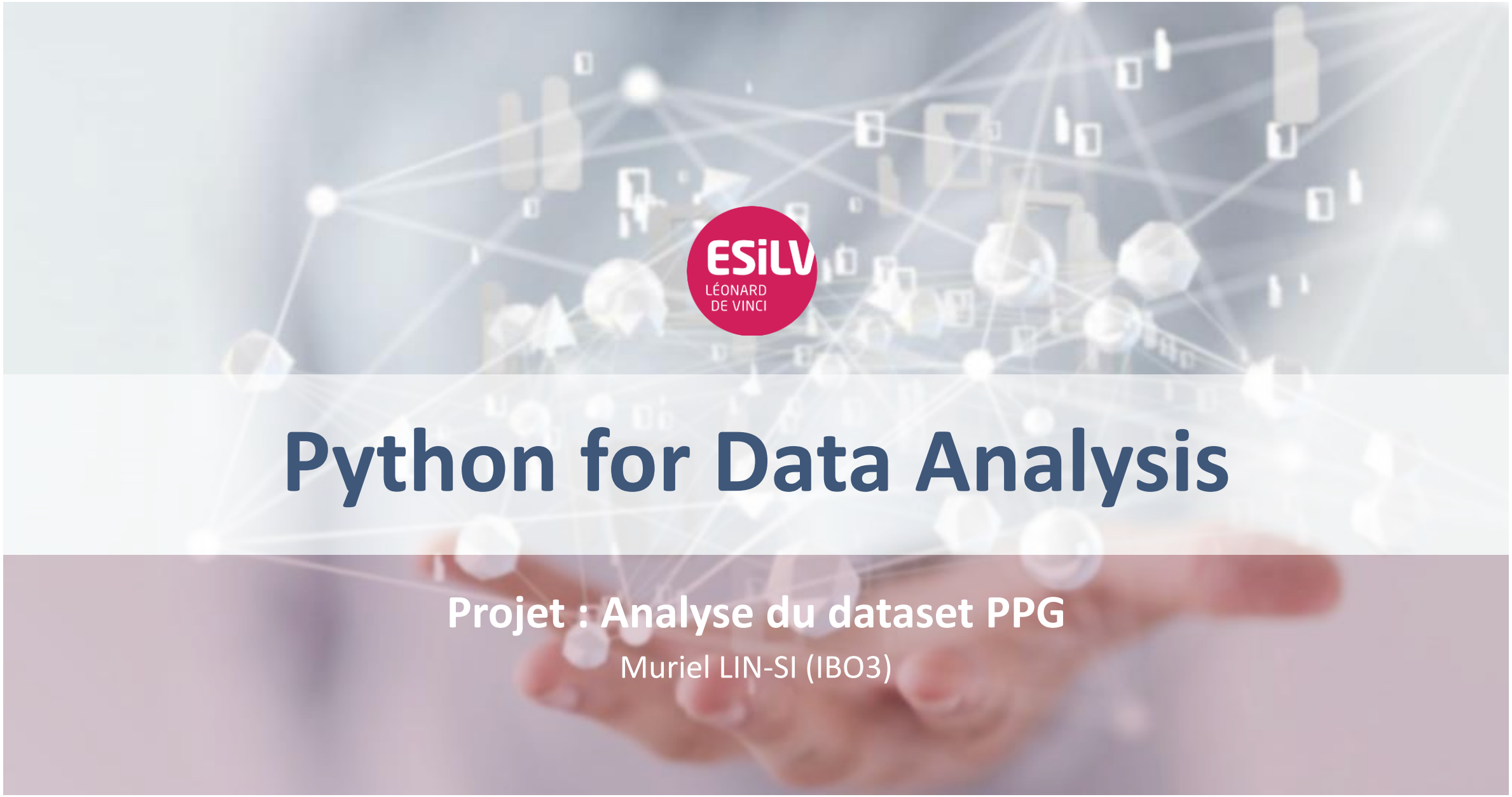


Python for Data Analysis

Projet : Analyse du dataset PPG

Muriel LIN-SI (IBO3)

