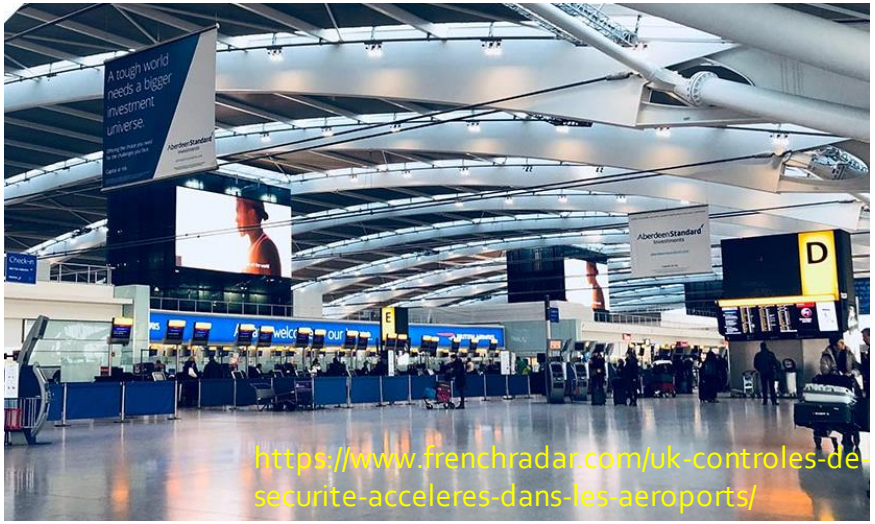
n°2600

TIPE 2021

Enjeux sociétaux

Sécurité aéroportuaire



https://www.frenchradar.com/uk-controles-de-securite-acceleres-dans-les-aeroports/

Automatisation

Comment mettre le Machine Learning au service des contrôles de sécurité ?

