédiaires de prédictions brutes, qui contiennent les coûts associés à chacune des décisions. Par défaut, VW va prédire la classe associée au coût le plus faible. Pour du multi-label, on peut prédire toutes les classes dont le coût est inférieur à un seuil donné.

2. https ://github.com/JohnLangford/vowpal_wabbit

2.4 Résultats pour la prédiction multi-label

Préparation des données En phase exploratoire, la librairie R `mldr` [3] permet de visualiser facilement des données et de représenter la problématique multi-label. Sur la figure 3, l'image de gauche 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'). La combinaison de maladies la plus fréquente est houppier + tronc, avec 539 occurrences. L'image de droite permet de visualiser les relations de co-occurrences entre maladies. Elle relie les maladies entre elles avec une surface qui croît avec le nombre d'occurrences de cette relation : on voit bien que houppier + tronc est la relation la plus fréquente.

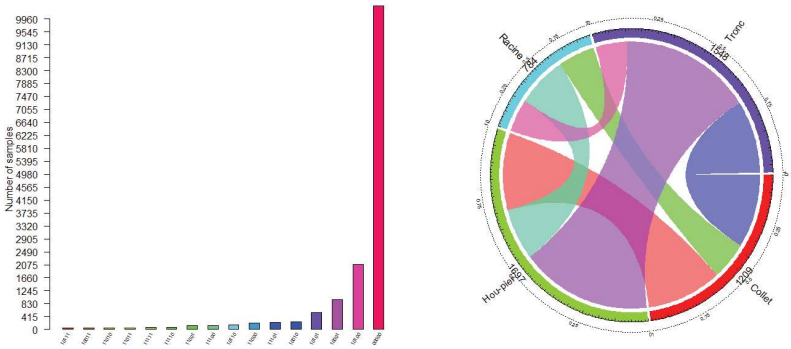

FIG. 3 – *Exploration de la problématique multi-label avec R* `mldr`

Nous souhaitons donc construire un classifieur CSOAA qui nous permette de prédire simultanément ces combinaisons de classes Sain ou Défaut / Collet / Houppier / Racine / Tronc. Pour cela, il faut transformer les données pour qu'elles puissent être ingérées par VW. La représentation de la problématique multi-label est spécifique à l'algorithme CSOAA. On associe une valeur numérique à chaque classe (Sain = 1, Collet = 2, Houppier = 3, Racine = 4, Tronc = 5). Pour chaque exemple, on associe ensuite le coût lié à chacune de ses réponses par une paire de points. Nous avons décidé d'associer le même coût de 1 pour chaque mauvaise réponse. Ainsi, un arbre sain se code `1:0 2:1 3:1 4:1 5:1`, un arbre malade du houppier et du tronc `1:1 2:1 3:0 4:1 5:0`. Nous pourrions chercher à affiner ces valeurs de coûts si on voulait pénaliser plus fortement certaines mauvaises réponses. Nous avons utilisé une technique un peu "brutale" mais communément utilisée par les férus de VW, qui consiste à déclarer chaque variable comme à la fois quantitative et qualitative.

Entraînement du modèle A partir des données transformées, nous entraînons le modèle en mode *batch* (et non pas en mode *online learning* qui est le fonctionnement par défaut de VW, c'est-à-dire qu'il réajuste ses paramètres à chaque nouvelle entrée, comportement que nous

3. https ://cran.r-project.org/web/packages/mldr/

	Micro	Macro
Précision	**0,82** *(0,70)*	**0,66** *(0,64)*
Rappel	**0,74** *(0,47)*	**0,46** *(0,37)*

TAB. 4 – *Performance du modèle CSOAA VW - multi-label*

Mesure	Résultat
Exactitude	0,86 (0,86)
Précision	**0,88** (0,82)
Rappel	**0,91** (0,72)

TAB. 5 – *Performance du modèle CSOAA VW - uni-label*

souhaitons éviter ici). Nous avons utilisé le paramétrage de base, mais le modèle pourrait être amélioré en jouant sur de nombreux paramètres de customisation (nombre de passes sur les données, etc.). Sur les données du défi, l'entraînement du modèle est quasi-instantané sur un ordinateur de bureau de bonne facture (processeur i7, 16Go de mémoire).

