Institut National des Sciences Appliquées
Université de Rennes

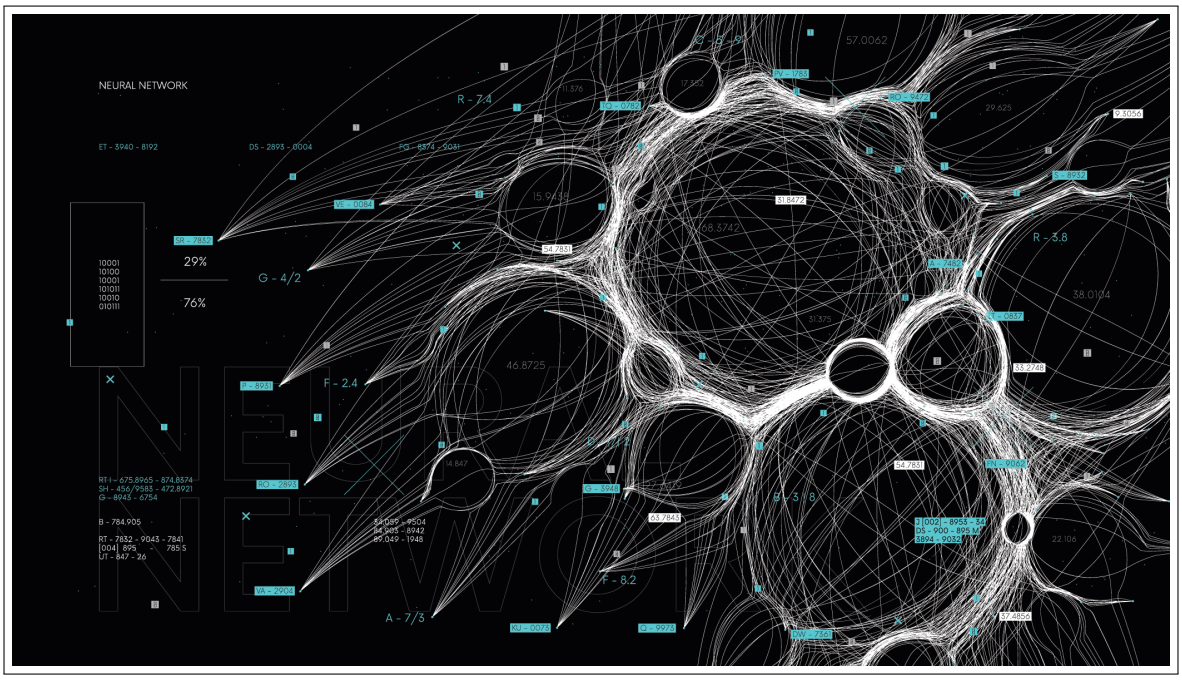
Attaques par Inférence d'Appartenance (MIA)

Une première approche avec le concours Snake Strikes Back

État d'avancement du projet au 9 décembre

Auteurs :
Thomas Aubin
Selyan Da Silva
Moussa Ouassou
Émile Peltier

Responsable de projet :
Cédric Eichler
Créateurs de la compétition :
Tristan Allard
Mathias Bernard



Mots-clés : Machine Learning, Confidentialité, Classification, Séries Temporelles, Apprentissage supervisé

8 décembre 2024

Résumé

Ce document est une version intermédiaire du rapport de projet à rendre pour la fin du module. De nombreuses parties, notamment les plus théoriques, doivent encore être complétées, le groupe s'étant concentré sur l'aspect opérationnel de la compétition, c'est-à-dire être en mesure de réaliser une prédiction d'appartenance d'un dataset à un autre, aussi fausse puisse-t-elle être.

Cette phase a été accomplie, aussi sont présentées plus bas *(partie VIII.2)* les conclusions du groupe à ce stade. Seule la tâche 2 a été abordée car elle semblait la plus simple.

Les chapitres VI et VII expliquent les décisions qui ont mené à ces résultats, la partie II présentant la compétition et la partie I, plus académique, récapitulant les notions à connaître pour mener celle-ci.

À ce stade, seul l'attaque par *Shadow Models* est envisagée, les autres méthodes étant trop incertaines ou compliquées à maîtriser. Voici néanmoins quelques pistes abordées puis écartées :

— Adopter une approche "whitebox" et changer les hyperparamètres du GAN ou même son fonctionnement interne
— Utiliser un apprentissage non supervisé *(impossible en l'absence d'oracle)*

Pour l'ensemble du rapport, les notations suivantes sont adoptées.

— $\mathbb{S}$ désigne un ensemble de données comprises entre 0 et 1 structurées en 7 classes
— $\mathbb{C}$ est un ensemble du même format et de taille 100. Les attaquants doivent déterminer si chacune d'entre elles est un membre ou non du dataset privé
— $\mathcal{G}$ est l'abréviation de "DopelGanger", le modèle génératif que l'on cherche à attaquer
— $\mathcal{C}$ est un algorithme de classification visant à attribuer la classe 0 ou 1 à chaque tuple d'un $\mathbb{S}$ donné, cet attribut désignant l'appartenance ou non au dataset public.

Table des matières

Table des figures

Liste des tableaux

Liste des Équations

Table des éléments de code

Introduction

...

Bien que le projet ait pour coeur la participation à la compétition, celui-ci a nécessité un important travail de montée en compétences et de documentation en Machine Learning pour l'ensemble du groupe, ce domaine n'étant que peu abordé à ce stade de la formation. C'est pourquoi la partie opérationnelle et technique du projet est précédée d'une part d'un court travail de bibliographie ayant pour visée la synthèse des connaissances mathématiques et algorithmiques indispensables à la participation au concours, et d'autre part par une présentation des tenants et aboutissants du concours, laquelle prend soin d'expliquer le plus finement possible les données sur lesquelles nous nous entraînons ainsi que le modèle attaqué.