1) Présentation du dataset

Informations générales

- **Dataset :** PPG
- **Taille du dataset :** 22,4 Go
- **Contexte :** étude utilisant la photopléthysmographie pour mesurer le rythme cardiaque, mouvement, température, respiration, etc. de 15 individus pendant environ 2,5 heures
- **But :** trouver l'activité d'un patient en fonction des données du dataset
- **Structure du dataset :** 15 fichiers au format .pkl contenant les données de chaque individu

Structure plus détaillée du dataset

- **Données pour un patient :**
  - **'activity' :** id de l'activité du patient (numéroté de 0 à 8)
  - **'label' :** rythme cardiaque instantané moyen sur une fenêtre de 8 secondes
  - **'questionnaire' :** information sur le patient
  - **'rpeaks' :** index des pics de contraction maximale du coeur
  - **'signal' :** données récoltés par les capteurs
    - **'chest' :** capteurs se situant sur la poitrine (ACC : accéléromètre 3D, ECG : echocardiogramme, RESP : respiration)
    - **'wrist' :** capteurs se situant sur le poignet (ACC : accéléromètre 3D, BVP : volume de pression sanguin, EDA : conductance de la peau, TEMP : température)
  - **'subject' :** id du patient

2) Difficultés du dataset

Le dataset étant assez complexe à traiter, j'ai dû faire face à quelques problèmes durant son analyse.
Ci-dessous, les principaux problèmes auxquels j'ai dû faire face

Structure des données

- Les données étaient mises sous la forme de dictionnaire
- Les données de chaque patient se trouvaient dans des fichiers différents
- L'élément 'signal' du dictionnaire contenait un tableau dont les cases contenaient eux-mêmes des tableaux
- **Solution :** Aplatir les données pour obtenir uniquement des colonnes dans un dataframe; Pour les informations uniques du patient, les répéter sur toutes les lignes

Fréquence de capture des capteurs

- Les capteurs utilisés n'ont pas été réglés sur la même fréquence (4Hz, 32Hz, 64Hz, 700Hz, etc.)
- Cela impliquait donc que les colonnes du dataset n'étaient pas de la même taille
- **Solution :** Mettre les données sur la même fréquence

3) Mise en forme des données

A) Transformation en DataFrame

- **1ère étape : Créer un dataframe par patient**
  - Mettre les informations du patient (poids, taille, âge, sexe, etc.) sur plusieurs colonnes.
  - Pour 'signal', séparer chaque case en une colonne du dataframe
- **2ème étape : Fusionner tous ces dataframes**
  - Créer un dataframe contenant les données de chaque patient
  - Fusionner en un seul dataframe pour avoir toutes les données accessibles facilement

