كلية العلوم و التقنيات فاس
ⵜⴰⵖⵚⵍⵓⵜ ⵏ ⵜⵙⴰⵏⴰⵢⵉⵏ ⴷ ⵜⵓⵙⵙⵙⵉⵏ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵡⴰⵍⵜ ⵓⵙⵙ ⵎⵓⵃⴰⵎⴷ ⵏ ⵄⴰⴱⴷⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

TP1: SVM

Réalisé par:

- El Ghoul Abdessalam

Master – S.I.R
2021/2022

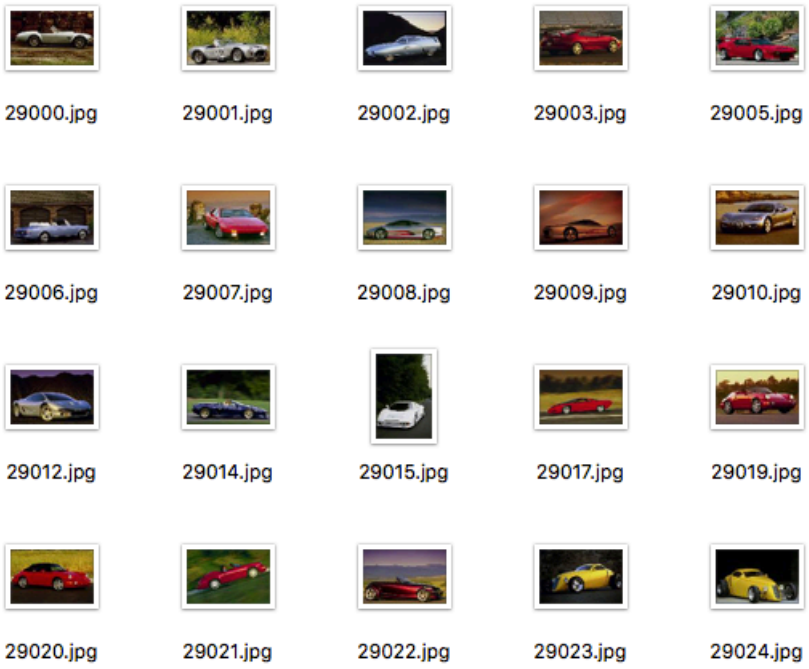
TP 1 : SVM

- ## Introduction :

  Les SVMs sont une famille d'*algorithmes* d'*apprentissage automatique* qui permettent de résoudre des problèmes tant de *classification* que de *régression* ou de détection d'anomalie. Ils sont connus pour leurs solides garanties théoriques, leur grande flexibilité ainsi que leur simplicité d'utilisation même sans grande connaissance de *data mining*.

- ## Data set utiliser :

La base de donnée utiliser est une base de donnée contient des images des voitures et des bateaux



1. Importation de librairies :

```
#importation des bibliothèque
import numpy as np
import cv2
import imutils
import csv
import argparse
import glob
import math
import pandas as pd
import matplotlib.pyplot as plt
```

2. Les Fonction utiliser dans le prétraitement :

```
#Variables
bins = (8, 12, 3)
#colorDiscripteur
def describe(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    features = []
    (h, w) = image.shape[:2]
    (cX, cY) = (int(w * 0.5), int(h * 0.5))
    segments = [(0, cX, 0, cY), (cX, w, 0, cY), (cX, w, cY, h),
        (0, cX, cY, h)]
    (axesX, axesY) = (int(w * 0.75) // 2, int(h * 0.75) // 2)
    ellipMask = np.zeros(image.shape[:2], dtype = "uint8")
    cv2.ellipse(ellipMask, (cX, cY), (axesX, axesY), 0, 0, 360, 255, -1)
    for (startX, endX, startY, endY) in segments:
        cornerMask = np.zeros(image.shape[:2], dtype = "uint8")
        cv2.rectangle(cornerMask, (startX, startY), (endX, endY), 255, -1)
        cornerMask = cv2.subtract(cornerMask, ellipMask)
        hist = histogram(image, cornerMask)
        features.extend(hist)
    hist = histogram(image, ellipMask)
    features.extend(hist)

    return features
```

```
#Histogram
def histogram(image, mask):
    hist = cv2.calcHist([image], [0, 1, 2], mask, bins,
        [0, 180, 0, 256, 0, 256])
    if imutils.is_cv2():
        hist = cv2.normalize(hist).flatten()
    else:
        hist = cv2.normalize(hist, hist).flatten()

    return hist

#Sercher
def search(queryFeatures, limit = 3):
    results = {}
    with open(indexPath) as f:
        reader = csv.reader(f)

        for row in reader:
            features = [float(x) for x in row[1:]]
            d = chi2_distance(features, queryFeatures)
            results[row[0]] = d
        f.close()
    results = sorted([(v, k) for (k, v) in results.items()])

    return results[:limit]

def chi2_distance(histA, histB, eps = 1e-10):
    d = 0.5 * np.sum([((a - b) ** 2) / (a + b + eps)
        for (a, b) in zip(histA, histB)])

    return d
```