...

Première partie

Notions d'*Adversarial Machine Learning*

Chapitre I

Concepts utiles d'Intelligence Artificielle

I.1 Fonctionnement général du Machine Learning

I.2 Algorithmes de classification

I.2.1 Intérêt et fonctionnement de la classification

I.2.2 Exemples d'algorithme de classification

I.2.2.1 Régression logistique

I.2.2.2 Bayésien naïf

I.3 Réseaux de neurones et Deep Learning

I.3.1 Princice du Deep Learning

I.3.2 Un modèle à deux réseaux : le *Generative Adversarial Network* (GAN)

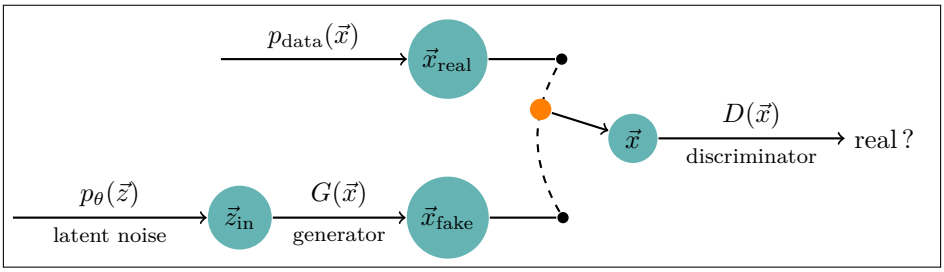


FIGURE I.1 – Schéma haut niveau d'un GAN

Chapitre II

Attaques par Inférences d'Appartenance : contextualisation du projet

II.1 Types d'attaques

II.2 Conséquences d'une attaque MIA réussie

Deuxième partie

Le concours *Snake Strikes Back* : position du problème

Chapitre III

Contexte et enjeux de la compétition

III.1 Principe

Un modèle génératif de type GAN est présenté, dénoté $\mathcal{G}$. À l'inverse des GAN classiques généralement utilisés pour générer des images, celui-ci génère des séries temporelles, c'est-à-dire des ensembles ordonnées de données mesurées à intervalles réguliers.

Le but de la compétition est de trouver les éventuelles faiblesses du GAN en matière de confidentialité, c'est-à-dire s'il est possible de connaître ses données d'entraînement à partir de ses seules données de sortie *dites données "synthétiques"* $\mathbb{S}_{Synth}$ dans le cas d'une étude "black-box" ou de ces mêmes données et du modèle lui-même pour une étude "white-box".

III.1.1 *Taxi Trips (2013-2023)* : un dataset public remanié comme donnée d'entrée

Le cas considéré ici est tiré d'un dataset disponible publiquement présentant 23 caractéristiques de quelque 200 millions de voyages en taxi dans la ville de Chicago, entre 2013 et 2023.

Pour la compétition, on extrait de ces données un nouveau dataset comportant les revenus journaliers d'un taxi sur deux semaines. Des opérations de normalisation et de lissage aux extrêmes sont conduites mais non présentées ici.

Enfin, le dataset présenté aux équipes comporte 1 million de lignes et 7 colonnes correspondant, les revenus de la deuxième semaine étant reportés sur une nouvelle ligne.

III.1.2 Objectifs et scoring

Les équipes se voient confier 4 tâches $T_i, i \in [1;4]$ dont la particularité est expliquée plus bas. Pour chacune de ces 4 tâches, 100 lignes issues des deux datasets publics $\mathbb{S}_{Pub_{T_1,T_2}}$ et $\mathbb{S}_{Pub_{T_3,T_4}}$, sont regroupées dans un ensemble "cible" que l'on appellera $\mathbb{C}_i, i \in [1;4]$.

L'objectif de l'équipe est de déterminer si chaque ligne appartient effectivement aux datasets d'entraînement $\mathbb{S}_{Pri_i}$ de $\mathbb{G}$

Le score de chaque tâche est basée sur la différence entre le taux de vrais et de faux positifs, la formule complète étant détaillée dans [2].

III.2 Les particularités de chacune des 4 tâches

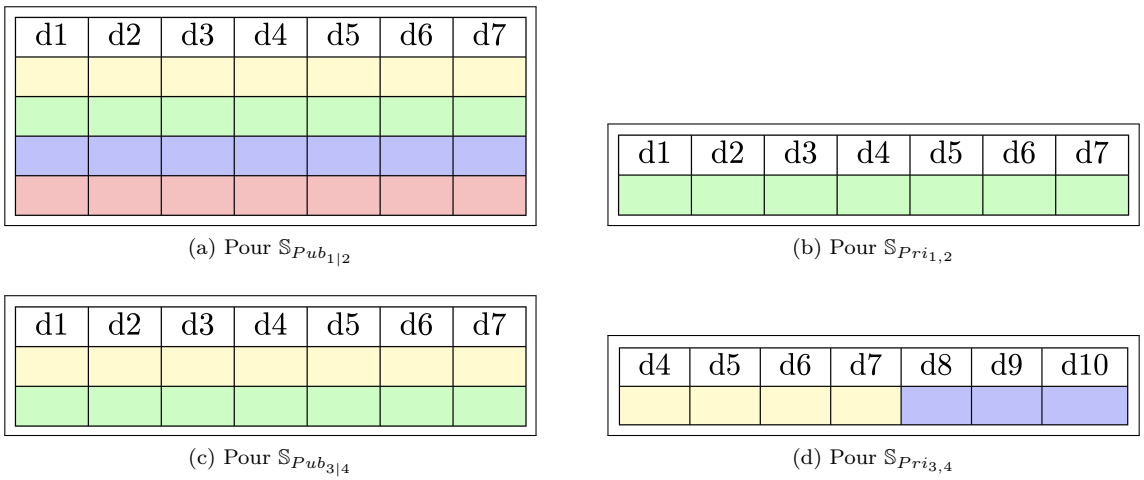
III.2.1 Taille de l'échantillon d'entraînement

