ISIFC3 Microrobotique, e-santé : Crypto.
oct. 2024, Devoir en Temps Libre.

J.-F. COUCHOT

24 octobre 2024

1 Introduction

Ce travail est à réaliser par groupe de trois ou quatre personnes. Les réponses aux questions et le code sont à placer dans un ou plusieurs jupyter notebooks. Ces notebooks contiendront une entête avec le nom et le prénom de chaque membre du groupe. Tout sera consigné dans une archive zip. L'ensemble est à envoyer par mail avant le 08 nov 2024 à 18h à mon adresse mail.

Vous êtes en charge du développement d'un dispositif D_1 qui mémorise dans un jeu de données :

— des résultats d'analyse sanguine ;

— des données générales comme l'âge, le genre...

Par exemple, ce jeu de données construites pourrait ressembler à celui accessible sur le site de Kaggle, Patient Treatment Classification.

Vous êtes aussi en charge d'un dispositif d'apprentissage D_2 qui va apprendre sur ces données et émettre un avis éclairé sur la perspective de réaliser les soins associés en ambulatoire ou avec une prise en charge à l'hôpital plus lourde.

2 Envoi de données de D_1 vers D_2

Entre le dispositif D_1 et D_2, des données vont transiter.

1. Que doit être mis en place pour éviter que quiconque puisse lire des données médicales entre l'envoi de celles-ci par D_1 et la réception par D_2.
2. Quel(s) type(s) d'algorithme(s) va-t-on déployer pour réaliser ce partage ? Justifier.
3. Donner un code complet qui prend en entrée les adresses de D_1, de D_2 et les données de D_1 à partager et qui réalise ce partage.

3 Apprentissage sur des données originales par D_2

1. Le dispositif d'apprentissage D_2 travaille sur des données médicales et a pour objectif d'assister à la création d'un diagnostic. C'est donc un dispositif critique et d'après l'IA-act, ses résultats doivent être motivés. Il apparaît comme judicieux de mettre en place alors un arbre de décision pour assister au diagnostic. Justifier ce choix.
2. A titre d'exemple, on donne le code de la figure 1 de décision à partir d'un arbre sur le jeu de données PIMA vu en cours. Expliquer ce code dans les détail. Expliquer l'arbre construit, ce qu'il y a dans chacun des noeuds.
3. Mettre en place un tel algorithme sur le jeu de données de Kaggle, présenté en introduction. Faire varier la profondeur de l'arbre. Analyser l'arbre construit.

```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

df = pd.read_csv('diabetes.csv')
df['Outcome'].replace({'NO': 0, 'YES': 1},inplace=True)
X = df.iloc[:, 0:8]
y = df.iloc[:, 8]

for column in ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin']:
    X[column] = X[column].replace(0, np.NaN)
    mean = int(X[column].mean(skipna=True))
    X[column] = X[column].replace(np.NaN, mean)
for column in ['BMI','DiabetesPedigreeFunction']:
    X[column] = X[column].replace(0, np.NaN)
    mean = X[column].mean(skipna=True)
    X[column] = X[column].replace(np.NaN, mean)
```

(a) Prétraitement

```python
import pretraitement
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score,
from sklearn.model_selection import cross_val_score, StratifiedShuffleSplit
from sklearn.tree import DecisionTreeClassifier, plot_tree
import numpy as np

X,y = pretraitement.X,pretraitement.y

clf = DecisionTreeClassifier(criterion="entropy",max_depth=5)
cv = StratifiedShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
scores = cross_val_score(clf, X, y, cv=cv, scoring='accuracy')

mean_score = np.mean(scores)
std_deviation = np.std(scores)
print(f"Scores de validation croisée: {scores}")
print(f"Précision moyenne: {mean_score:.4f}")
print(f"Écart type des précisions: {std_deviation:.4f}")

clf = DecisionTreeClassifier(criterion="entropy",max_depth=5)
clf.fit(X.values, y)
print(np.array(clf.predict([[1, 184, 84, 33, 0, 35.5, 0.355, 41]])))

plt.figure()
plot_tree(clf,filled=True)
plt.title("Decision tree trained on all the pima features")
plt.savefig("dt.svg")
```

(b) Classification

FIGURE 1 – Décision par arbre appliquée au dataset pima

4 Apprentissage sur des données nettoyées

1. Dans cette question, vous allez mettre en place du 10-anonymat sur ces données, en considérant que sont quasi-identifiants les attributs âge et Sexe. Construire ce jeu de données 10 annonymes.
2. Mettre en place un apprentissage sur le dataset ainsi construit. Que pensez-vous d'un tel apprentissage en termes de précision ?
3. Une seconde manière de nettoyer ces données consiste à appliquer sur celle-ci de la confidentialité différentielle locale selon le mécanisme laplacien, sauf pour le sexe et la source ou l'on prendra une réponse randomisée. Ceci pourrait être réalisé sur le dataset PIMA au moyen du code donné à la figure 2. Expliquer ce code en détail
4. Mettre en place un tel algorithme d'assainissement en considérant un budget de fuite $\epsilon = 10$ pour chaque enregistrement que l'on divisera en 11, puisqu'il y a 11 attributs dans votre dataset.
5. Mettre en place un apprentissage sur le dataset ainsi construit. Que pensez-vous d'un tel apprentissage en termes de précision ? En terme de protection de la vie privée ?

```python
import pandas as pd
import numpy as np
import math as m

df = pd.read_csv('diabetes.csv')
df['Outcome'].replace({'NO': 0, 'YES': 1},inplace=True)


for column in ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin']:
    df[column] = df[column].replace(0, np.NaN)
    mean = int(df[column].mean(skipna=True))
    df[column] = df[column].replace(np.NaN, mean)
for column in ['BMI','DiabetesPedigreeFunction']:
    df[column] = df[column].replace(0, np.NaN)
    mean = df[column].mean(skipna=True)
    df[column] = df[column].replace(np.NaN, mean)

naturalAttributes = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin','Age']
floatPositiveAttributes = ['BMI', 'DiabetesPedigreeFunction']
booleanAttributes = ['Outcome']

def ranges(listOfAtt,df):
    d={}
    description = df.describe()
    for att in listOfAtt:
        mi = description[att]['min']
        mx = description[att]['max']
        d[att] = mx-mi
    return d

rangeOfAttribute = ranges(naturalAttributes + floatPositiveAttributes,df)

def addLaplaceNoise(rangeOfAttribute,val,isNatural=False,eps=1):
    r = max(0,val + np.random.laplace(0,rangeOfAttribute/eps))
    return int(round(r)) if isNatural else r

def generateRandomResponse(val,eps=1):
    p = (m.exp(eps)/(1+m.exp(eps)))
    if np.random.rand() < p :
        return val
    else :
        return 1- val

eps = 10
nb_att= df.shape[1]

for att in naturalAttributes:
    df[att] = df[att].map(lambda a: addLaplaceNoise(rangeOfAttribute[att],a,True,eps/nb_att))

for att in floatPositiveAttributes:
    df[att] = df[att].map(lambda a: addLaplaceNoise(rangeOfAttribute[att],a,False,eps/nb_att))
df['Outcome'] = df['Outcome'].map(lambda a: generateRandomResponse(a,eps/nb_att))

df.to_csv('diabetes_sanitized.csv',index=False)
```

FIGURE 2 – Application de la confidentialité différentielle locale sur le dataset pima