label	activity	weight	gender	age	height	skin	sport	subject	...	chestACC1	chestACC2	chestECG0	chestResp0	wristACC0	wristACC1	wristACC2
49.611369	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0678	-0.3656	0.015610	4.441833	-0.765625	-0.078125	0.671875
49.611369	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0654	-0.3688	-0.015747	4.876709	-0.765625	-0.078125	0.671875
49.611369	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0662	-0.3708	-0.008743	3.340149	-0.765625	-0.078125	0.671875
49.611369	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0632	-0.3640	-0.339523	0.740051	-0.750000	-0.078125	0.671875
49.611369	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0674	-0.3694	-0.089905	-1.475525	-0.765625	-0.078125	0.671875
49.611369	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0676	-0.3646	0.004349	-1.942444	-0.750000	-0.078125	0.671875
49.611369	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0626	-0.3588	0.103592	-0.576782	-0.765625	-0.078125	0.671875
49.611369	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0598	-0.3536	-0.166672	1.838684	-0.765625	-0.062500	0.687500
50.323992	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0666	-0.3826	0.159714	2.915955	-0.750000	-0.078125	0.671875
50.323992	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0606	-0.3870	-0.129089	1.408386	-0.750000	-0.078125	0.671875
50.323992	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0648	-0.3718	-0.007095	-1.054382	-0.765625	-0.078125	0.671875
50.323992	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0648	-0.3732	0.035477	-2.641296	-0.765625	-0.078125	0.671875
50.323992	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0696	-0.3740	-0.012955	-3.044128	-0.750000	-0.078125	0.671875
50.323992	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0602	-0.3686	0.112839	-2.711487	-0.765625	-0.078125	0.671875
50.323992	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0700	-0.3718	-0.244720	-2.360535	-0.765625	-0.078125	0.671875
50.323992	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0642	-0.3784	-0.066467	-2.201843	-0.750000	-0.078125	0.671875
52.708336	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0102	-0.3502	0.014603	-1.623535	-1.046875	-0.046875	0.843750
52.708336	0.0	78.0	m	34	182.0	3	6	S1	...	-0.0602	-0.3432	0.057541	-1.005554	-0.781250	-0.296875	0.312500

Dataset après transformation en dataframe

3) Mise en forme des données

B) Mettre les données sur la même fréquence

- **1ère étape : Bien lire la documentation du dataset**
  - Pour connaître la fréquence à laquelle chaque élément du dataset ont été capturé
- **2ème étape : Mettre sur une même fréquence (4Hz)**
  - Pour les fréquences > 4Hz :
    - Prendre 1 valeur sur (fréquence/4)
    - Par exemple pour 32Hz : ne garder que 1 ligne sur 8
  - Pour les fréquences < 4Hz :
    - Répliquer les données plusieurs fois

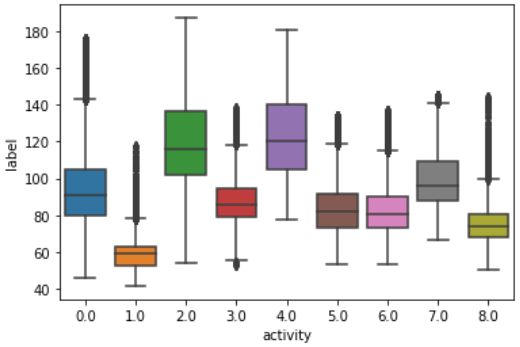
Élément du dataset	Fréquence de capture
activity	4 Hz
label	8 secondes
signal / chest	700 Hz
signal / wrist / ACC	32 Hz
signal / wrist / BVP	64 Hz
signal / wrist / EDA	4 Hz
signal / wrist / TEMP	4 Hz