Différencier un pistolet d'un couteau

- Méthode des k plus proches voisins

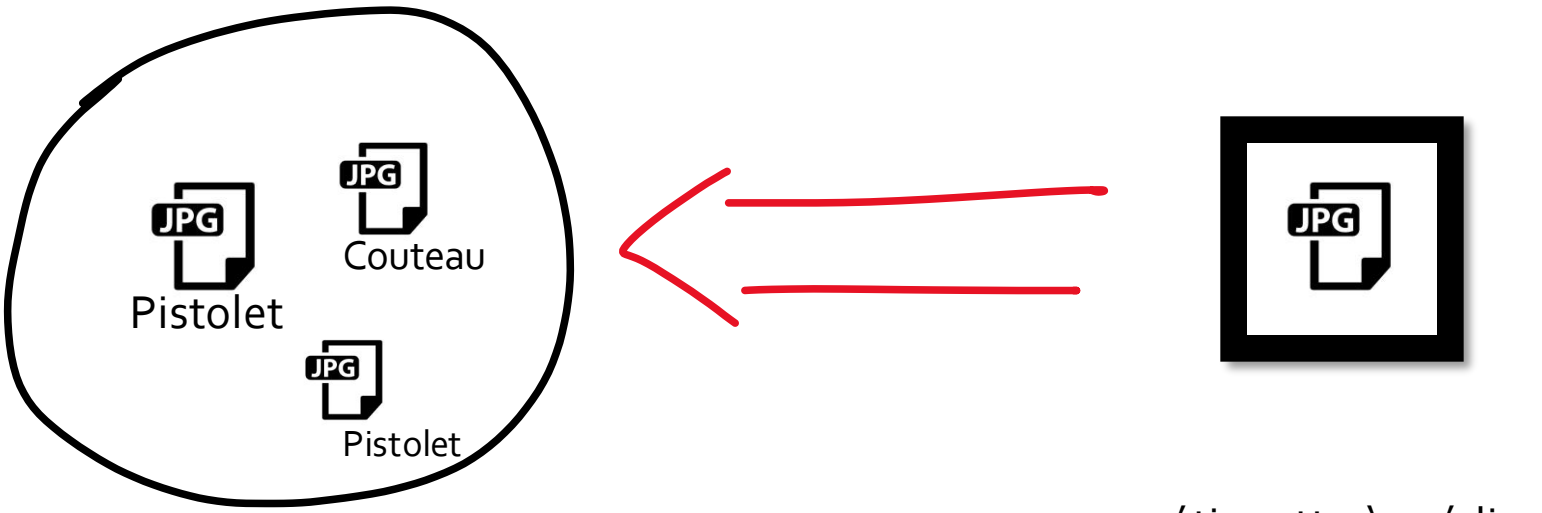
L'étiquette d'une image

F(image) = étiquette

{"pistolet","couteau"}

Images enregistrées

Nouvelle image



avec étiquettes

étiquette à prédire

Une image : une matrice de pixels

- Comparaison des distances (norme 1)
- Parmi les k plus proches, l'étiquette majoritaire

```
def gris(im):   #Renvoie une image qui correspond à im en noir et blanc
    im_gris = np.empty([len(im),len(im[0])])
    for i in range(len(im)):
        for j in range(len(im[0])):
            im_gris[i][j] = round(np.mean(im[i][j]))
    return im_gris
```

Griser les images

```python
def Im_distance(A,B):   #Calcule la distance entre les images A et B
    distance = 0
    n = len(A)
    for i in range(n):
        for j in range(n):
            distance += abs(A[i][j] - B[i][j])
    return distance
```

```python
def etiquette_majoritaire(X,indices_min):   #Renvoie l'étiquette majoritaire parmis
                                            #les images dont l'indice est dans la liste
                                            #indices_min

    nb_pistol,nb_knife = 0,0
    etiquettes = []

    for indice in indices_min :
        etiquettes.append(X[indice][1])

    for el in etiquettes:
        if el == "pistol":
            nb_pistol += 1
        if el == "knife":
            nb_knife += 1

    if nb_pistol > nb_knife:
        return "pistol"
    if nb_knife > nb_pistol:
        return "knife"
    else :
        return etiquette_majoritaire(X, indices_min[:-1])   #si il n'y a pas d'etiquette majoritaire,
                                                            #on ignore celle de l'image la plus éloignée
```

```
def Im_kNN(X,A,k):      #Programme des k plus proches voisins,X: liste de couples (image,étiquette)
                        #A: nouvelle image
    liste_distances = len(X)*[0]

    for i in range(len(X)):
        liste_distances[i] = Im_distance(X[i][0],A)

    Somme = somme_liste(liste_distances)
    indices_min = []

    for _ in range(k):
        Index_min = liste_distances.index(min(liste_distances))
        indices_min.append(Index_min)
        liste_distances[Index_min] = Somme

    return etiquette_majoritaire(X, indices_min)
```

Le programme

Nos données

Images enregistrées

- Pistolet : 595
- Couteau : 386

Images de test

- Pistolet : 100
- Couteau : 100

Taille des images : 160*120 pixels

Exemples d'images

```
def Test_kNN(X,x,k):    #teste sur plusieur images les capacités prédictives de
    n = len(x)          #Im_kNN(), x: liste couple (nouvelle image, etiquette)
    S = 0
    for i in range(n):

        if x[i][1] == Im_kNN(X,x[i][0],k):
            S += 1

    return S/n
```

