Projet FDEC - Prétraitement des données

Nous avons à notre disposition 29 variables nous permettant de prédire s'il y a un défaut sur l'arbre (uni-label) et où (multi-label).

En guise de pré-traitement des données, nous pouvons commencer par **supprimer les variables qui sont inutiles à la prédiction**, dans l'ordre :

- ADR_SECTEUR : en effet, pour l'aspect géographique, nous allons regrouper les arbres par secteurs géographiques grâce aux coordonnées, ce qui va nous permettre de diviser Grenoble en plus de 6 secteurs géographiques (voir plus loin) ;
- CODE : elle correspond à l'identifiant unique de l'arbre, elle n'apporte aucune information sur le défaut potentiel de l'arbre ;
- CODE_PARENT_DESC : elle correspond à l'endroit où se trouve la station, ce qui n'apporte aucune information supplémentaire ;
- IDENTIFIANTPLU : il s'agit encore d'un identifiant unique ;
- SOUS_CATEGORIE : il s'agit de l'identifiant de la sous-catégorie. Nous allons plutôt garder la variable SOUS_CATEGORIE_DESC car elle est plus lisible ;
- STADEDEDEVELOPPEMENT : la variable STADEDEDEVELOPPEMENTDIAG est plus pertinente (nous avons compris que STADEDEDEVELOPPEMENT correspond au stade théorique de développement, et STADEDEDEVELOPPEMENTDIAG correspond au stade réel).

Par la suite on va se changer de **gérer les données manquantes** (les "?") :

- pour les **données catégorielles**, si on a plus de 1% de données manquantes, on remplace par "valeur manquante" ; sinon, on remplace par la valeur la plus présente.
- pour les **données numériques**, nous pouvons remplacer les valeurs manquantes par la valeur moyenne (arrondie).

Dans un troisième temps nous allons convertir les **variables catégorielles** en format numérique afin d'être traité par nos algorithmes d'apprentissage supervisé.

Nous allons principalement utiliser trois types de transformations :

- **Ordre important** : Utiliser **Ordinal Encoding**.
  Exemples : Taille de vêtements (petit, moyen, grand), niveau de satisfaction (faible, moyen, élevé).
- **Pas d'ordre et peu de catégories** : Utiliser **One-Hot Encoding**.
  Exemples : Couleurs, type de produit, jours de la semaine.
- **Pas d'ordre et trop de catégories** : Utiliser **Label Encoding**.
  Exemples : Noms de villes, pays, identifiants uniques (comme les ID d'utilisateurs).

VARIABLE	TYPE	TRANSFORMATION
ANNEEDEPLANTATION	numérique	/
ANNEEREALISATIONDIAGNOSTIC	numérique	/
ANNEETRAVAUXPRECONISESDIAG	numérique	/
CODE_PARENT	nominale	Label Encoding
DIAMETREARBREAUNMETRE	ordinale	Ordinal Encoding
ESPECE	nominale	Label Encoding
FREQUENTATIONCIBLE	ordinale	Ordinal Encoding
GENRE_BOTA	nominale	Label Encoding
INTITULEPROTECTIONPLU	nominale	One-Hot Encoding
NOTEDIAGNOSTIC	ordinale	Ordinal Encoding
PRIORITEDERENOUVELLEMENT	ordinale	Ordinal Encoding
RAISONDEPLANTATION	nominale	One-Hot Encoding
REMARQUES	textuelle	/
SOUS_CATEGORIE_DESC	nominale	One-Hot Encoding
STADEDEVELOPPEMENTDIAG	ordinale	Ordinal Encoding
TRAITEMENTCHENILLES	ordinale	Ordinal Encoding
TRAVAUXPRECONISESDIAG	nominale	Label Encoding
TROTTOIR	binaire	Label Encoding
TYPEIMPLANTATIONPLU	nominale	One-Hot Encoding
VARIETE	nominale	Label Encoding
VIGUEUR	ordinale	Ordinal Encoding

Après numérisation des données, nous allons **normaliser et standardiser les valeurs**.

