Multimodal Processing, Recognition, and Interaction (MPRI)

Challenge

Auteur : Spinelli Isaia et

Razafimamonjy Liva

Prof : Stefan Carrino

Simon Ruffieux et

Elena Mugellini

Carrino Francesco

Date : 22.12.2020

Salle : A2 – Lausanne

Classe : MPRI

Table des matières

[Introduction - 2 -](#_Toc57621513)

[Conclusion - 3 -](#_Toc57621514)

[Difficultés rencontrées - 3 -](#_Toc57621515)

[Compétences acquises - 3 -](#_Toc57621516)

[Résultats obtenus - 3 -](#_Toc57621517)

[Annexe - 3 -](#_Toc57621518)

# Introduction

Ce challenge à pour objectif d’appliquer les concepts théoriques vus en classe. Il permet de travailler sur des données réelles et d’essayer d’améliorer au maximum la précision d’un modèle.

L'ensemble de données contient des données enregistrées dans le cadre de la thèse de doctorat de Quentin Meteier. Différents capteurs ont été utilisés pour enregistrer les données physiologiques lors des séances de conduite réalisé avec un simulateur de voiture. Les sujets expérimentaient une conduite autonome de niveau 3. Cela signifie qu'ils pouvaient effectuer une tâches secondaires (au lieu de conduire) mais devaient reprendre le contrôle du système lorsque des événements dangereux se produisaient (passage d'animaux, objets sur la route, etc.).

Il y a donc deux classes distinctes :

1. Conduire normalement sans tâche secondaire
2. Conduire en faisant une tâche secondaire

Pour plus d’informations concernant cet expérience, vous pouvez consulter l’énoncé du challenge en annexe [[1].](#_Annexe)

Afin de mener à bien ce challenge, voici dans l’ordre toutes les étapes qu’il a fallu effectuer :

1. Prise en main du code fourni
2. Survol des données fournies
3. Mise en place d’un système afin de choisir si on souhaite retraiter les données EDA et/ou ECG.
4. Mise en place d’un système afin d’enregistrer et charger l’ensemble des données traitées.
5. Divisé les données entre entrainement et test.
6. Mise en place de notre algorithme.
7. Afficher des informations concernant notre modèle et ses performances (Gini importance, learning curve, la matrice de confusion, les performances générales).
8. Mise en place d’un système pour enregistrer et charger notre meilleur modèle.
9. Comparaison des améliorations avec le seed le plus « compliqué » (limité, pas très représentatif des réelle nouvelle données)
10. Comparaison des améliorations avec Kfold.
11. Amélioration des caractéristiques (features).
12. Recherche des meilleurs paramètres.
13. Comparaison des différents levels de segmentation
14. Fusion des résultats des modèles

Question: Does segmenting improve accuracy ?

Maintain some temporal information ?

En vue du nombre d’étape, uniquement les plus intéressantes seront présentées. Dans tous les cas, l’entièreté de code python est en annexe [[2]](#_Annexe).

# Retraitement des données

# Sauvegarde et chargement des données prétraitées

# Affichage des informations concernant notre modèle et ses performances

# Sauvegarde et chargement le meilleur modèle

# Comparaison des améliorations

Pour commencer, j’ai pensé à chercher et observer le pire cas de séparation des données afin d’améliorer mon système en me basant sur le pire cas. Pour ce faire, il a fallu tester plusieurs seed lors de la séparation des données afin d’utiliser un set de test compliqué. Voici la commande permettant de séparer les données en fonction du seed choisis :



Ensuite, on peut voir un tableau résumant différents seed testés et la valeur F1 moyenne observée :



Dans notre cas, on peut voir que le seed N°21 sépare les données de la manière le plus compliquée.

Après plusieurs comparaisons, cette méthode a été abonnée car elle est surement limitée et donc représente mal des nouvelles réelles données.

Afin de comparer KFOLD !!

# Conclusion

## Difficultés rencontrées

* Trop de data !...

## Compétences acquises

* ..

## Résultats obtenus

….

# Annexe

1. Enoncé du challenge
2. Code Python

Date : 22.12.20

Nom de l’étudiant : Spinelli Isaia et Razafimamonjy Liva