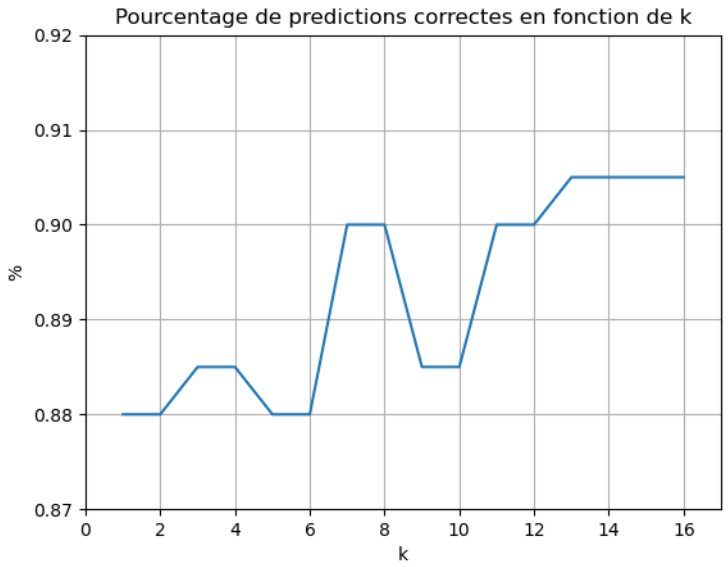
Tester les perfomances

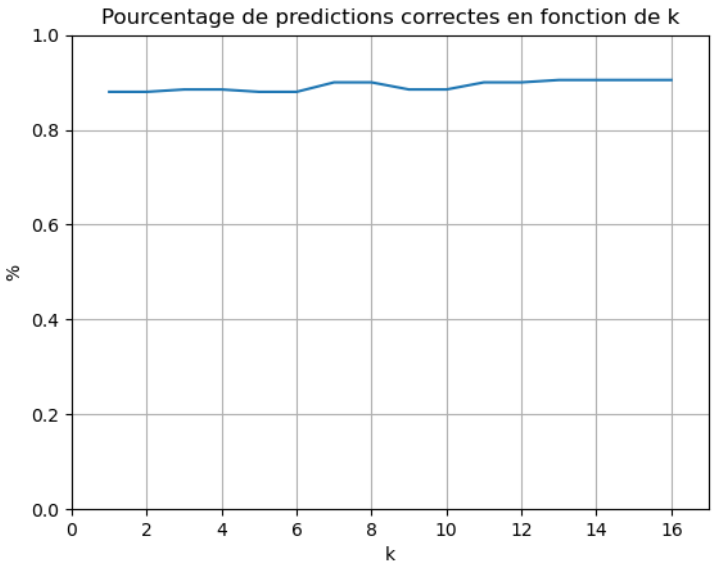
1er résultat avec k=1

```
In [10]: Test_kNN(DATA_train,Pistol_test,1)
Out[10]: 0.95
```

```
In [12]: Test_kNN(DATA_train,Knife_test,1)
Out[12]: 0.81
```

```
In [13]: (0.95+0.81)/2
Out[13]: 0.88
```

