Apprentissage supervisé (Partie 1)

1. *Accès aux données / Analyse globale*

- ***Lecture*** *: Utilisez fonction read.csv la bibliothèque pandas pour récupérer les données.*
- ***Features et samples*** *:*
  1. *En utilisant la fonction dtypes et celles permettant de compter les valeurs, quelle est la taille du DataSet, le nom des variables et leurs types.*
  2. *Combien y a-t-il de variables de types (object, int et float).*
- ***Modèle et Target*** *:*
  1. *Identifier la target, quel est le type d'apprentissage du modèle (régression ou classification).*
  2. *Analysez les différentes valeurs de la target.*
  3. *Quelle est la répartition de ces valeurs (pourcentage), y-a-t-il une répartition équilibrée entre les différentes valeurs.*
- ***Valeurs Nan et doublons*** *:*
  1. *Y a-t-il des doublons.*
  2. *Y a-t-il des valeurs manquantes dans le DataSet. Vous pourrez pour cela utiliser soit la fonction info, soit la fonction isnull.*
  3. *Créer une liste de toutes les features avec des valeurs manquantes. Vous pourrez utiliser le boolean indexing basé sur le nombre de valeurs nulles.*
  4. *Quel est la répartition des valeurs manquantes par rapport à la target. Vous afficherez le nombre de 0 et de 1 de ces features, ainsi que le pourcentage de 1 des valeurs manques. Est-ce que ce pourcentage est très différent du pourcentage de 1 dans tout le dataSet.*
- ***Analyse des features :***
  1. *Affichez les statistiques sur les données en utilisant la fonction describe() que peut-on en conclure en comparant cette analyse avec les statistiques sur les valeurs 1 et 0 de la target. Pensez-vous qu'il y ait des valeurs extrêmes. On considèrera qu'une valeur est*

extrême, pour la potabilité, si elle s'écarte de la moyenne de plus de 3 fois la valeur la dispersion (std).

2. *Utilisez la fonction boxplot de la bibliothèque seaborn, pour visualiser la répartition des variables, et confirmer ce qui pouvait apparaitre sur l'analyse statistique.*
3. *Comment se répartissent les données en fonction la target, vous utiliserez pour cela les fonctions histplot ou kdeplot de seaborn. Ces courbes confirment-elles l'analyse sur les données statistiques.*
4. *Affichez les taux de corrélation. Pensez-vous que les variables soient liées entre elles.*
5. *Utilisez la fonction pairplot avec une séparation des affichages en fonction de la target (option hue).*

2. *Premier modèle*

- ***Données manquantes** : Créez une fonction nan_values qui supprime les données manquantes.*
- ***Split des données** : Créez une fonction split_data qui découpe les données en un testSet et un trainSet avec un taux de 80%. Cette fonction recevra trois paramètres, un dataFrame, la liste des colonnes sans la target et la target. Cette fonction retournera les dataFrames créés.*
- ***Fonction d'évaluation** : Créez une fonction évaluation(model, df) qui doit évaluer un modèle et afficher les résultats obtenus.*
    1. *Cette fonction débute par la copie du dataframe df, puis l'appel des fonctions de traitement des valeurs nulles et le découpage du dataSet.*
    2. *Entrainez le modèle (fonction fit), puis calculez les sorties estimées (fonction predict).*
    3. *Affichez ensuite les performances du modèle. Pour cela vous utiliserez les fonctions confusion_matrix et classification_report de la bibliothèque sklearn.metrics.*
- ***Arbre de décision** :*
    1. *Créer un modèle d'arbre de décision DecisionTreeClassifier puis évaluer-le.*
    2. *Quel sont les résultats obtenus sur les données du test et du train.*
    3. *Profondeur de l'arbre : Testez pour le model quelle est la meilleure profondeur pour l'arbre, vous testerez les profondeurs de 1 à 50. Quel est le meilleur résultat, pour votre modèle.*

