PPG Analysis

De : Hippolyte ROBICHON
Enseignant : Romain JOUIN

Problème

Nous souhaitons prédire l'activité physique (marche, vélo, monter les escaliers) d'une personne à partir de son activité cardiaque. Nous avons à disposition des données issues de deux capteurs (RespiBAN et Empatica E4), mesurées sur 15 personnes distinctes.

Preprocessing

Conversion des mesures suivantes en 4Hz :

- wrist ACC, initialement à 32Hz
- wrist BVP, initialement à 64Hz
- chest ACC, ECG et Resp, initialement à 700Hz
- wirst EDA et Temp reste inchangé, puisque mesurés en 4Hz

Exclusion des données suivantes, considéré comme “dummy” :

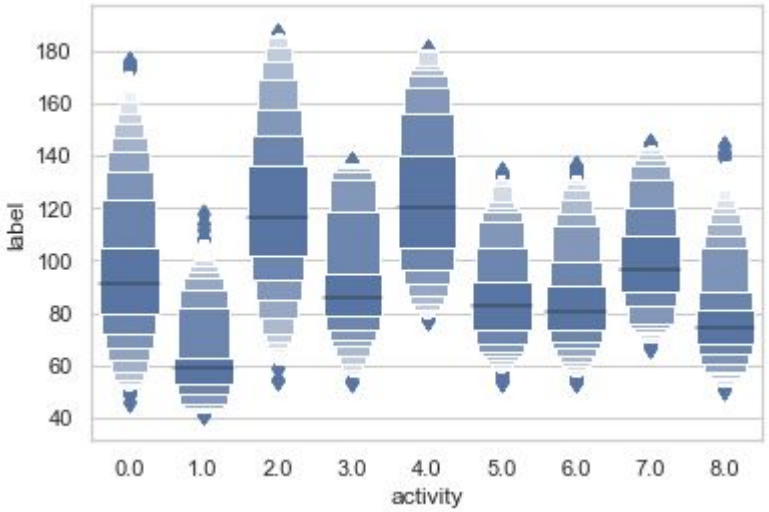
- chest EDA, EMG, et Temp

Conversion de la colonne label (le rythme cardiaque) :

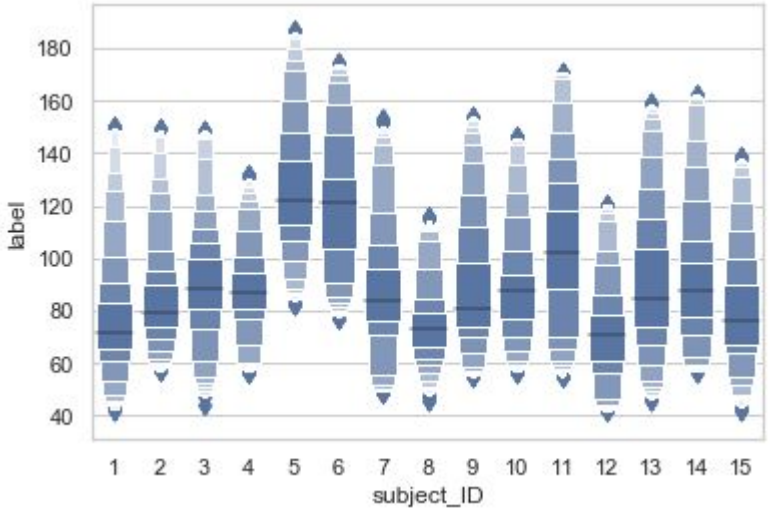
- on duplique ses données 8x afin qu’elles soient de mêmes tailles que les autres colonnes