Le nombre de lignes varie d'un facteur 10 entre les tâches 1-3 et 2-4. Ainsi :
— $\mathbb{S}_{Pri_1}$ et $\mathbb{S}_{Pri_2}$ contiennent 10000 lignes
— $\mathbb{S}_{Pri_1}$ et $\mathbb{S}_{Pri_2}$ contiennent 1000 lignes

III.2.2 Connaissance *a priori* des données d'entraînement

Les lignes de $\mathbb{S}_{Pri_1}$ et $\mathbb{S}_{Pri_2}$ sont des copies exactes des lignes de $\mathbb{S}_{Pub_{1|2}}$.

Les tâches 3 et 4 sont beaucoup plus subtiles, car les lignes de $\mathbb{S}_{Pri_3}$ et $\mathbb{S}_{Pri_4}$ ne correspondent pas exactement à celles de $\mathbb{S}_{Pub_{3|4}}$. Chaque ligne est en fait une concaténation des quatre premiers jours d'une ligne de $\mathbb{S}_{Pub_{3|4}}$ et de trois jours n'appartenant pas à ce dataset, et par conséquent hors de la connaissance de l'attaquant.



(a) Pour $\mathbb{S}_{Pub_{1|2}}$

(b) Pour $\mathbb{S}_{Pri_{1,2}}$

(c) Pour $\mathbb{S}_{Pub_{3|4}}$

(d) Pour $\mathbb{S}_{Pri_{3,4}}$

FIGURE III.1 – Exemples de lignes de $\mathbb{S}_{Pub_{1|2}}$, $\mathbb{S}_{Pub_{3|4}}$ et $\mathbb{S}_{Pri_i}$

Chapitre IV

Parcours des ressources fournies

IV.1 L'environnement `Snakemake`

`Snakemake` est un framework permettant de modifier des échelles de données et de créer des fichiers par pipeline. Similaire à `GNU-Make`, cet outil conçu pour la bio-informatique est utilisé ici pour créer et compresser le dataset de Chicago.

Le fichier de configuration, appelé `Snakefile`, comprend une vingtaine d'étapes allant du téléchargement des données jusqu'à la sortie des ensembles $\mathbb{S}_{Synth_i}$ par $\mathcal{G}$. À cet égard il est crucial de comprendre finement son fonctionnement, car il doit être manipulé si l'on veut créer des *Shadow Models (voir VI)*.

Une fois que la commande `Snakemake` est terminée, l'équipe peut commencer à travailler avec un dossier comprenant notamment :

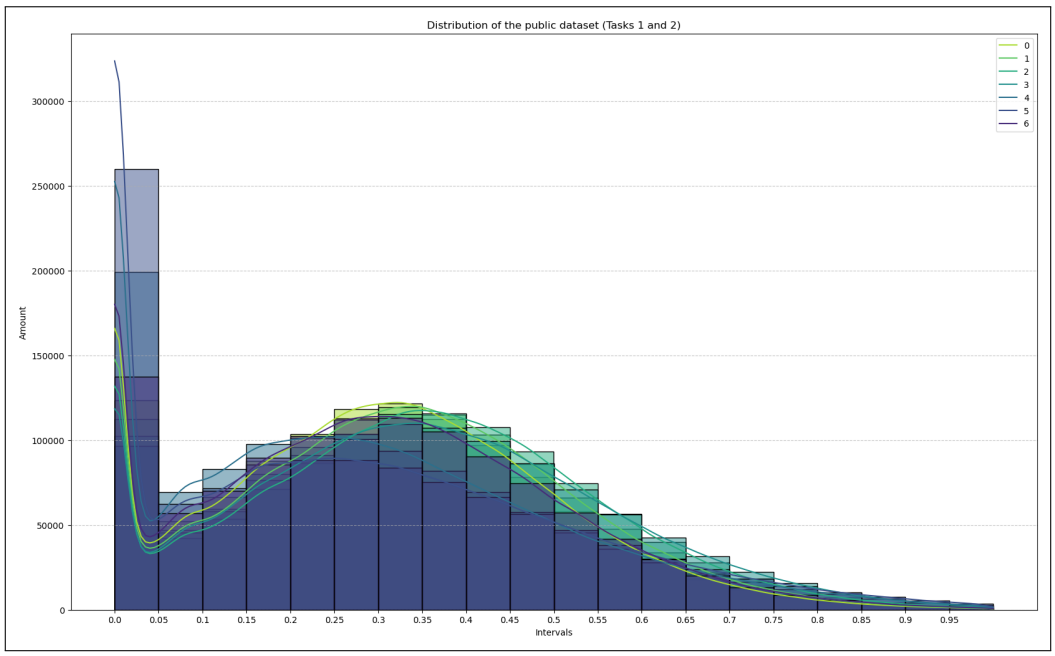
— Le dossier `data` comprenant les datasets $\mathbb{S}_{Pub}$ et $\mathbb{S}_{Synth}$. Les formats utilisés *(`.parquet, .npz, .pckl`)* sont facilement convertissables en dataframes par `pandas`.
— Les scripts nécessaires au fonctionnement de $\mathcal{G}$
— Un certain nombre de fichiers de config `.yaml`

Pour des raisons évidentes, les datasets $\mathbb{S}_{Pri}$ sont hachés.

