Online Signature Verification Challenge

Adnane Mehdaoui

Abdelghaffour Mouhsine

Yendoubouam Alexandre Lalle

Karim jtit

youssef aderdar

Rachid Baiou

Entrée [1]:

```
import numpy as np
import os
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing
from sklearn import svm
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
```

Extraction des donnees

Entrée [2]:

```
#Creation du matrice qui contient 7 column
def Create_Matrix(file):
    Rows_Affected=0
    Total_Rows=0
    Matrix=None
    with open(file,'r') as data:
        rows=data.readlines()
        for row in rows:
            if Rows_Affected==0:
                Total_Rows=int(row)
                Matrix=np.zeros((Total_Rows,7))
                Rows_Affected+=1
            else:
                line=row.strip().split(" ")
                if len(line)==7:
                    Matrix[Rows_Affected-1,0]=float(line[0])
                    Matrix[Rows_Affected-1,1]=float(line[1])
                    Matrix[Rows_Affected-1,2]=float(line[2])
                    Matrix[Rows_Affected-1,3]=float(line[3])
                    Matrix[Rows_Affected-1,4]=float(line[4])
                    Matrix[Rows_Affected-1,5]=float(line[5])
                    Matrix[Rows_Affected-1,6]=float(line[6])
                    Rows_Affected+=1
        Rows_Affected-=1
        while((Total_Rows-Rows_Affected)!=0):
            Matrix=np.delete(Matrix,Total_Rows-1, 0)
            Total_Rows=Total_Rows-1
    return Matrix
```

Creation de la Methode DTW

Entrée [3]:

```
def Distance_X_Y(pt1,pt2):
    return np.sqrt((pt1[0]-pt2[0])**2+(pt1[1]-pt2[1])**2)

def Distance_Other_Fea(pt1,pt2):
    return np.sqrt((pt1-pt2)**2)

def Extra_Point(Mat):
    Points=np.zeros((Mat.shape[0],2))
    for i in range(Mat.shape[0]):
        Points[i,0]=Mat[i,0]
        Points[i,1]=Mat[i,1]
    return Points

def Extra_Feat(Mat,dim):
    if dim>=2 and dim<7:
        Feat=np.zeros((Mat.shape[0],1))
        for i in range(Mat.shape[0]):
            Feat[i,0]=Mat[i,dim]
        return Feat
    else:
        return None

```

Entrée [4]:

```
def DTW_X_AND_Y(Mat_Ref,Mat_Requ):
       X_Y_Ref=Extra_Point(Mat_Ref)
       X_Y_Requ=Extra_Point(Mat_Requ)
       n =X_Y_Ref.shape[0]
       m =X_Y_Requ.shape[0]
       cost_matrix = np.full((n+1, m+1), np.inf)
       cost_matrix[0, 0] = 0
       for i in range(1, n+1):
         for j in range(1, m+1):
             cost = Distance_X_Y(X_Y_Ref[i-1], X_Y_Requ[j-1])
             cost_matrix[i, j] = cost + min(cost_matrix[i-1, j], cost_matrix[i, j-1]

       i = n
       j = m
       path = [(i, j)]
       while i > 1 and j > 1:
           if cost_matrix[i-1, j] < cost_matrix[i-1, j-1] and cost_matrix[i-1, j] <
              i -= 1
           elif cost_matrix[i, j-1] < cost_matrix[i-1, j-1] and cost_matrix[i, j-1]
              j -= 1
           else:
              i -= 1
              j -= 1
           path.append((i, j))
       path.append((1, 1))

       return cost_matrix[n, m] / len(path)


