Compte_Rendu-Gwenn_Le_Roch

March 22, 2022

1 Introduction

Vous pourrez retrouver ici le détail des différentes étapes réalisées lors du projet DigitsRecognition visant à développer un modèle IA pour la reconnaissance des Digits provenant d'un enregistrement audio.

2 Sommaire

1. Collection d'une base de données.
2. Analyse, prétraitement et visualisation des données.
3. Préparation des données pour l'apprentissage.
4. Préparation de Pipeline en utilisant « sklearn.pipeline.Pipeline ».
5. Choix de meilleur modèle.
6. Vérification de l'efficacité de ce modèle à des nouvelles données.
7. Intégration de ce modèle dans une application pour test temps réel.

3 Collection d'une base de données

Afin de développer un modèle IA pour la reconnaissance des Digits provenant d'un enregistrement audio, il faut tout d'abord collecter des données sur lesquelles entraîner les modèles étudiés.

Cette récolte a été possible à l'aide de la fonction `collection()` du module `Tools`, qui permet de sauvegarder un enregistrement audio de chaque digit et de leur appliquer un cepstre avant de stocker les résultats dans un fichier csv.

Différents enregistrements ont été effectués, sur la base d'une dizaine par fichiers csv. Plus les données enregistrées seront nombreuses, plus l'estimation des digits audio sera pertinent ; cependant des enregistrements réalisés dans des conditions différentes peuvent dégrader les performances du modèle.

4 Analyse, prétraitement et visualisation des données

La majeure partie de cette étape a été réalisée dans la fonction `collection()`, il ne reste alors plus qu'à charger en mémoire les données à l'aide de la librairie `pandas`, de les analyser avec la méthode `info()` et de les visualiser avec la fonction `plot()` du module `matplotlib.pyplot`.

5 Préparation des données pour l'apprentissage

Les données ont été préparées à l'aide de la fonction `train_test_split()` de la bibliothèque `sklearn.model_selection` afin de les séparer en données d'entraînement et en données de test.

Avant d'être utilisées pour développer les différents modèles, les données d'entraînement sont normalisées avec `MinMaxScaler()` puis standardisées avec `StandardScaler()` du module `sklearn.preprocessing`.

6 Préparation de Pipeline en utilisant « sklearn.pipeline.Pipeline »

Plusieurs classificateurs ont été étudiés lors de ce projet :

- KNN (`KNeighborsClassifier` du module `sklearn.neighbors`)
- SVM (`SVC` du module `sklearn.svm`)
- Decision Tree (`DecisionTreeClassifier` du module `sklearn.tree`)
- Random Forest (`RandomForestClassifier` du module `sklearn.ensemble`)
- Gradient Boosting (`GradientBoostingClassifier` du module `sklearn.ensemble`)
- MLP (`MLPClassifier` du module `sklearn.neural_network`)
- XGBoost (`XGBClassifier` du module `xgboost`)

Ils ont été intégrés au sein de plusieurs pipeline créés à l'aide de la classe `Pipeline` de la bibliothèque `sklearn.pipeline`, qui contruisent les modèles après avoir appliqué aux données d'entraînement les transformations indiquées plus haut.

7 Choix du meilleur modèle

Ce choix s'effectue à l'aide de la classe `GridSearch` de la bibliothèque `sklearn.model_selection` qui permet à partir d'un estimateur ou d'un pipeline et de plusieurs ensembles de paramètres, d'estimer le paramétrage le plus pertinent et d'évaluer sur les données d'entraînement le modèle retenu.

8 Vérification de l'efficacité de ce modèle à des nouvelles données

Afin de valider le modèle choisi, on le reconstruit en lui attribuant les meilleurs paramètres associés, à l'aide de la méthode `set_params()` et en le réentraînant sur les données d'entraînement.

À l'aide du module `sklearn.metrics`, le modèle est alors évalué sur les données de test avec la méthode `predict()`, avant d'étudier la précision du modèle avec la fonction `accuracy_score()` et de caractériser la qualité du modèle à l'aide de la classe `ConfusionMatrixDisplay`.

9 Intégration de ce modèle dans une application pour test temps réel

Une fois le modèle choisi, on peut alors l'intégrer à l'aide de la bibliothèque `sklearn.metrics` : il est enregistré à avec la fonction `dump()` et récupéré avec la fonction `load()`.

À partir du notebook `Etapes_du_Projet.ipynb`, il est donc possible d'enregistrer de nouvelles données et d'entraîner de nouveaux modèles.

Le notebook TestTempsRéel.ipynb sert quant à lui à reconnaître un digit provenant d'un enregistrement audio à partir du modèle `20220320_2330_XGBClassifier_0.91.pkl` qui a été développé à partir d'enregistrements effectués à l'aide du microphone intégré dans un ordinateur portable, à un volume sonore bas-moyen.

Les meilleurs tests ont ainsi été réalisés dans les mêmes conditions que les données d'entraînement, c'est-à-dire sans bruits parasites, à voix normale et à, à peu près, 10cm du microphone.