Prédiction et résultats Une ligne de prédiction brute aura l'apparence suivante :

$$1 : -0.0810122 \quad 2 : 1.18642 \quad 3 : 1.15586 \quad 4 : 1.07331 \quad 5 : 1.07771$$

On a donc la valeur numérique associée à chaque classe, suivie d'une prédiction de coût. Plus le coût est petit, plus on a de chance d'être face à une bonne réponse. Nous transformons alors ce résultat brut en un résultat multi-label classique : chaque classe associée à un coût inférieur à un certain seuil va être taguée positivement (c'est une bonne réponse, car son coût est faible), chaque classe associée à un coût supérieur à ce seuil sera taguée négativement (elle coûte "cher"). Nous avons choisi un seuil égal à 0.5, valeur qui pourrait peut-être être optimisée. Pour ce seuil, la ligne précédente devient, qui est un arbre sain :

$$1 : 1 \quad 2 : 0 \quad 3 : 0 \quad 4 : 0 \quad 5 : 0$$

Nous évaluons cette approche avec un validation croisée 5-*folds*. La performance est évaluée avec les métriques précision et rappel micro et macro disponibles dans `scikit-learn`. Concernant la baseline, il est proposé une précision micro de 70% et un rappel micro de 47%, ainsi qu'une précision macro de 64% et un rappel macro de 37%. Nos résultats sont présentés dans le tableau 4 (la *baseline* du défi est rappelée entre parenthèses).

Ce modèle peut être utilisé pour la prédiction uni-label, puisqu'il intègre directement la cible sain/malade (tab. 5). Sans aucun travail supplémentaire de *feature engineering*, on obtient donc un modèle performant, au-dessus de la *baseline* du défi (multi-label et uni-label), et également au dessus de nos premiers modèles utilisés pour la tâche de prédiction uni-label. Ce modèle pourrait encore être amélioré (paramétrage de l'entraînement, affinement du seuil, online learning...).

3 Tâche 2 : Enrichissement des données

Pour cette tâche, nous avons voulu isoler les données des experts qui permettent d'obtenir les résultats décrits pour la tâche 1 : en effet, sans cela, aucune amélioration significative n'était

observée par l'ajout de données externes. Les champs que nous avons considérés comme renseignés par un expert sont donnés section 2.2. Pour clarifier le discours, nous n'utiliserons que 2 mesures pour la comparaison : l'exactitude et la F-mesure (moyenne harmonique de la précision et du rappel). Sans les champs "experts", de base, nous obtenons au mieux *0.76 d'exactitude*, et *0.64 pour la F-mesure*, tout modèle confondu.

3.1 Données sur les maladies

Pour enrichir les données présentes, nous avons étudié la question du risque pour les arbres de contracter une maladie. Pour ce faire, nous nous sommes largement appuyés sur le guide d'observation et de suivi des organismes nuisibles en zones non agricoles réalisé par "Plante & Cité" sous le pilotage du Ministère chargé de l'agriculture (Plante et Cité, 2011). Dans ce document, il y est répertorié les sensibilités de plantes-hôtes en fonction de leur groupe agronomique. En tout, 5 niveaux de sensibilités y sont décrits (de 4 à 0) : très sensibles, sensibles, moyennement sensibles, peu sensibles, et non sensibles (quand absent de la liste). Au niveau des maladies et autres nuisibles, nous avons extrait ce niveau de sensibilité pour 31 d'entres elles rattaché à chaque groupe botanique. Ainsi, pour chaque genre d'arbre, nous lui avons associé une probabilité (0 à 1) de contracter une maladie, en fonction de sa sensibilité : $4 \rightarrow 1.0$, $3 \rightarrow 0.75$, $2 \rightarrow 0.5$, $1 \rightarrow 0.25$, $0 \rightarrow 0.0$. Nous avons ensuite construit une simulation multi-agents définie comme suit :

— E : environnement (espace géographique discrétisé)
— A : ensemble des agents (arbres)
— M : ensembles des maladies ($|M| = 31$)
— D : opérations de diffusion des maladies

Chaque agent (arbre) possède les caractéristiques suivantes : en fonction de son genre botanique, une probabilité de contracter une maladie (une par maladie, soit 31), un ensemble de valeurs $\{q_1, ..., q_{31}\}$ initialisées à zéro, qui correspondra au nombre de fois que cet arbre aura été infecté par un autre arbre pour une maladie donnée. Le modèle de diffusion utilisé est très simple : un arbre donné a une certain probabilité d'être infecté dans un rayon r défini arbitrairement (fig. 4 à droite). Il est complexe de construire un modèle précis qui aurait réagi en fonction de la nature de la maladie, et nécessiterait une étude à part entière à l'instar de Chapman et al. (2015) qui liste pas moins de 468 modèles sur le sujet. Un aperçu de la simulation est donné figure 4 à gauche. Celle-ci a été réalisée avec Netlogo [4].