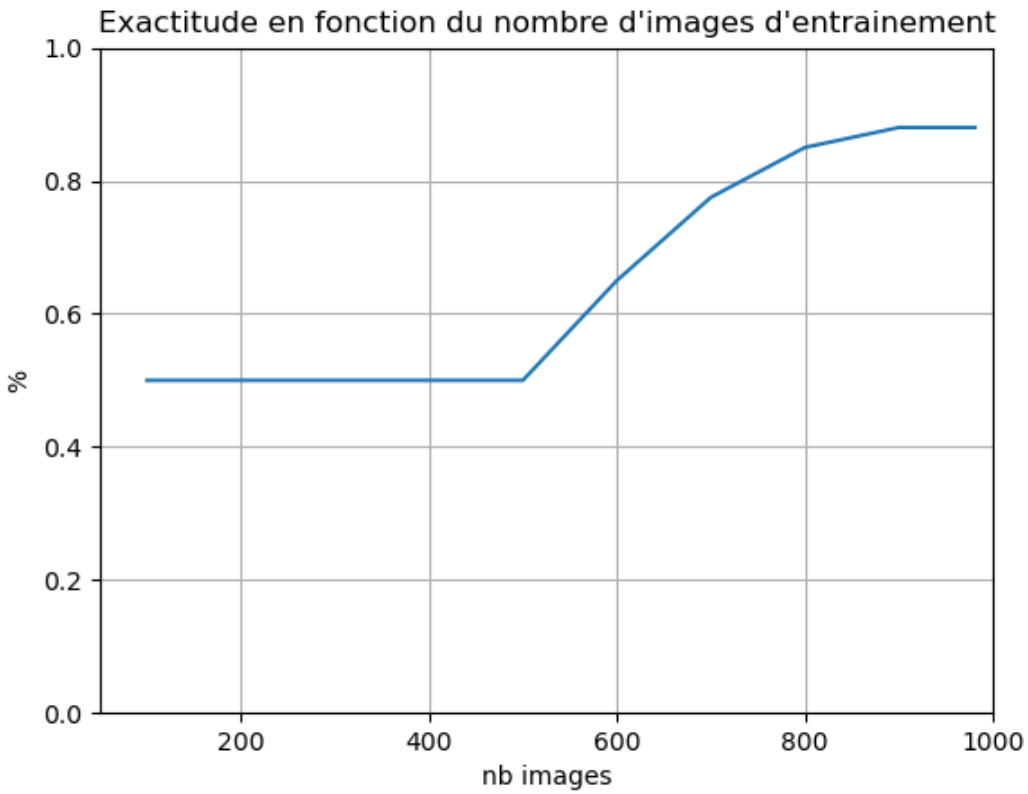


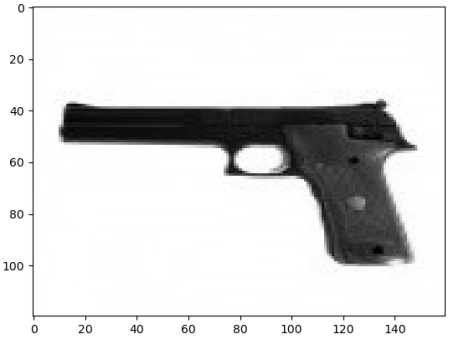


Performances en fonction de k

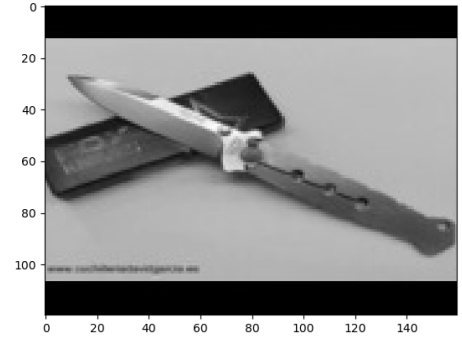
En fonction du nombre d'images

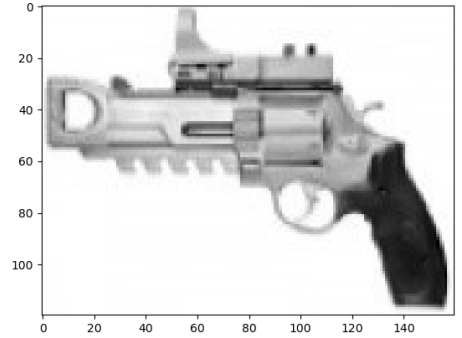
Pour k=1





Les images avec une mauvaise prédiction

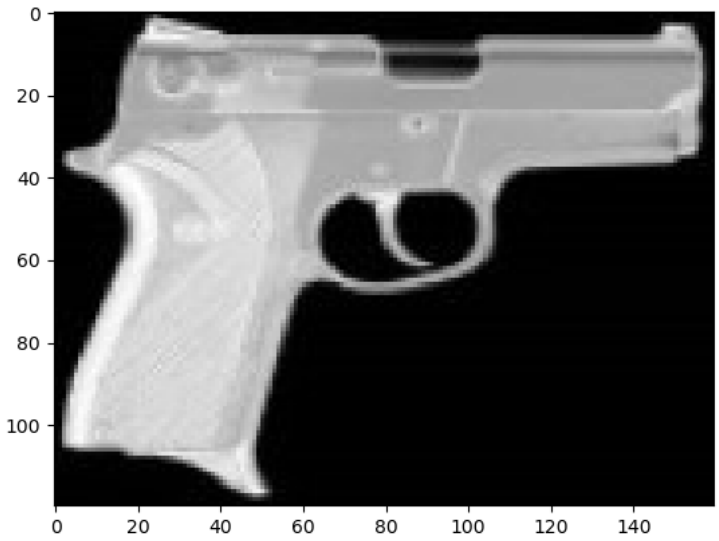




Piéger le modèle

L'image en négatif

```
def negatif_image(Im):  #Renvoie l'image en négatif
    n,N = len(Im),len(Im[0])
    negatif = np.empty([n,N])

    for i in range(n):
        for j in range(N):
            negatif[i][j] = 255 - Im[i][j]

    return negatif
```