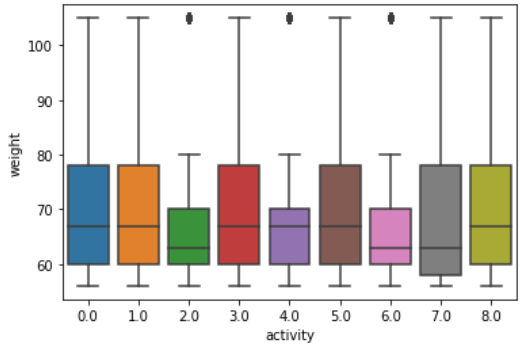
4) Variables créées

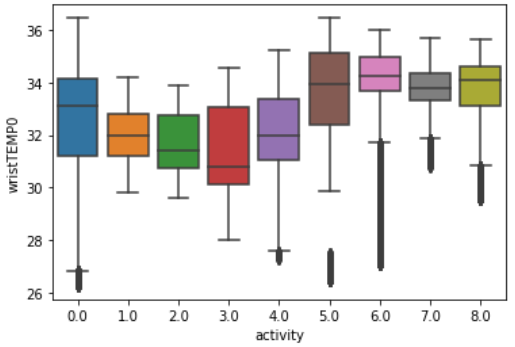
- Après mise en forme, la grande majorité des données sont exploitables directement.
- 2 variable créées :
  - **Nombre de rpeaks par 0,25 sec** (4Hz) :
    - Description de la variable :
      - Cela correspond au nombre de fois que le cœur s'est contracté au maximum
      - Plus les rpeaks sont rapprochés, plus le rythme cardiaque est élevé
    - Création de la variable :
      - Les données ont été capturé par un échocardiogramme à une fréquence de 700Hz
      - Cela signifie que l'on compte le nombre d'index compris entre 700/4 = 175
  - **Sexe masculin de l'individu :**
    - Description de la variable :
      - 'male' sera 1 si l'individu est masculin, 1 si l'individu est féminin
      - Cette variable a été crée pour pouvoir être utilisé dans les modèles de prédiction
    - Création de la variable :
      - Transformation de la colonne 'gender' : tous les 'f' en 0 et de tous les 'm' en 1

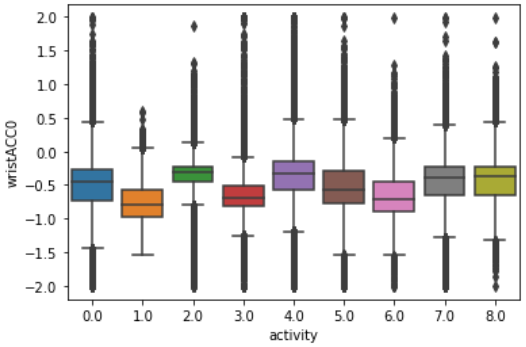
5) Visualisation des données

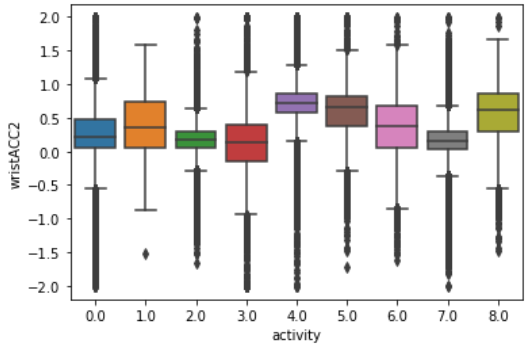
- J'ai visualisé les données sur des boxplot en comparant l'activité avec les features
- Ci-dessous des exemples :