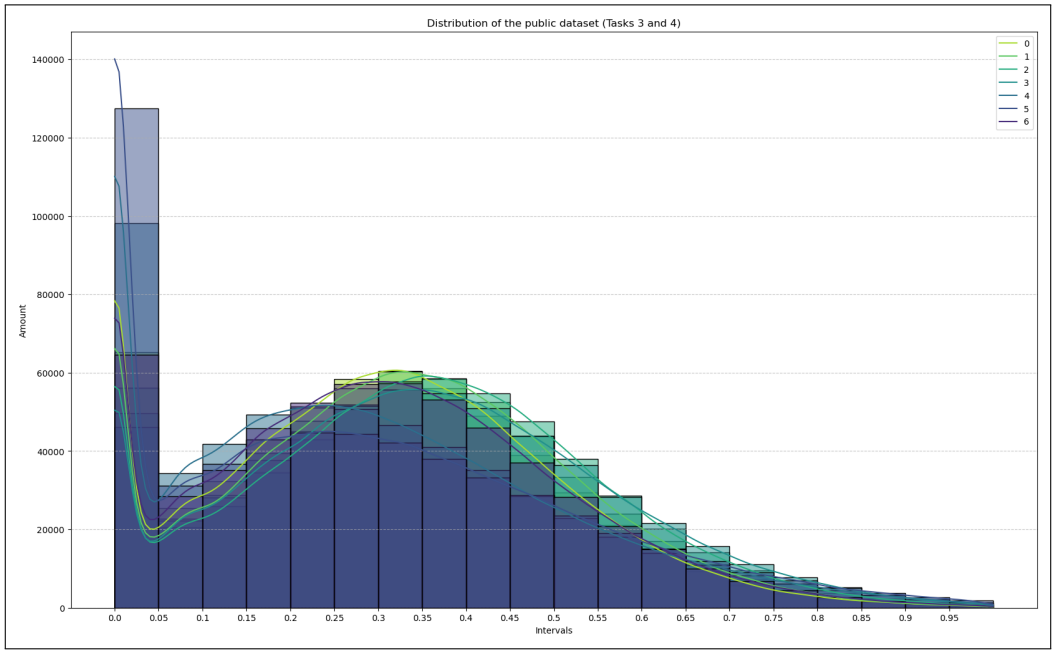
> Le concours étant encore en version *beta*, l'équipe a rencontré de nombreux problèmes lors de l'installation des données fournies. Il est donc préférable à ce stade de développement d'utiliser un environnement Linux et de ne télécharger le dataset initial qu'une fois pour toute l'équipe.

IV.2 Les datasets publics

Comme expliqué précédemment, le dataset prend la forme de 7-tuplets de valeurs entre 0 et 1. On peut alors répartir les valeurs par intervalles pour chaque classe du tuple et évaluer leur quantité sur chacun de ces intervalles. Cette représentation visuelle élémentaire sera notée $\mathfrak{H}(\mathbb{S})$ et est donnée ci-dessous.



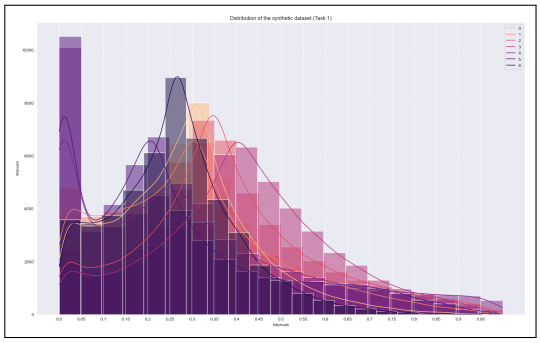
(a) Pour les tâches 1 et 2



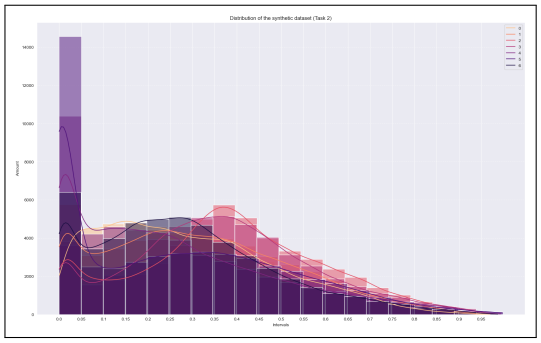
(b) Pour les tâches 3 et 4

FIGURE IV.1 – Distribution des données des datasets $\mathbb{S}_{Pub_i}$, par jour

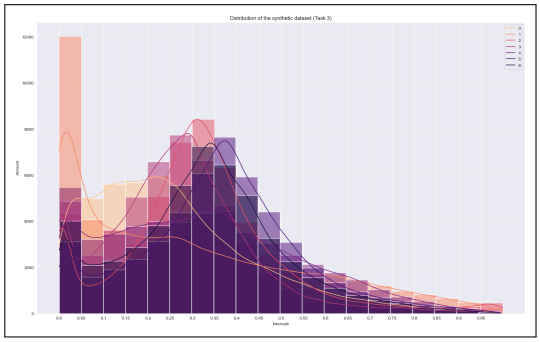
IV.3 Les datasets synthétiques



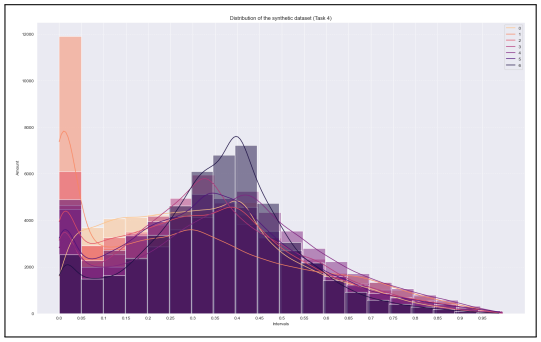
(a) Tâche 1



(b) Tâche 2



(c) Tâche 3



(d) Tâche 4

FIGURE IV.2 – Distribution des données des datasets $\mathbb{S}_{Pub_i}$, par jour.