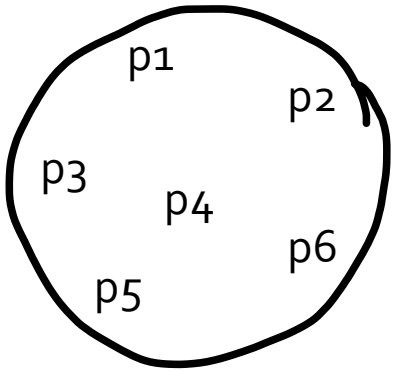

```
print(Test_kNN(DATA_train,Pistol_test_negatif,1))
print(Test_kNN(DATA_train,Knife_test_negatif,1))
```

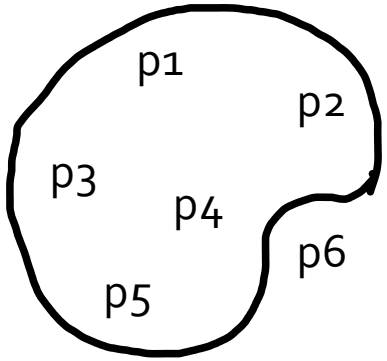
```
In [21]: runfile(
0.58
0.79
```

```
In [23]: (0.58+0.79)/2
Out[23]: 0.685
```

Détection

Ensemble des pistolets enregistrés





Calcule distance élément - ensemble privé de cet élément

Répété pour chaque élément

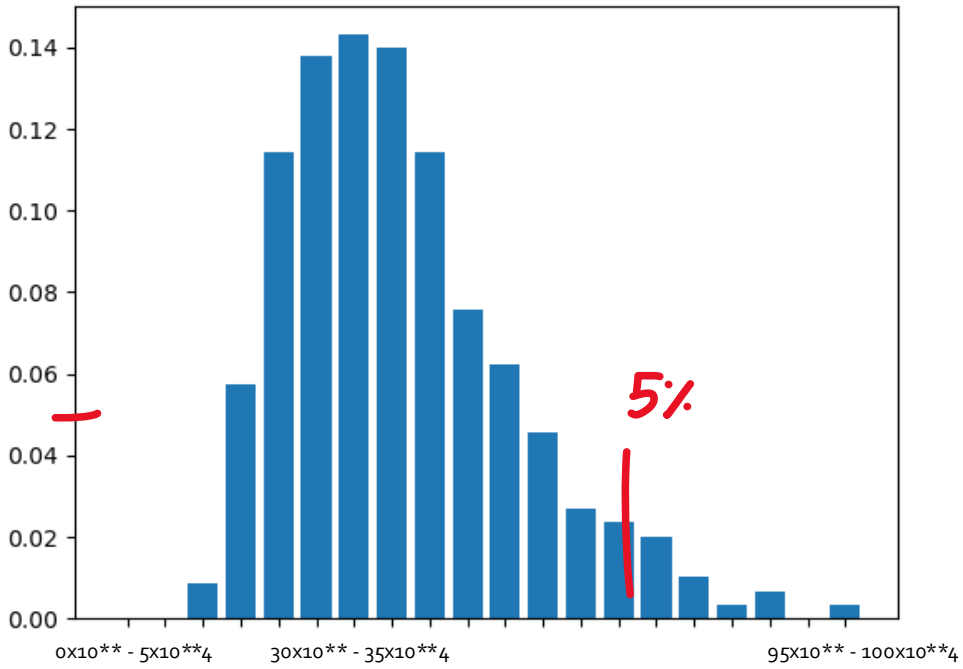
Définir seuil

```python
def distances_mutuelles(X):     #Renvoie la liste des distances entre les éléments de X et X privé de
    X_copy = copier_liste(X)    #celui-ci
    distances = []

    for i in range(len(X)-1):
        A = X_copy[i]
        X_copy = X_copy[:i] + X_copy[i+1:]
        distances_temp = []
        print(i,"/980")

        for j in range(i,len(X)-1):
            distances_temp.append(Im_distance(A[0],X_copy[j][0]))


        distances.append(min(distances_temp))
        X_copy = X_copy[:i] + [A] + X_copy[i:]

    return sorted(distances)
```

```python
def detection_seuil(X,A,SEUIL):     #Vérifie si la distance entre A et X est inférieur à un seuil
    distance_min = distance_minimal(X,A)

    if distance_min < SEUIL :
        return True
    else :
        return False
```

```python
def seuil_pourcents(L,p,pas):    #L: liste triée,p: le pourcentage souhaité
    Max = L[-1]                  #renvoie le seuil tq p% des éléments de L soient supérieur
    seuil = Max

    while compter_presence(L,seuil,Max) < p:
        seuil -= pas

    return seuil
```