4. *Utilisez maintenant un GridSearchCV avec des valeurs pour les variables min_samples_split [10,20,30], min_samples_leaf et max_leaf_nodes, [10,20,100], max_depth=10, pour ces paramètres. Les paramètres de GridSearchCV sont fournis sous la forme de dictionnaire, vous choisirez une cross validation de 5.*
5. *Evaluer le modèle avec best_estimator_ pour afficher les résultats obtenus.*
6. *En utilisant les résultats de feature_importances_ des arbres de décision quels sont les variables qui ont le plus d'importance pour l'évaluation de la qualité de l'eau.*

- ***Modification des données manquantes** :*
    1. *On souhaite maintenant ne pas supprimer les données manquantes mais les remplacer par la valeur moyenne des données de leur colonne. Créez la liste des colonnes ayant des données manquantes, puis remplacez les données manquantes (fillna()) par la valeur moyenne de la colonne (mean()).*
    2. *Combien de valeur sont testées et quel est le résultat obtenu.*
    3. *Remplacer les valeurs manquantes en fonction de la moyenne de la classe à laquelle elles appartiennent. Pour cela vous utiliserez dans la fonction fill les valeurs issues des groupby sur lesquels vous aurez appliqué la fonction transform(mean) pour obtenir la moyenne des groupes en fonction de la classe.*
    4. *Evaluer le modèle, commentez les résultats obtenus. Quels sont les variables qui ont le plus d'impact sur le résultat.*
    5. *On souhaite maintenant voir si en remplaçant les valeurs manquantes par celles qui sont les plus proches le modèle est amélioré. On utilise alors un KNNimputer avec 3 voisins, vous appliquerez le transformer sur un premier dataFrame composé des samples potables puis sur un second composé des non-potables. Vous pourrez ensuite créer un dataframe composé des deux résultats.*
    6. *En utilisant un KNNimputer transformez les données manquantes en les remplaçant pas les données qui lui ressemble le plus. Pour cela vous transformerez les données.*

3. *Changement de modèle*

- ***Données déséquilibrées*** *: Le déséquilibre entre les classes peut conduire à de mauvaises interprétations. Nous allons donc essayer d'équilibrer les données dans les classes.*
    1. *Pour commencer nous devons remplacer les données manquantes par la moyenne de leur classe. Puis créer un X_train et un X_test à partir du train_test_split sur le Data complet. Le train et le test doivent donc contenir toutes les features et aussi la target.*
    2. *On choisit ensuite de réduire le nombre samples de la classe majoritaire dans le jeu d'apprentissage (X_train). Pour cela on doit créer deux dataframes pour les potables et les non potables. Puis la taille du dataFrame des données non potables seront réduite à la taille des données potables. Vous utiliserez pour cela la fonction sample des dataFrame avec comme paramètre la taille des données potables.*
    3. *Vous devrez ensuite joindre (concat) la classe minoritaire avec la classe majoritaire réduite.*
    4. *Effectuez l'apprentissage sur les données X_train réduit et la prédiction sur les données X_test d'origine. Analysez les résultats obtenus.*
    5. *En utilisant la fonction sample sur la classe minoritaire, avec l'option replace=True, et la taille de la classe majoritaire, des données sont dupliquées afin de rééquilibrer les classes. Quel est le résultat obtenu. Il existe d'autres techniques de sur-échantillonnage des données comme la fonction SMOTE de imblearn.over_sampling ou resample de sklearn.utils.*
- ***RandomForest*** *: On souhaite maintenant modifier le modèle.*
    1. *Créer un modèle de type RandomForestClassifier puis en avec un GridSearchCV rechercher quels sont les meilleurs paramètres pour les valeurs suivantes min_samples_split et max_depth : [10, 20, 30], min_samples_leaf et max_leaf_nodes : [10, 20, 100].*
    2. *Entrainez le modèle à partir d'un jeu de données avec les données manquantes remplacées, quels sont les meilleurs paramètres du modèle.*
    3. *Réduisez la classe majoritaire pour les données d'entrainement, et évaluer le modèle.*
    4. *Quel est l'influence des différentes variables sur la target.*