Chapitre V

DoppelGANger : un générateur de séries temporelles puissant ... mais vulnérable

V.1 Fonctionnement global

Le modèle de la compétition n'inclut pas de métadonnées dans ses entrées et sorties, pour des raisons de simplicité.

V.2 Les hyperparamètres choisis pour la compétition

V.2.1 Batch size

V.2.2 Generator learning

V.2.3 Discriminator

V.2.4 Learning rate

V.2.5 Generator number of hidden layers

V.2.6 Generator hidden layer size

V.2.7 Nombre d'échantillons

V.3 Sensibilité du modèle à une MIA

V.4 Vers une méthodologie d'attaque

Une fois les ressources appropriées par l'équipe, il est possible de déterminer une première approche du problème, en trouvant un compromis entre exploitation des ressources disponibles, adaptation à des néophytes en Machine Learning, temps et efficacité de la méthode de résolution.

L'atout principal dont l'équipe dispose est que G est mis à disposition. Il est alors possible de conduire de nombreuses expérimentations en le prenant comme point de départ.

Bien que les scripts soient intégralement fournis, l'équipe adopte en premier lieu une approche "black-box" par souci de simplicité. La méthode d'attaque peut alors être résumée comme suit :

1. Sélection d'un faux dataset privé $\mathbb{S}'_{Pri}$
2. Génération d'un faux dataset synthétique $\mathbb{S}'_{Synth}$ en entraînant $\mathcal{G}$ sur $\mathbb{S}'_{Pri}$
3. Entraînement d'un algorithme de classification $\mathcal{C}$ sur un dataset d'entraînement $\mathbb{S}'_{Etr}$ dont l'appartenance des lignes à $\mathbb{S}'_{Pri}$ est connue
4. Classification de $\mathbb{C}$ par l'algorithme

Cette méthode utilisant un avatar du générateur initial est appelée **attaque par Shadow Model**. Celle-ci suppose de déterminer intelligemment $\mathbb{S}'_{Pri}$ d'une part, et $\mathcal{C}$ d'autre part. C'est l'objet de la partie suivante qui consacre un chapitre à chacun de ces points, puis résume leur efficacité sur chaque tâche.

Troisième partie

Attaque d'un modèle de Machine Learning par l'utilisation de *Shadow Models*

Chapitre VI

Entraînement de *Shadow Models* sur des ensembles connus

Le Shadow Model n'est efficace que sous la condition d'adopter un comportement similaire à celui du modèle initial.

Ce chapitre a donc pour but la détermination d'un ensemble $\mathbb{S}'_{Pri}$ pour lequel $\mathcal{G}$ génère $\mathbb{S}'_{Synth}$ tel que :

$$\mathbb{S}'_{Pri} \approxeq \mathbb{S}_{Pri} \tag{VI.1}$$

Rappelons que $\mathbb{S}_{Pri}$ n'est pas connu *a priori*.

À cet égard, un certain nombre de points doivent retenir l'attention des attaquants :

1. La taille du dataset $l\left(\mathbb{S}'_{Pri}\right)$
2. La proportion de membres que l'on appelera $r_M\left(\mathbb{S}'_{Pri}\right)$, ainsi la proportion de non-membres est $1 - r_M$
3. L'origine de chaque ligne

De plus et afin de garantir l'équivalence entre les deux modèles, un critère de sélection est adopté pour guider le choix de ces paramètres : **l'équipe fait l'hypothèse forte que pour une paire de modèles, une similarité entre les données de sortie suffit à prouver une similarité entre les données d'entraînement.**. Ainsi :

$$\mathbb{S}'_{Pri} \approxeq \mathbb{S}_{Pri} \Longleftrightarrow \mathbb{S}'_{Synth} \approxeq \mathbb{S}_{Synth} \tag{VI.2}$$

Pour s'assurer que ce critère est respecté, les représentations visuelles $\mathfrak{H}\left(\mathbb{S}'_{Synth}\right)$ et $\mathfrak{H}\left(\mathbb{S}_{Synth}\right)$ sont comparées de façon qualitative.