Data visualization

Rythme cardiaque par activité



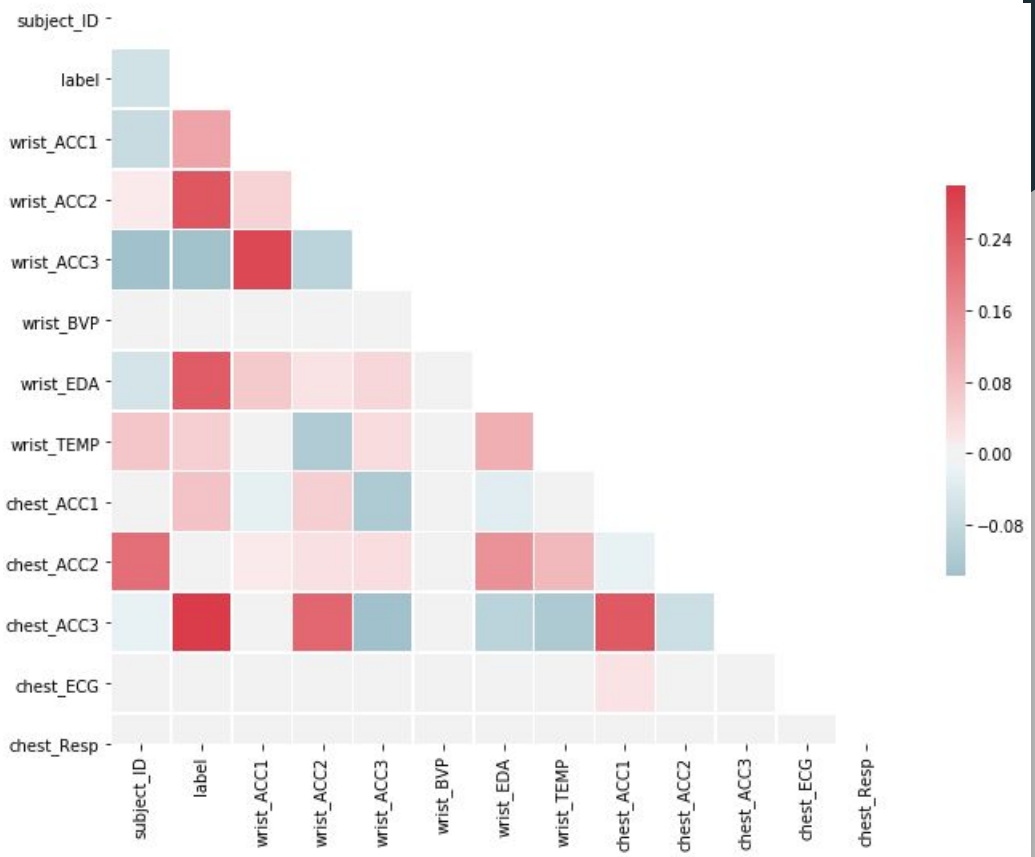
Rythme cardiaque par sujet



Data visualization

Les corrélations sont représentées par les couleurs :

- Rouge indique une corrélation positive
- Bleu indique une corrélation négative



Métrique

accuracy_score, de sklearn

Retourne une valeur entre 0 et 1 représentant la précision de la prédiction.

Model : Logistic Regression

LogisticRegression de sklearn

Prédiction obtenue : 0.5724380261024017

Ce résultat n’étant pas vraiment satisfaisant, je me tourne vers d’autres solutions

Model : XGBClassifier

XGBClassifier de xgboost

Première prédiction avec des paramètres aléatoires : 0.9118, ce qui est déjà très satisfaisant. En améliorant les paramètres de manière empirique, j'obtiens à ma grande surprise 0.9935 :

```
XGBClassifier(objective ='reg:linear',
              colsample_bytree = 0.4, learning_rate = 0.2,
              max_depth = 10, alpha = 10, n_estimators = 150,
              subsample=0.7)
```

Mais une précision aussi élevée ressemble à du "overfitting", j'applique donc la cross-validation :

	0
0	0.498364
1	0.473884
2	0.470162

```
# Cross Validation
kfold = StratifiedKFold(n_splits=3, random_state=7)
results = cross_val_score(xg_reg, X, y, cv=kfold)
```

Le résultat à gauche est assez décevant, j'avoue ne pas bien comprendre pourquoi il est si bas.

Model : Random Forest

RandomForestClassifier de sklearn

A partir d'une max_depth de 50, la précision du model ne s'améliore plus.

On obtient un précision 0.9735 sur l'ensemble du model

Là encore, les résultats de la cross validation (à droite) sont décevants :

	0
0	0.403533
1	0.562876
2	0.439339
3	0.501081
4	0.60224
5	0.540129
6	0.615708
7	0.489574
8	0.516305
9	0.531327

Comparaison des résultats

XGBClassifier obtiens le meilleur résultats, mais est aussi le modèle le plus long à entraîner.

RandomForestClassifier obtient un score honorable pour une vitesse d'entrainement plutôt rapide.

LogisticRegression obtient un score trop médiocre pour être exploité.

La cross validation a montré que les modèles XGBClassifier et RandomForestClassifier était malheureusement overfitté.

API Django

J’ai tenté d’implémenter une api django, mais j’ai rencontré de grosses difficultés : la version de python de Django et celle de Spyder (l’IDE python que j’ai utilisé pour les modèles) étaient différentes, ce qui entraînait de nombreuses incompatibilités, notamment pickle pour sérialiser les modèles.

J’ai donc décidé de me tourner vers Flack, qui permet d’exploiter l’IDE Spyder pour construire une api très simplement.

API Flask

Grâce à cette librairie, j'ai pu créer une API pour tester chacun de mes trois modèles.

On peut soit ne rien passer en paramètre

- l'activité d'une ligne aléatoire du dataset sera prédite

Soit passer un X et un y (optionnel) de le body de la requete

- l'activité des données passées en paramètre sera prédite et la précision de la prédiction sera calculée (si y est fournit)
- Vous trouverez un exemple dans le Readme du github

Sources

https://towardsdatascience.com/productionize-a-machine-learning-model-with-a-django-api-c774cb47698c

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://seaborn.pydata.org/examples/many_pairwise_correlations.html

https://www.geeksforgeeks.org/exploring-correlation-in-python/