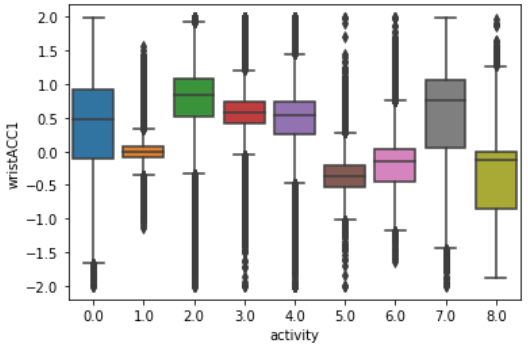












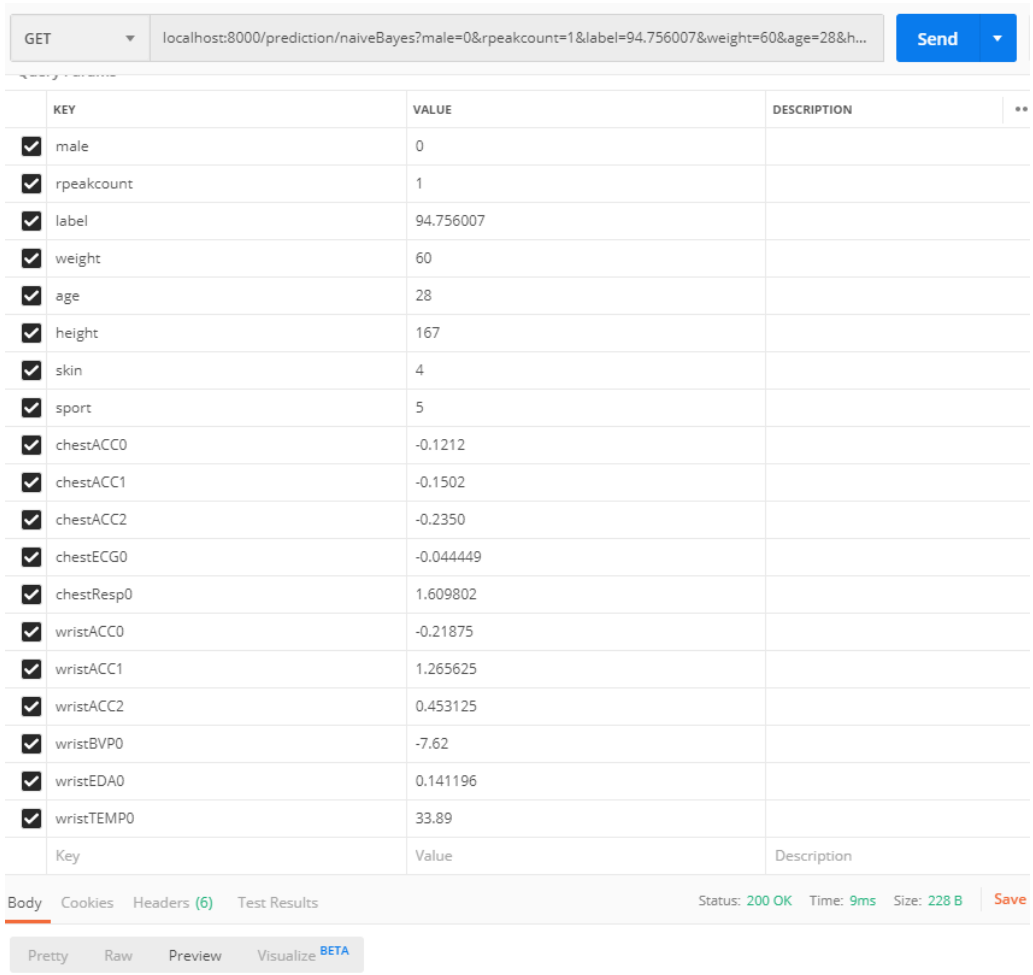
6) Comparaison des algorithmes de prédiction

- 3 algorithmes ont été utilisés pour prédire l'activité d'un individu
  - **Random Forest :** n_estimators = 100, random_state = 54
  - **Naive Bayes**
  - **Nearest Neighbours :** n_neighbours = 3

Algorithme	Accuracy score
Random Forest	0.9868418512154152
Naive Bayes	0.5690247716872936
Nearest Neighbours	0.7181375861602919

7) API Django

- L'API que j'ai implémenté permet de :
  - Prédire l'activité d'un patient à un moment *t*
  - Choisir l'algorithme de prédiction
- 3 chemins possibles :
  - /prediction/randomForest
    - Pour utiliser l'algorithme Random Forest pour prédire l'activité
  - /prediction/naiveBayes
    - Pour utiliser l'algorithme Naive Bayes pour prédire l'activité
  - /prediction/nearestNeighbour
    - Pour utiliser l'algorithme Nearest Neighbour pour prédire l'activité
- Query params à renseigner (obligatoire) :
  - 'male', 'rpeakcount', 'label', 'weight', 'age', 'height', 'skin', 'sport', 'chestACC0', 'chestACC1', 'chestACC2', 'chestECG0', 'chestResp0', 'wristACC0', 'wristACC1', 'wristACC2', 'wristBVP0', 'wristEDA0', 'wristTEMP0'



Exemple d'appel API pour naiveBayes