Rapport: **Meilleur Classifieur pour La reconnaissance des Digits audios**

Hichem, Thierry, Perrine, Quentin

Ce brief a pour objectif de manipuler les différentes techniques de classification supervisée sous Python, comme KNN, SVM, Forêt Aléatoire, XGBoost et autres.
Dans
cet atelier, nous appliquons apprentissage supervisé pour reconnaître les
Digits à partir d'un enregistrement audio.

- Collecte audio pour créer une base de données.

Pour collecter les données nous avons utilisé la fonction collection() fournie dans le brief pour nous enregistrer en prononçant à haute voix les chiffres de 0 à 9. Nous avons ainsi créé des jeux de données contenant 10 à 20 séries de chiffres, chaque fichier audio correspondant à un chiffre étant ensuite scindé en 12 fréquences différentes qui servent de bases pour entraîner les différents modèles.

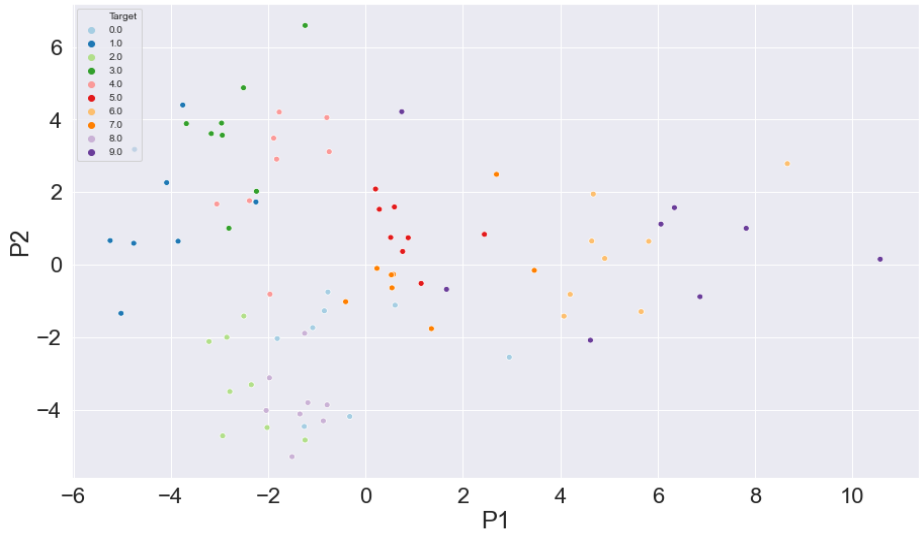
Observations:

La qualité des captations audio n'est pas vérifiée

- Analyse, pré-traitement et visualisation des données.

Représentation des données formatées par une PCA afin de les représenter en 2 dimensions. On peut observer le regroupement des enregistrements d'une même classe (digit) et contrôler leur similarité

- Réalisation d'un pipeline et d'un gridsearch



- Choix de meilleur modèle:

  Pour choisir le meilleur modèle nous les appliquons tous à la suite sur des mêmes jeux d'entraînement et de test et nous comparons leurs scores respectifs. Le fonctions grid-search nous permet d'obtenir les meilleurs hyperparamètres pour chaque modèle. Une fois la sélection faite, nous pouvons sauvegarder le modèle à l'aide du module joblib afin de pouvoir l'appliquer lors des tests en temps réel.

Pour les tests en temps réel nous appliquons donc le modèle choisi avec ses meilleurs hyperparamètres sur les données obtenues via la fonction Tools.Rec et nous effectuons une prédiction sur celles-ci.

- Intégration de ce modèle dans une application pour test temps réel:

Application en Temps Réel

Entrée [135]:
```
from Tools.tools import rec
from Tools.tools import collection
```

Entrée [138]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scale

r.transform(X_train)
model = SVC(C= 10, kernel= 'rbf')
model.fit(X_train, y_train)
```

Out[138]: SVC(C=10)

Entrée [139]:
```
rec(scaler, model)
```

```
Attention, l'enregistrement commence dans :
5
4
3
2
1
0
Prononcer votre Digit :


------------------
Digit :  9.0
```