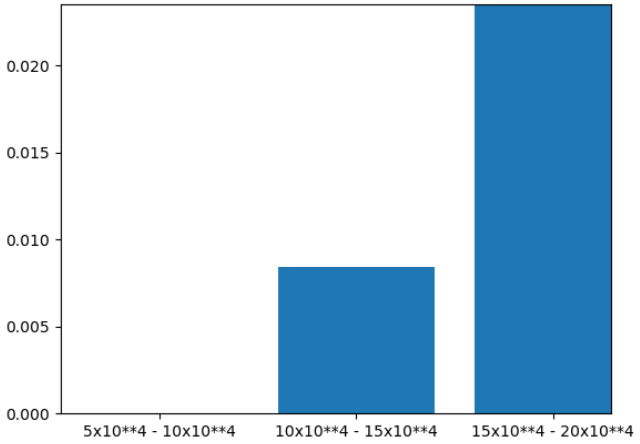
Définir le seuil



Pour pistolets

```
In [12]: moyenne(distances_mutuelles_pistol)
Out[12]: 391728.0

In [13]: ecart_type(distances_mutuelles_pistol,391728.0)
Out[13]: 162480.9167624245
```

```
In [34]: seuil_pourcents(distances_mutuelles_pistol,0.05,100)
Out[34]: 688749.0
```

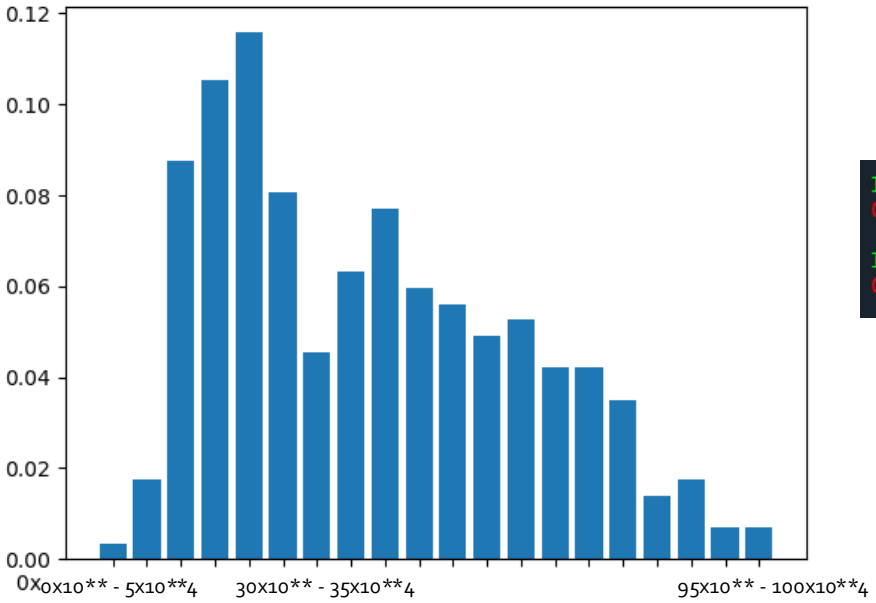


Résultat

```
In [40]: test_detection(DATA_train[0:595],Pistol_test,688749.0)
Out[40]: 0.99
```

Pour couteaux

```
In [45]: seuil_pourcents(distances_mutuelles_knife,0.05,100)
Out[45]: 818091.0

In [46]: test_detection(DATA_train[695:1081],Knife_test,818091.0)
Out[46]: 0.98
```




```
In [50]: moyenne(distances_mutuelles_knife)
Out[50]: 412910.0

In [51]: ecart_type(distances_mutuelles_knife,412910.0)
Out[51]: 247349.3065827949
```

Conclusion

En pratique

- Des outils plus efficaces
- Les réseaux de convolution
- Des fonctions déjà disponibles