Voici le déroulé de la simulation :

— Pour chaque arbre $a \in A$
  — Pour chaque maladie $m \in M$
    — Pour chaque arbre voisin v de a dans un rayon r
      1. Générer une valeur flottante aléatoire entre 0 et 1 (loi uniforme)
      2. Déterminer si la valeur générée est supérieure à la probabilité de l'arbre de contracté la maladie m
      3. Si oui, l'arbre v incrémente son nombre d'infections contractées pour la maladie m : $q_m = q_m + 1$

Pour enrichir les données d'origine, nous avons fait jouer plusieurs simulations (environ 10), avec plusieurs rayons (de 1 à quelques mètres) et avons ajouté pour chaque arbre 31 colonnes correspondant au ratio d'infection global par maladie. En faisant rejouer le modèle, nous n'obtenons malheureusement qu'un faible gain, avec une exactitude qui passe de 0.76 à 0.77 (pas de changement pour la F-mesure).

4. http ://ccl.northwestern.edu/netlogo/

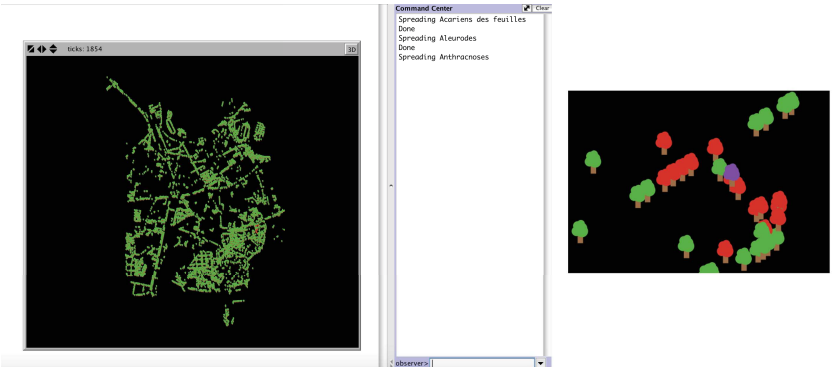

FIG. 4 – *Simulation de propagation de maladies*

Station	Latitude	Longitude	$PM10$	O_3	NO_2
1	45.190425	5.686977	269	290	18
2	45.180708	5.720161	471	_	345
3	45.182825	5.753152	220	299	84
4	45.158395	5.703683	648	_	329
5	45.161876	5.735583	290	202	4

TAB. 6 – *Scores cumulés de pollution à Grenoble sur l'année 2015*

3.2 Données sur la pollution

Nous avons également voulu vérifier si l'impact de la pollution de la ville pouvait ou non avoir un rapport avec le fait d'avoir un arbre malade ou non. Nous nous sommes intéressés à 3 composants polluants connus pour avoir des effets nocifs sur les arbres (Gillig et al., 2008) :

— *les particules* $PM10$ (diamètre inférieur à 10 μm) : peut former des couches gênant l'absorption lumineuse ou provoquant la corrosion et des lésions de la cuticule foliaire.
— *l'ozone* (O_3) : entraine des lésions sur la surface supérieure des feuilles, des tâches punctiformes, des chloroses et des nécroses.
— *le dioxyde d'azote* (NO_2) : contribue à la formation de pluies acides.

Les données ont été extraites de la plate-forme nationale de prévision de la qualité de l'air, PREV'AIR [5], dont l'outil permet de connaitre le taux de pollution journalier (maximum ou moyenne) par station. Nous avons travaillé sur les données journalières maximales issues de l'année 2015, en transformant les 6 intervalles de concentration en score de pollution (de 0 à 5). Si une station relève une pollution au NO_2 entre 80 et 120 $\mu g/m^3$, cela donnera un score de 1, la station relevant une pollution entre 120 et 160 $\mu g/m^3$ donnant un score de 2. Pour NO_2, les valeurs réelles vont de 0 à plus de 400 $\mu g/m^3$, pour $PM10$, les valeurs réelles vont de 0 à plus de 125 $\mu g/m^3$, et pour O_3, les valeurs réelles vont de 0 à plus de 240 $\mu g/m^3$.

Le tableau 6 récapitule les 5 stations relevant les scores cumulés, la valeur pour chaque polluant correspondant à la somme des scores journaliers sur l'année 2015 (pollution cumulée).

5. http ://www2.prevair.org

A partir de ces données, nous donnons une valeur de pollution cumulée pour chaque arbre par interpolation via la pondération inverse à la distance (exemple d'interpolation des scores de pollution NO_2, fig. 5, du moins pollué (bleu) au plus pollué (rouge)). En ajoutant les scores des 3 polluants à nos modèles pour chaque arbre, nous obtenons une exactitude de 0.78 (contre 0.76 sans) et une F-mesure de 0.67 (contre 0.64 sans), ce qui permet de déduire un certain gain de la donnée liée à la pollution pour notre tâche de prédiction.