VI.1 Taille de l'ensemble et proportion de membres r_M

VI.2 Cas d'un entraînement sur plusieurs sous-ensembles

VI.2.1 Avec overlap

On casse l'hypothèse des DS disjoints du DS privé

VI.2.2 Sans overlap

VI.3 Le problème du surapprentissage

Chapitre VII

Méthodes de classification des données générées

VII.1

VII.2 Régression logistique

VII.3 Bayésien naïf

VII.4 Recherche des plus proches voisins (KNN)

Chapitre VIII

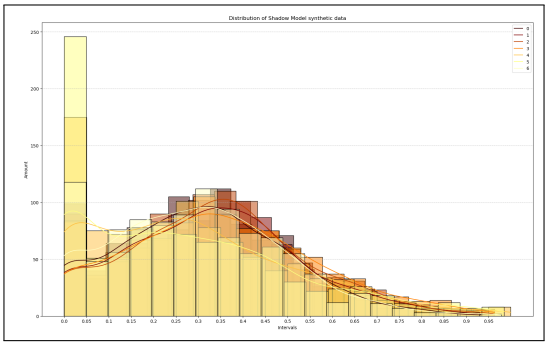
Synthèse des résultats

VIII.1 Tâche 1

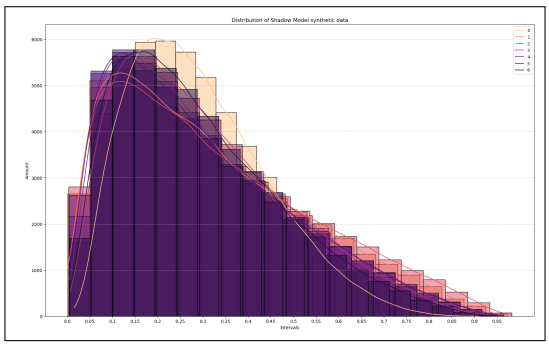
VIII.2 Tâche 2

On choisit d'aborder cette tâche en premier en raison de la petite taille de son dataset privé : $l\left(\mathbb{S}_{Priv_2}\right) = 1000$. En effet, on peut supposer que la variation d'un paramètre aura une plus forte sensibilité sur un petit ensemble.

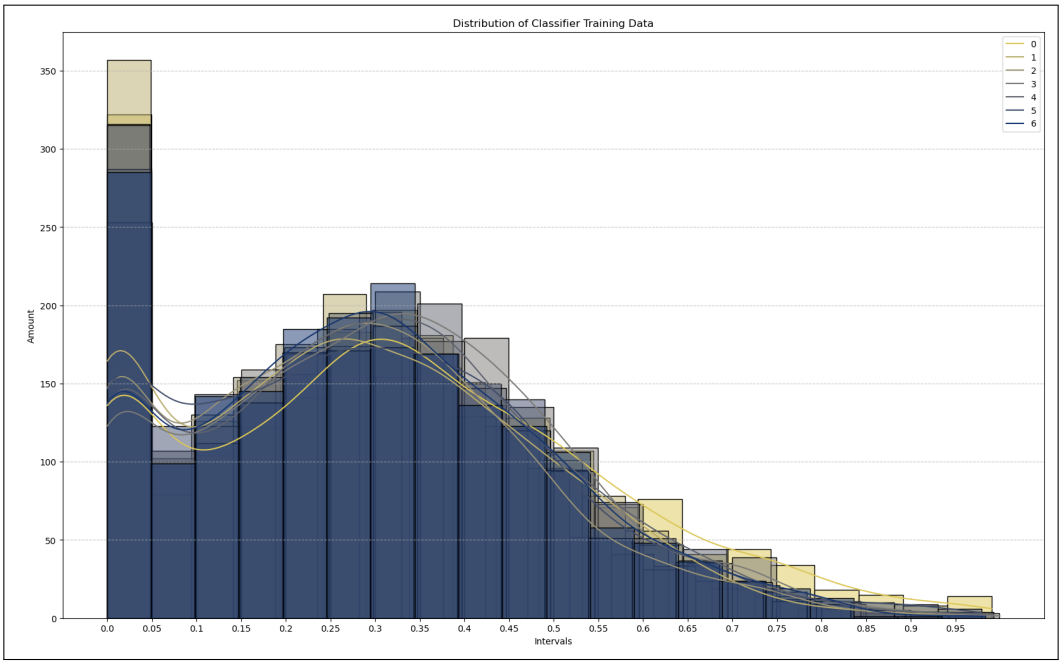
On choisit $\mathbb{S}'_{Priv_2} = \mathbb{S}_{Synth_2}$. Du côté du classifieur, $\mathbb{S}'_{Etr_2}$ est composé de $\mathbb{S}'_{Priv_2}$ et des lignes 100 à 1100 de $\mathbb{S}_{Pub_{1|2]}}$.



(a) $\mathbb{S}'_{Priv_2}$



(b) $\mathbb{S}'_{Synth_2}$



(c) $\mathbb{S}'_{Etr_2}$

FIGURE VIII.1 – Distribution des données du Shadow Model

Si $\mathbb{S}'_{Priv_2}$ et $\mathbb{S}'_{Etr_2}$, semblent correspondre, on remarque immédiatement que les données générées ont une allure tout à fait différente de celle observées jusqu'ici. La pertinence de l'étude semble alors compromise.

Le classifieur obtient une précision de 67% pendant son entraînement. De même, ce score est encourageant mais largement perfectible. Par curiosité, on peut quand-même observer ce qu'il indique lors de la classification de $\mathbb{C}_2$:

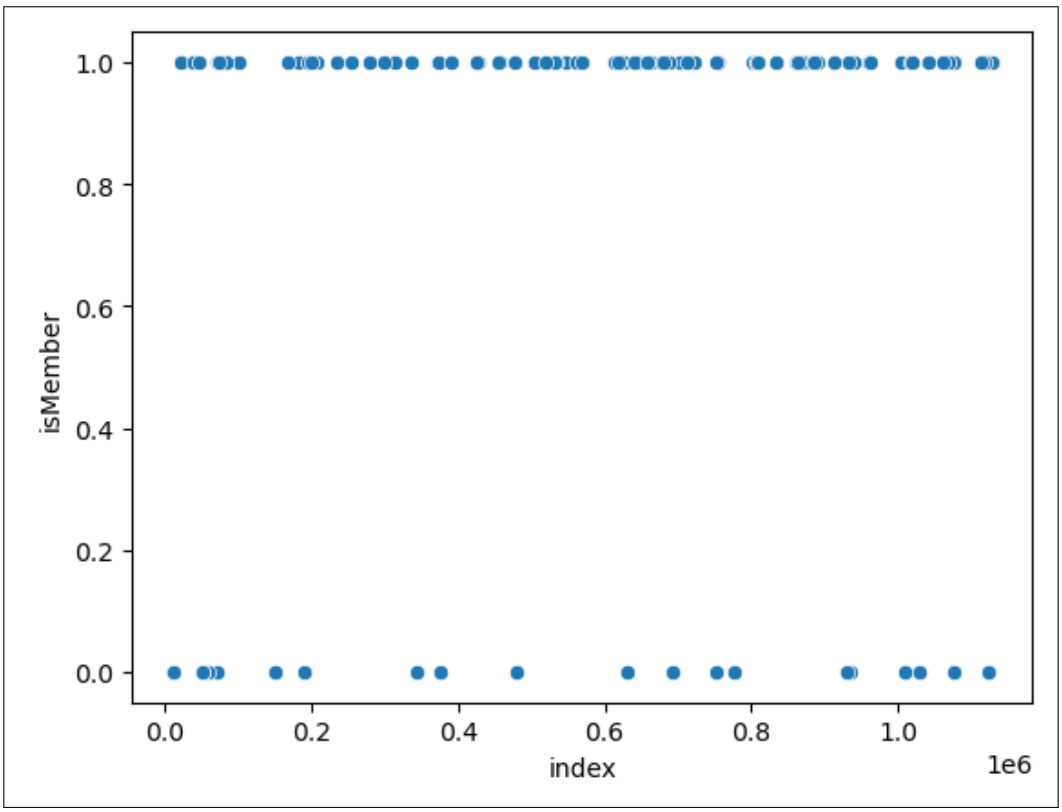


FIGURE VIII.2 – Prédiction par indice de l'appartenance des lignes $\mathbb{C}_2$ à $\mathbb{S}_{Priv_2}$

VIII.3 Tâche 3

VIII.4 Tâche 4

Conclusion

Quatrième partie

Annexes

Annexe 1 : Programmes conçus par l'équipe

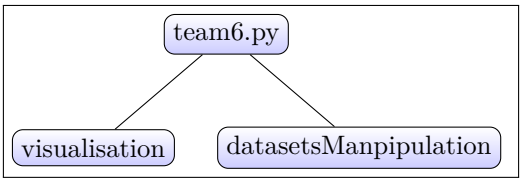


FIGURE VIII.3 – [WIP] Organisation des fichiers `Python` créés

```
    import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import sklearn

# Constants for the whole competition
NUMBER_OF_COLUMNS_DATASET = 7

# Graphic conventions for our figures
palette_PublicDS = sns.color_palette("viridis_r", n_colors=NUMBER_OF_COLUMNS_DATASET)
palette_PrivateDS = sns.color_palette("magma_r", n_colors=NUMBER_OF_COLUMNS_DATASET)

# Making dataframes from files so that we load them only one time

publicData_Tasks12 = pd.read_parquet('../publicData/publicDatasetTask1-2.parquet')
publicData_Tasks34 = pd.read_parquet('../publicData/publicDatasetTask3-4.parquet')
targets_Task1 = pd.read_csv('../publicData/targetsTask1.csv')
targets_Task2 = pd.read_csv('../publicData/targetsTask2.csv')
targets_Task3 = pd.read_csv('../publicData/targetsTask3.csv')
targets_Task4 = pd.read_csv('../publicData/targetsTask4.csv')


def npzPrivateToDataframe(npz_FilePath):
    npz = np.load(npz_FilePath)
    features_array = npz["data_feature"]
    # Step 4: Reshape the array to remove the extra dimension
    if features_array.ndim == 3 and features_array.shape[2] == 1:
        features_array = features_array.reshape(features_array.shape[0], features_array.shape
    [1])
    dataFrame = pd.DataFrame(features_array)
    return dataFrame

syntheticData_Task1 = npzPrivateToDataframe("../publicData/syntheticTask1.npz")
syntheticData_Task2 = npzPrivateToDataframe('../publicData/syntheticTask2.npz')
syntheticData_Task3 = npzPrivateToDataframe('../publicData/syntheticTask3.npz')
syntheticData_Task4 = npzPrivateToDataframe('../publicData/syntheticTask4.npz')

# Functions to make our train and test sets

def makeSubsetFromDataset(dataFrame, subsetStartIndex, subsetEndIndex_excluded):
    # Make small datasets from the original dataset
    return dataFrame.iloc[subsetStartIndex:subsetEndIndex_excluded].copy()

def addMembershipToSubset(subset, isMember):
    newSubset = subset.copy()
    if isMember == 0 or isMember == 1:
        subset["isMember"] = isMember
        return newSubset
    else:
        print("Second parameter must be 0 or 1. Your dataset hasn't changed")
        return newSubset
```