3. Importation les données d'apprentissage:

L'extraction des caractéristiques et les écrire dans le fichier index001.csv

```
#list des noms de colomns (s0 -> s1440)
list_Colomns = ['s' + str(x) for x in range(0,1441)]
```

```
# index001.cvs est un fichier contient des caractéristiques des images
Path = "index001.csv"
#le nom du base de donnees d'apprentissage
Trainingdatabase = "DB2C"
#ouvrir le fichier index001.csv
output = open(Path, "w") # avec l'option d'ecriture
# glob pour parcourir les dossiers qui contient les images
for imagePath in glob.glob("./"+Trainingdatabase + "/*/*.jpg"):
    # l'extraction le nom de l'image
    imageID = imagePath[imagePath.rfind("/") + 1:]
    target = imagePath[imagePath.find("/",2) + 1:imagePath.rfind("/")]
    image = cv2.imread(imagePath)
    # l'extraction des caractéristiques des images
    features = describe(image)
    features = [str(f) for f in features]
    #ecrire dans le ficher index001.csv
    output.write("%s;%s\n" % (imageID+";"+target,";".join(features)))
# fermer le fichier index001.csv
output.close()
# lire a partire du index001.csv
data = pd.read_csv('index001.csv',sep=";")
y = data.obj_ship
X_train = data.drop('obj_ship',1)
X_train.columns = list_Colomns
```

4. Importation les données de test:

```
#le nom du base de donnees du Test
TestDatabase = "DataToPredict"
#ouvrir le fichier index001.csv
output = open(Path, "w")
images = []
for imagePath in glob.glob("./"+TestDatabase + "/*.jpg"):
    imageID = imagePath[imagePath.rfind("/") + 1:]
    images.append(imageID)
    image = cv2.imread(imagePath)
    features = describe(image)
    features = [str(f) for f in features]
    output.write("%s;%s\n" % (imageID, ";".join(features)))

output.close()

X_test = pd.read_csv('index001.csv',sep=";")
X_test.columns = list_Colomns
```

5. Traitement sur les données de test et d'apprentissage :

```
# remplacer le obj_car par 1 et obj_ship par 0 pour utiliser Y dans Fit
y[y=='obj_car']=1
y[y=='obj_ship']=0

#pour supprimer le colomn s0 qui contient le nom des images
X_test = X_test.drop('s0',1)
#pour supprimer le colomn s0 qui contient le nom des images
X_train = X_train.drop('s0',1)

#Remplacer les données inconnu par 0
X_train = X_train.replace('unknown',0)
X_test = X_test.replace('unknown',0)

y=y.astype('int')
```

6. La fonction Fit (apprentissage) :

```python
from sklearn import svm
from sklearn.model_selection import train_test_split

# utiliser le SVC avec les parametre kernel et verbose et degree
classificateur=svm.SVC(kernel='linear',verbose =True,degree = 4)

#appliquer le fit
classificateur.fit(X_train,y)
# la prediction de X_test
Y_pred = classificateur.predict(X_test)

# y pour les donnees de test a predire
y_test = [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0]

#ce variable va contenir les classes (car , ship)
classe = []
for i in Y_pred:
    if i==1:
        classe.append('obj_car')
    else:
        classe.append('obj_ship')
```

7. Insérer les résultats dans le fichier result.csv:

```python
#le nom du base de donnees du Test
TestDatabase = "DataToPredict"
output = open(indexPath, "w")
imageID = []
for imagePath in glob.glob("./"+TestDatabase + "/*.jpg"):
    imageID.append(imagePath[imagePath.rfind("/") + 1:])
result = pd.DataFrame({'name': imageID[1:] , 'classe': classe})
result.to_csv('result.csv',index=False)
```

8. Le calcule du taux de test :

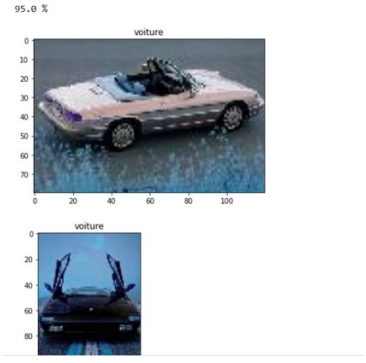
```python
TestDatabase = "DataToPredict"

output = open(indexPath, "w")
imageID = []
for imagePath in glob.glob(TestDatabase + "/*.jpg"):
    imageID.append(imagePath[imagePath.rfind("/") + 1:])

imageID = imageID[1:]

print("le taux de test est : "+str(100*clf.score(X_test,y_test))+"%")
for i in range(len(imageID)):
    if Y_pred[i]==1:
        classe = 'voiture'
    else:
        classe = 'bateau'
    img = cv2.imread("./DataToPredict/"+imageID[i])
    plt.imshow(img)
    plt.title(classe)
    plt.show()
```

9. Résultat et taux de test (95%):



10. Conclusion :

Support Vector Machine (SVM) est utilisé comme classificateur linéaire ou non linéaire basé sur le noyau utilisé. Si nous utilisons un noyau linéaire, alors le classificateur et donc la limite de prédiction sont linéaires. Ici, pour séparer deux classes, nous devons tracer une ligne. La ligne est telle qu'il y a une **marge maximale**. Cette ligne est tracée à égale distance des deux ensembles. Nous dessinons deux autres lignes de chaque côté, appelées vecteurs de support. Les SVM apprennent des vecteurs de support, contrairement aux autres modèles d'machine learning qui apprennent à partir des données correctes et incorrectes. Par exemple, supposons que nous ayons deux classes: les pommes et les oranges. Dans ce cas, SVM apprend les exemples qui sont les plus à droite dans les pommes (une pomme ressemblant à une orange) et les plus à gauche dans les oranges (une orange ressemblant à une pomme); c'est-à-dire qu'ils examinent les cas extrêmes. Par conséquent, ils fonctionnent mieux la plupart du temps.