```

Entrée [5]:

```
def DTW_Other_Features(Mat_Ref,Mat_Requ):
    Coe={}
    dim=2
    while(dim<7):
        Feat_Ref=Extra_Feat(Mat_Ref,dim)
        Feat_Req=Extra_Feat(Mat_Requ,dim)
        n = Feat_Ref.shape[0]
        m = Feat_Req.shape[0]
        cost_matrix = np.full((n+1, m+1), np.inf)
        cost_matrix[0, 0] = 0
        for i in range(1, n+1):
         for j in range(1, m+1):
             cost = Distance_Other_Fea(Feat_Ref[i-1], Feat_Req[j-1])
             cost_matrix[i, j] = cost + min(cost_matrix[i-1, j], cost_matrix[i, j-1]

        i = n
        j = m
        path = [(i, j)]
        while i > 1 and j > 1:
            if cost_matrix[i-1, j] < cost_matrix[i-1, j-1] and cost_matrix[i-1, j] <
              i -= 1
            elif cost_matrix[i, j-1] < cost_matrix[i-1, j-1] and cost_matrix[i, j-1]
              j -= 1
            else:
              i -= 1
              j -= 1
            path.append((i, j))
        path.append((1, 1))
        if dim==2:
            Coe['time stamp']=cost_matrix[n, m] / len(path)
        elif dim==3:
            Coe['button status']=cost_matrix[n, m] / len(path)
        elif dim==4:
            Coe['azimuth']=cost_matrix[n, m] / len(path)
        elif dim==5:
            Coe['altitude']=cost_matrix[n, m] / len(path)
        elif dim==6:
            Coe['pressure']=cost_matrix[n, m] / len(path)
        dim+=1
    return Coe

def DTW_ALL_Features(SigRef,SigRequ):
    Mat_Ref=Create_Matrix(SigRef)
    Mat_Requ=Create_Matrix(SigRequ)
    dict=DTW_Other_Features(Mat_Ref,Mat_Requ)
    dict['X_Y']=DTW_X_AND_Y(Mat_Ref,Mat_Requ)
    return dict
```

CREATION DU DATAFRAME

on a calculer DTW entre deux signatures du meme utilisateur chaque utilisateur possède un dossier qui contient leur signatures

Dans un premier temps en calcule DTW entre 2 signatures pour toutes les signatures de chaque utilisateur

on refait la meme action pour tous les utilisateurs et on donne a la colonne Similary la valeur 1

Entrée [6]:

```
df = pd.DataFrame(columns=['X_Y', 'time stamp','button status','azimuth','altitude',
#SIMILAR
for i in range(1,34):
dir='online_signature_verification/data/Task1/New_Data/User'+str(i)
for k in range(len(os.listdir(dir))):
   file1=os.listdir(dir)[k]
   file1=dir+'/'+file1
   for j in range(k+1,len(os.listdir(dir))):
      file2=os.listdir(dir)[j]
      file2=dir+'/'+file2
      dict=DTW_ALL_Features(file1,file2)
      dict['Similary']=True
      df.loc[df.shape[0]]=[dict['X_Y'],dict['time stamp'],dict['button status'],dict
#NONE
for i in range(1,32):
   dir1='online_signature_verification/data/Task1/New_Data/User'+str(i)
   for k in range(len(os.listdir(dir1))-3):
       file1=dir+'/'+os.listdir(dir)[k]
       dir2='online_signature_verification/data/Task1/New_Data/User'+str(i+1)
       for j in range(len(os.listdir(dir2))-3):
          file2=dir2+'/'+os.listdir(dir2)[j]
          dict=DTW_ALL_Features(file1,file2)
          dict['Similary']=False
          df.loc[df.shape[0]]=[dict['X_Y'],dict['time stamp'],dict['button status'],

df
```

Entrée [7]:

```
df.to_csv('online_signature_verification/data/Task1/New_Data.csv',index=False, heade
df=pd.read_csv('online_signature_verification/data/Task1/New_Data.csv')
df=df.replace({False:0,True:1})
df.head()
```

Out[7]:

	X_Y	time stamp	button status	azimuth	altitude	pressure	Similary
0	770.575241	2299.813953	0.00	31.509434	13.951613	60.000000	1
1	614.169296	4173.857143	0.00	35.445545	38.931298	60.888889	1
2	345.186713	6476.132530	0.00	102.149533	29.701493	63.509615	1
3	826.504983	14148.638298	0.02	36.967213	87.181818	62.816794	1
4	345.188928	1872.219780	0.00	11.698113	33.816794	40.369748	1

A cette etape en implémente 5 modele et en fait une élection a chaque teste pour voir la majeur réponse

Entrée [8]:

```

Featu=df.drop(['Similary'],axis=1)
Label=df['Similary']
Model_Line=LogisticRegression().fit(Featu, Label)
Model_Arbre=tree.DecisionTreeClassifier().fit(Featu,Label)
Model_Fore=RandomForestClassifier(n_estimators=50).fit(Featu,Label)
Model_KNN=KNeighborsClassifier().fit(Featu,Label)
Model_SVM=svm.SVR().fit(Featu,Label)
Model_Used=[Model_Line,Model_Arbre,Model_Fore,Model_KNN,Model_SVM]


