

IMT Atlantique - FIL A1 - Mini-Projet ACDC

Rémi BARDON

Novembre 2020

Présentation

Thème

Nettoyage, préparation et visualisation de données

Responsable

Charles Prud'homme – charles.prudhomme@imt-atlantique.fr

Sujet

Les attentes sociétales en matière d'éthique, de droit et de bien-être animal ont considérablement progressé au cours des dernières années. Les formations en physiologie animale doivent adapter l'offre pédagogique aux attentes des apprenants et à l'évolution de la législation, dans le respect du bien-être animal.

Dans ce contexte, il vous est demandé de participer à la conception d'un simulateur réaliste piloté par de l'intelligence artificielle, se substituant complètement aux animaux vivants pour les travaux pratiques de physiologie expérimentale.

Votre travail consistera à nettoyer, préparer et visualiser les séries temporelles issues des données de physiologie expérimentale collectées au cours des dix dernières années à Oniris. Ce travail devra permettre de préparer le pilotage des fonctions vitales du robot-lapin développé en parallèle.

Commentaires

Vous trouverez des informations sur l'extraction de la tendance et de la saisonnalité sur le site suivante :

<https://www.machinelearningplus.com/time-series/time-series-analysis-python/>

En particulier, à partir de la section 6.

Dans notre cas, le modèle additif est tout à fait adapté.

<https://anomaly.io/seasonal-trend-decomposition-in-r/index.html>

https://en.wikipedia.org/wiki/Decomposition_of_time_series

Décisions d'équipe

Les décisions suivantes ont été prises le 8 octobre 2020 dans le but de faire émerger les besoins liés au code métier produit par chacun des 4 étudiants.

Seuls les besoins ont été décidés en commun, l'API, quant à elle, est le fruit du travail individuel des étudiants.

Données accessibles

- Donnée brute par **timestamp**, **tag** et mesure
- Tendance par **timestamp**, **tag** et mesure
- Saisonnalité par **timestamp**, **tag** et mesure
- Bruit par **timestamp**, **tag** et mesure
- Étendue (**min** et **max**) de la donnée brute, la tendance, la saisonnalité et le bruit, le tout par **tag** et mesure
- Plages de données omises (début/fin)
- **Tags** présents dans le fichier
- **Tags "préparation"** et **"euthanasie"** (disponibles, mais pas de convention de nommage)
- Commentaire en haut du fichier (le cas échéant)

Formattage

- Si une plage de données est retirée, la suite doit être "collée" (éditer les **timestamps** suivants)

Décisions personnelles

Données inaccessibles

- J'ai décidé de ne pas créer de **Tag "euthanasie"**, car il est impossible de savoir quand celle-ci a lieu. Nous n'avions pas pensé à ça.

Possibilités & restrictions

- 0 ou 1 tag sélectionné à la fois (séance entière ou seulement une partie)
- 1 ou plusieurs mesures sélectionnées à la fois

Use cases

"Les 4 graphiques" désigne ici les graphiques temporels (temps en abscisses) de données brutes, tendance, saisonnalité et bruit (en ordonnées).

- Pas de fichier importé
 - Affichage de l'**interface vide**
 - 1 mesure sélectionnée (valeur par défaut)
 - Pas de tag sélectionné
- Fichier importé
 - Affichage de la **pression artérielle** sur les 4 graphiques pour la **séance entière**
 - 1 mesure sélectionnée
 - Pas de tag sélectionné (séance entière)
 - Affichage de la **pression artérielle** sur les 4 graphiques pour la **période de début de la séance**
 - 1 mesure sélectionnée
 - 1 tag sélectionné
 - Affichage de la **fréquence cardiaque** et de la **fréquence respiratoire** sur les 4 graphiques pour la **période de fin de la séance**
 - 2 mesures sélectionnées
 - 1 tag sélectionné

- Le `timestamp` stocké dans `DataPoint` est un `float` et non un `Float` pour réduire la taille des objets en mémoire.
- La valeur d'un `DataPoint` est un `Float` et non un `float` pour permettre l'utilisation des valeurs `NaN`.
- `DataPoint` ne contient pas de référence à `Tag` pour réduire la taille des objets en mémoire.

Par exemple, certaines méthodes comme `toString()` ou `hashCode()` ne sont pas présentes.



```
        loader.cleanData()
        loader.decomposeData()
    } catch (final IOException e) {
        e.printStackTrace();
    } catch (final CsvValidationException e) {
        e.printStackTrace();
    } catch (final NumberFormatException e) {
        e.printStackTrace();
    }
}
```