Listing VIII.1 – Intégration des données du problème et manipulation des dataset

```
import os
from data.teamExperiments.team6 import npzPrivateToDataframe, addMembershipToSubset,
    makeSubsetFromDataset

os.chdir('/Users/thom/Personnel/Scolaire/MIA/snake2-beta-insa-main/data/teamExperiments')
print(os.getcwd())

from team6 import*
from visualisation import *

members = npzPrivateToDataframe('falseTrain_Task2_1.npz')
addMembershipToSubset(members, 1)
members

nonMembers = makeSubsetFromDataset(publicData_Tasks12, 100, 1100)
addMembershipToSubset(nonMembers, 0)
nonMembers

trainingSet = pd.concat([members, nonMembers])
trainingSet

x = trainingSet.drop(['isMember'], axis=1)
y = trainingSet['isMember']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()
classifier.fit(x_train, y_train)

print("training test score: %f" % classifier.score(x_train, y_train))
print("test score: %f" % classifier.score(x_test, y_test))

toTest = targets_Task2.drop(['Unnamed: 0', 'index'],axis=1)
myPred = classifier.predict(toTest.values)
myPred

myPred = pd.DataFrame(myPred, columns=['isMember'])
myPredVisu = pd.concat([targets_Task2, myPred], axis=1)

myPredVisu.keys()
myHist = myPredVisu[['index', 'isMember']]
sns.histplot(myHist, x='isMember')
```

Listing VIII.2 – Classification par bayésien naïf

```python
import team6
from team6 import *

figureLength = 20
figureWidth = 12

def plotDistributionByDay(dataframe, precision, palette, title):
    myLabels = []

    plt.figure(figsize=(figureLength, figureWidth))
    for j in range(NUMBER_OF_COLUMNS_DATASET):
        sns.histplot(
            dataframe[j],
            bins=precision,
            kde=True,
            color = palette[j],
        )
        myLabels.append(j)

    plt.title(title)
    plt.xlabel('Intervals')
    plt.ylabel('Amount')
    plt.xticks(
        ticks=[i/precision for i in range(precision)],
        labels=[f'{i/precision}' for i in range(precision)]
    )
    plt.grid(axis='y', linestyle='--', alpha=0.7)

    plt.legend(
        labels=myLabels,
    )

    # Show the plot
    plt.show()
```

Listing VIII.3 – Visualisation par histogrammes de la répartition des données

Annexe 2 : Retour d'expérience et chronologie du projet

Annexe 3 : Framework utilisé

Cinquième partie

Bibliographie

Fondamentaux de mathématiques et de programmation

[5] Chloé-Agathe Azencott. *Introduction au Machine Learning*. (2nd). InfoSup. Dunod, fév. 2022.

[9] Matt Harrison. *Machine Learning - Les Fondamentaux. Exploiter des données structurées en Python*. O'Reilly, 2020.

[10] Benjamin JOURDAIN. *Probabilités et statistiques pour l'ingénieur*. Jan. 2018.

[13] *Machine Learning*. Page Wikipedia du Machine Learning. Nov. 2024. URL : `https://en.wikipedia.org/wiki/Machine_learning`.

Sur le Machine Learning Antagoniste *(Adversarial Machine Learning)*

[1] *Adversarial Machine Learning*. Page Wikipedia de l'Adversarial Machine Learning. Nov. 2024. URL : https://en.wikipedia.org/wiki/Adversarial_machine_learning#Adversarial_attacks_and_training_in_linear_models.

[2] Tristan Allard et Mathias Bernard. « Snakes Strikes Back ». In : (oct. 2024).

[4] Author. *Membership inference attacks from first principles*. How published. Some note. Month Year. URL : https://www.youtube.com/watch?v=1CNxfhMlk-A.

[8] *Generative adversarial network*. Page Wikipedia du modèle GAN. Nov. 2024. URL : https://en.wikipedia.org/wiki/Generative_adversarial_network.

[12] Zinan Lin et al. « Using GANs for Sharing Networked Time Series Data : Challenges, Initial Promise, and Open Questions ». In : (jan. 2021). Présentation du modèle DoppelGANger. URL : https://arxiv.org/abs/1909.13403.

[17] Reza Shokri. *Membership Inference Attacks against Machine Learning Models*. Vidéo de vulgarisation du papier du même nom. Mai 2017. URL : https://www.youtube.com/watch?v=rDm1n2gceJY&t=53s.

[18] Reza Shokri et al. « Membership Inference Attacks Against Machine Learning Models ». In : ().

Ressources diverses pour le Machine Learning

[3] Tatev Aslanyan. *Machine Learning in 2024 – Beginner's Course*. Fév. 2024. URL : https://www.youtube.com/watch?v=bmmQA8A-yUA&t=1769s.

[6] *Classification naïve bayésienne*. Page Wikipedia du modèle bayésien naïf. Août 2024. URL : https://fr.wikipedia.org/wiki/Classification_na%C3%AFve_bay%C3%A9sienne.

[7] *Gaussian Naive Bayes Explained With Scikit-Learn*. Tutoriel pour construire un classifieur bayésien naïf. Nov. 2023. URL : https://builtin.com/artificial-intelligence/gaussian-naive-bayes.

[11] Learndataa. *81 Scikit-learn 78 Supervised Learning 56 Naive Bayes classifiers*. Youtube. 2021. URL : https://www.youtube.com/watch?v=9Tmnr4L5ZOQ.

[14] MarkovML. *Comparing and Evaluating Datasets : A Simplified Guide*. 24 nov. 2024. URL : https://www.markovml.com/blog/compare-datasets.

[15] Boris Meinardus. *How I'd learn ML in 2024 (if I could start over)*. Youtube. 2024. URL : https://www.youtube.com/watch?v=gUmagAluXpk.

[16] *Overfitting*. Page Wikipedia de l'Overfitting. Nov. 2024. URL : https://en.wikipedia.org/wiki/Overfitting#Machine_learning.

[19] *Training, validation, and test data sets*. Page Wikipedia rappelant la différence entre les datasets d'entraînement et de test. Nov. 2024. URL : https://en.wikipedia.org/wiki/Training,_validation,_and_test_data_sets.