Pour les données textuelles avec beaucoup de valeurs différentes comme **REMARQUES.**On peut utiliser l'algorithme LSH, pour simplifier la recherche par les plus proches voisins, on peut également utiliser n-grams (n=3) qui permet de trouver les similarités entre les différents textes.On va procéder comme suit :

- Étape 1 :

On va devoir prétraiter les textes pour qu'ils soient plus uniformes et plus cohérents. On va :

- Tout mettre en minuscule : "Arbre" devient "arbre"
- Retirer la ponctuation
- Supprimer les stop-words : on enlève les mots peu informatifs telles que "le" "de" etc.
- Tokenizer : "L'arbre est en bonne santé" devient ["arbre","bonne","santé"]

- Étape 2 :

On va devoir créer des groupes de mots (n-grms de mots) avec n mots adjacents dans le texte. Par exemple :

Texte : "L'arbre est en bonne santé"

- **1-gram (unigramme)** : ["arbre", "bonne", "santé"]
- **2-grams (bigrams)** : ["arbre bonne", "bonne santé"]
- **3-grams (trigrams)** : ["arbre bonne santé"]

- Étape 3 :

Une fois les n-grams extraits, chaque texte peut être représenté sous forme de vecteur où chaque dimension correspond à un **n-gram unique** présent dans l'ensemble des textes.

- **Méthode de bag-of-n-grams** :
  1. Créez un vocabulaire de tous les n-grams uniques présents dans les textes.
  2. Représentez chaque texte comme un vecteur où chaque élément du vecteur indique la fréquence d'un n-gram spécifique dans ce texte.
     - Exemple : Si les n-grams "arbre bonne" et "bonne santé" apparaissent dans votre vocabulaire, un texte serait représenté par un vecteur comme [1, 1] si les deux n-grams apparaissent une fois dans ce texte.

- Étape 4 :

Avec les vecteurs de n-grams, on va pouvoir utiliser des méthodes de similarité pour comparer les textes entre eux. Voici les deux méthodes les plus utilisées :

1. **Similarité cosinus** :

- Mesure l'angle entre les vecteurs de deux textes.
- Valeur proche de 1 = textes très similaires, proche de 0 = textes très différents.

$$\text{similarité cosinus}(A, B) = \frac{A \cdot B}{||A|| \times ||B||}$$

- Où AA et BB sont les vecteurs de n-grams des deux textes à comparer.

2. **Indice de Jaccard** :
   - Compare l'intersection et l'union des ensembles d'n-grams pour deux textes.

$$\text{Indice de Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Si les deux textes partagent beaucoup de n-grams, la similarité sera élevée (proche de 1).

Exemple concret

Texte 1 :

"Arbre en bonne santé"

- Après tokenisation : ["arbre", "bonne", "santé"]
- **Bigrams** : ["arbre bonne", "bonne santé"]

Texte 2 :

"Cet arbre est en bonne santé"

- Après tokenisation : ["arbre", "bonne", "santé"]
- **Bigrams** : ["arbre bonne", "bonne santé"]

Représentation des bigrams :

Si notre vocabulaire de bigrams contient ["arbre bonne", "bonne santé", "vert arbre"], alors les vecteurs seraient :

- Texte 1 : [1, 1, 0] (car "arbre bonne" et "bonne santé" apparaissent, mais pas "vert arbre")
- Texte 2 : [1, 1, 0] (les mêmes bigrams apparaissent)

Calcul de similarité cosinus :

- Similarité cosinus entre les deux vecteurs [1, 1, 0] et [1, 1, 0] serait 1, indiquant que les deux textes sont identiques en termes de bigrams.

- Étape 5 :

Dans le cas d'un nombre massif de données, celà peut prendre du temps, et peut être coûteux en termes de calcul. L'algorithme LSH permet de réduire cette complexité, en regroupant les textes similaires dans des "buckets" avant d'effectuer des comparaisons. Le principe de LSH est de transformer les vecteurs en "empreintes" via des fonctions de hachage, de telle sorte que des textes similaires obtiennent des valeurs de hachage proches.

En ce qui concerne la localisation, comme mentionné précédemment, nous avons décidé de retirer ADR_SECTEUR afin d'utiliser uniquement les coordonnées pour **clusteriser le lieu**. Ceci nous permettra de diviser Grenoble en plus de 6 secteurs géographiques. Pour cela, nous allons utiliser l'**algorithme des k-means** pour réaliser un clustering (avec **n=400**, choisi en testant plusieurs valeurs de n et en choisissant celui avec le meilleur indice de silhouette). Nous ajouterons ensuite le numéro de cluster aux données à la place des coordonnées.