```

Entrée [9]:

```
def Voting_Model(dict,Tab_Model):
    Vote_Oui=0
    Vote_Non=0
    for Model in Tab_Model:
        Predct=Model.predict([[dict['X_Y'],dict['time stamp'],dict['button status'],
        if Predct==True:
            Vote_Oui+=1
        else:
            Vote_Non+=1
    if Vote_Non<Vote_Oui:
        return True
    else:
        return False

def CompareSing(SigRef,SingReq,Model_Used):
    dict=DTW_ALL_Features(SigRef,SingReq)
    return Voting_Model(dict,Model_Used)

```

Entrée [10]:

```
CompareSing('online_signature_verification/data/Task1/New_Data/User1/U1S1.TXT',
            'online_signature_verification/data/Task1/New_Data/User1/U1S2.TXT',Model_

```

```
C:\Users\DATA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_
qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\ba
se.py:439: UserWarning: X does not have valid feature names, but LogisticR
egression was fitted with feature names
  warnings.warn(
C:\Users\DATA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_
qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\ba
se.py:439: UserWarning: X does not have valid feature names, but DecisionT
reeClassifier was fitted with feature names
  warnings.warn(
C:\Users\DATA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_
qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\ba
se.py:439: UserWarning: X does not have valid feature names, but RandomFor
estClassifier was fitted with feature names
  warnings.warn(
C:\Users\DATA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_
qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\ba
se.py:439: UserWarning: X does not have valid feature names, but KNeighbor
sClassifier was fitted with feature names
  warnings.warn(
C:\Users\DATA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_
qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\ba
se.py:439: UserWarning: X does not have valid feature names, but SVR was f
itted with feature names
  warnings.warn(
```

Out[10]:

```
True
```

CREATION DU FICHIER CSV

Entrée [21]:

```
dir='online_signature_verification/data/Task1/Test'
df = pd.DataFrame(columns=['Sing_FileName1','Sing_FileName2','Similary'])

for i in range(len(os.listdir(dir))-1):
    file1=dir+'/'+os.listdir(dir)[i]
    file2=dir+'/'+os.listdir(dir)[i+1]
    result=CompareSing(file1,file2,Model_Used)
    df.loc[df.shape[0]]=[file1,file2,result]

df

```

```
C:\Users\DATA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.1
0_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklear
n\base.py:439: UserWarning: X does not have valid feature names, but Log
isticRegression was fitted with feature names
  warnings.warn(
C:\Users\DATA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.1
0_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklear
n\base.py:439: UserWarning: X does not have valid feature names, but Dec
isionTreeClassifier was fitted with feature names
  warnings.warn(
C:\Users\DATA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.1
0_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklear
n\base.py:439: UserWarning: X does not have valid feature names, but Ran
domForestClassifier was fitted with feature names
  warnings.warn(
C:\Users\DATA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.1
0_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklear
n\base.py:439: UserWarning: X does not have valid feature names, but KNe
ighborsClassifier was fitted with feature names
```

ACCURACY

Entrée [33]:

```
df.to_csv('online_signature_verification/data/Task1/Result.csv',index=False, header=
df.shape
df=df.replace({False:0,True:1})
# Nbr Impaire False
# Nbr Paire True
Accu=0
result=0
for i in range(df.shape[0]):
    if i%2==0:
        result=1
        if df['Similary'][i]==result:
            Accu+=1
    else:
        result=0
        if df['Similary'][i]==result:
            Accu+=1

Accu/df.shape[0]
```

Out[33]:

```
0.8923076923076924
```

CONFUSION MATRIX

Entrée [34]:

```
Matrix_Conf=np.zeros((2,2))
result=0
for i in range(df.shape[0]):
    if i%2==0:
        result=1
    else:
        result=0
    if result==0 and df['Similary'][i]==0:
        Matrix_Conf[0,0]+=1
    elif result==1 and df['Similary'][i]==0:
        Matrix_Conf[0,1]+=1
    elif result==0 and df['Similary'][i]==1:
        Matrix_Conf[1,0]+=1
    else:
        Matrix_Conf[1,1]+=1

Matrix_Conf

```

Out[34]:

```
array([[29.,  4.],
       [ 3., 29.]])
```

Type *Markdown* and LaTeX: α^2

Entrée [ ]:

```
1
```

Entrée [ ]:

```
1
```