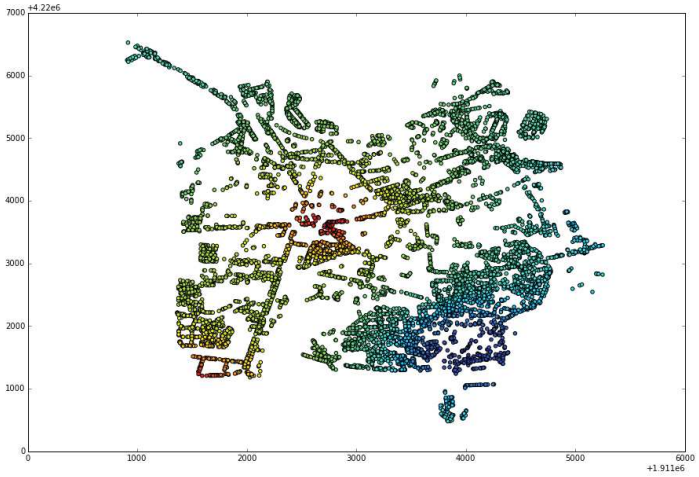

FIG. 5 – *Pollution au NO_2 cumulée sur l'année 2015*

4 Conclusion

Notre travail a permis de proposer une approche pour répondre au défi EGC'2017. Dans un premier temps, nous avons travaillé sur les variables (feature engineering) afin de pouvoir les rendre utilisables dans des modèles d'apprentissage supervisé, et d'en sortir un maximum de signal. Pour la tâche 1, après avoir obtenu des premiers résultats avec des modèles "classiques" (RF, ET), nous avons utilisé un modèle plus avancé (CSOAA) qui nous a donné les meilleurs résultats de notre étude. Concernant la tâche 2, d'autres données ont été considérées (maladies, pollution) afin de confirmer l'intérêt d'enrichir le jeu de données avec des informations extérieures, même si le gain peut rester faible. D'autres pistes seraient à creuser en fonction de la disponibilité des données, comme par exemple la nature du sous-sol, ainsi que l'aspect temporel (état des arbres dans le temps, météo).

Références

Andoni, A. et I. Razenshteyn (2015). Optimal Data-Dependent Hashing for Approximate Near Neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, New York, NY, USA, pp. 793–801. ACM.

Ankerst, M., M. M. Breunig, H.-P. Kriegel, et J. Sander (1999). OPTICS. In *Proceedings of the 1999 international conference on Management of data - SIGMOD'99*. ACM.

Beygelzimer, A., J. Langford, et B. Zadrozny (2015). Machine learning techniques - Reductions between prediction quality metrics. University Lecture.

Biernat, E. et M. Lutz (2015). *Data science : fondamentaux et études de cas*. Eyrolles.

Breiman, L. (2001). Random Forests. *Machine Learning 45*(1), 5–32.

Chapman, D., S. White, D. Hooftman, et J. Bullock (2015). Inventory and review of quantitative models for spread of plant pests for use in pest risk assessment for the EU territory. *EFSA Supporting Publications 12*(4), 795E–n/a. 795E.

Elkan, C. (2001). The Foundations of Cost-sensitive Learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, San Francisco, CA, USA, pp. 973–978. Morgan Kaufmann Publishers Inc.

Geurts, P., D. Ernst, et L. Wehenkel (2006). Extremely randomized trees. *Machine Learning 63*(1), 3–42.

Gillig, C.-M., C. Bourgery, et N. Amann (2008). *L'arbre en milieu urbain : plantations, conception et mise en œuvre*. Infolio.

Kondrak, G. (2005). N-gram similarity and distance. In *Proc. Twelfth Int'l Conf. on String Processing and Information Retrieval*, pp. 115–126.

Ozuysal, M., M. Calonder, V. Lepetit, et P. Fua (2010). Fast Keypoint Recognition Using Random Ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*(3), 448–461.

Pedregosa, F. et al. (2011). Scikit-learn : Machine Learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

Plante et Cité (2011). Guide d'observation et de suivi des organismes nuisibles en ZNA.

Rosenberg, A. et J. Hirschberg (2007). V-Measure : A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning(EMNLP-CoNLL)*, pp. 410–420.

Tsoumakas, G. et I. Katakis (2007). Multi-label classification : an overview. *International Journal of Data Warehousing and Mining 3*(3), 1–13.

Summary

The EGC'2017 conference proposes a contest whose context is the management of green space for the city of Grenoble, focused on trees. The aim is to propose a model based on the data provided that would better predict the diseased trees, as well as the potential location of the disease. After getting some interesting results with standard models, our approach using a Cost-Sensitive One Against All model (CSOAA) allows us to obtain an accuracy of 0.86, a precision of 0.88, and a recall of 0.91 for the unilabel prediction, and a precision/recall micro of 0.82/0.74 though a precision/recall macro 0.66/0.46 for multilabel prediction. The extraction of knowledge for task 2 allowed us to highlight the interest of adding data about diseases and the concentration of pollution in the city.