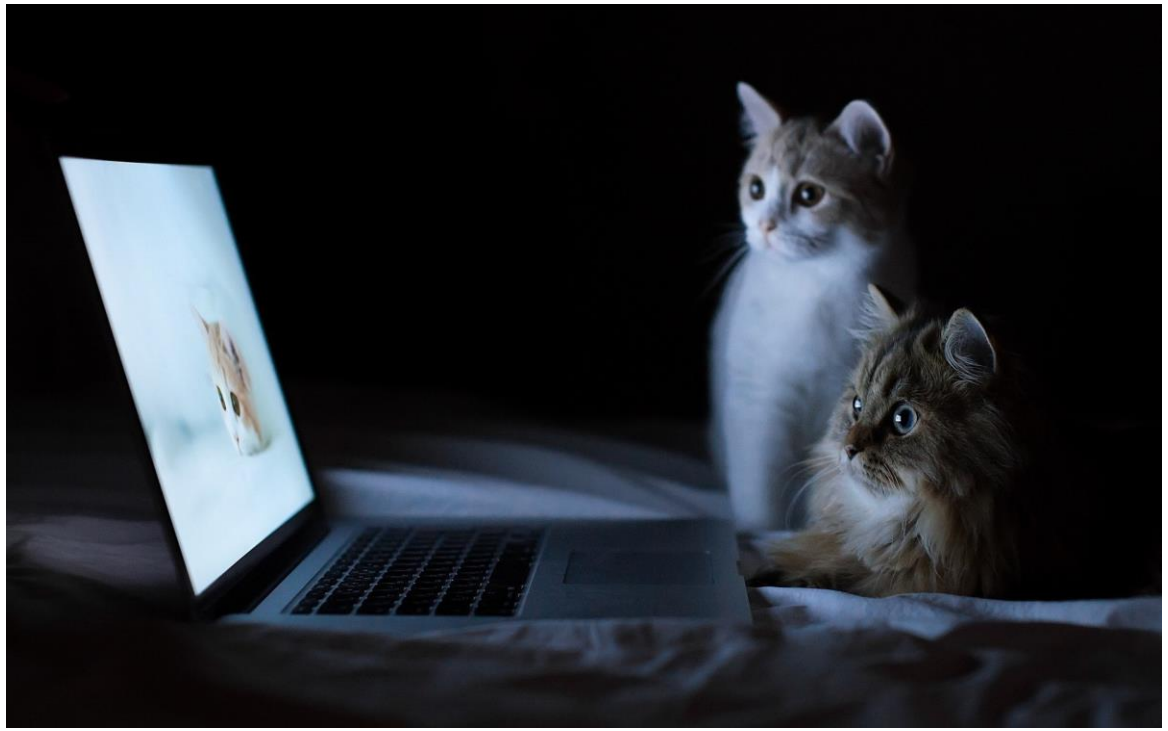
PPG-DaLiA Data Set

MERCADAL François

Plan du rapport

- Présentation de l’étude
- Présentation des variables
- Data visualisation
- Organisation des données
- Modèles créés
- API créée



Présentation de l’étude

- Contexte de l’étude : donner des estimations du rythme cardiaque des sujets suivants l’activité effectuée.

- Contexte de notre étude : pouvoir donner l’activité effectuée par le sujet à tout moment (activité déterminée par un chiffre compris entre 0 et 8)

- Situation : Classification

Présentation de l’étude

- Activités effectuées par les sujets :
  - 0 : transition
  - 1 : lecture assis
  - 2 : monter et descendre des escaliers
  - 3 : babyfoot
  - 4 : pratique du vélo
  - 5 : conduite d’une voiture
  - 6 : pause déjeuner
  - 7 : marche
  - 8 : travail sur un ordinateur

Présentation de l’étude

- Capteurs utilisés dans cette étude :
  - Capteur pectoral RespiBAN :
    - Accéléromètre sur 3 axes (700 Hz)
    - Respiration (700 Hz)
    - Electrocardiogramme (700 Hz)

  - Capteur au poignet Empatica E4 :
    - Accéléromètre sur 3 axes (32 Hz)
    - Photoplethysmographie (4 Hz)
    - Conductance (64 Hz)
    - Température (4Hz)

Présentation des variables

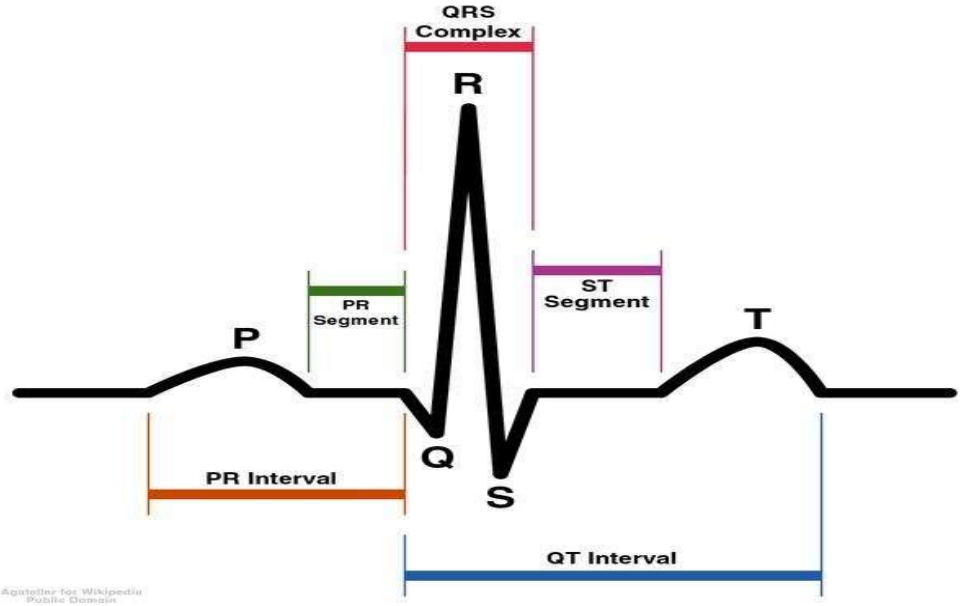
- Pour ce travail, toutes les variables issues des capteurs ont été ramenées à 4 Hz dans un soucis de calibration des données entre elles et avec la variable Activity qui est la feature à prédire.

Présentation des variables

- Variables extraites des fichiers .pkl :
  - Données uniques par sujet :
    - Genre
    - Age
    - Taille
    - Masse
    - Type de peau
    - Niveau en sport
  - Données multiples :
    - chest_ACC_X : position du RespiBAN sur l'axe x
    - chest_ACC_Y : position du RespiBAN sur l'axe y
    - chest_ACC_Z : position du RespiBAN sur l'axe z

Présentation des variables

- Variables extraites des fichiers .pkl (suite):
  - Données multiples (suite):
    - chest_ECG : Donnée issue du RespiBAN
    - chest_resp : Donnée issue du RespiBAN
    - rpeaks : liste des indices des pics R de l'électrocardiogramme captés pendant l'étude du sujet donné (dans un électrocardiogramme, le pic R correspond à la dépolarisation maximale en valeur absolue du ventricule gauche). Donnée issue du RespiBAN.



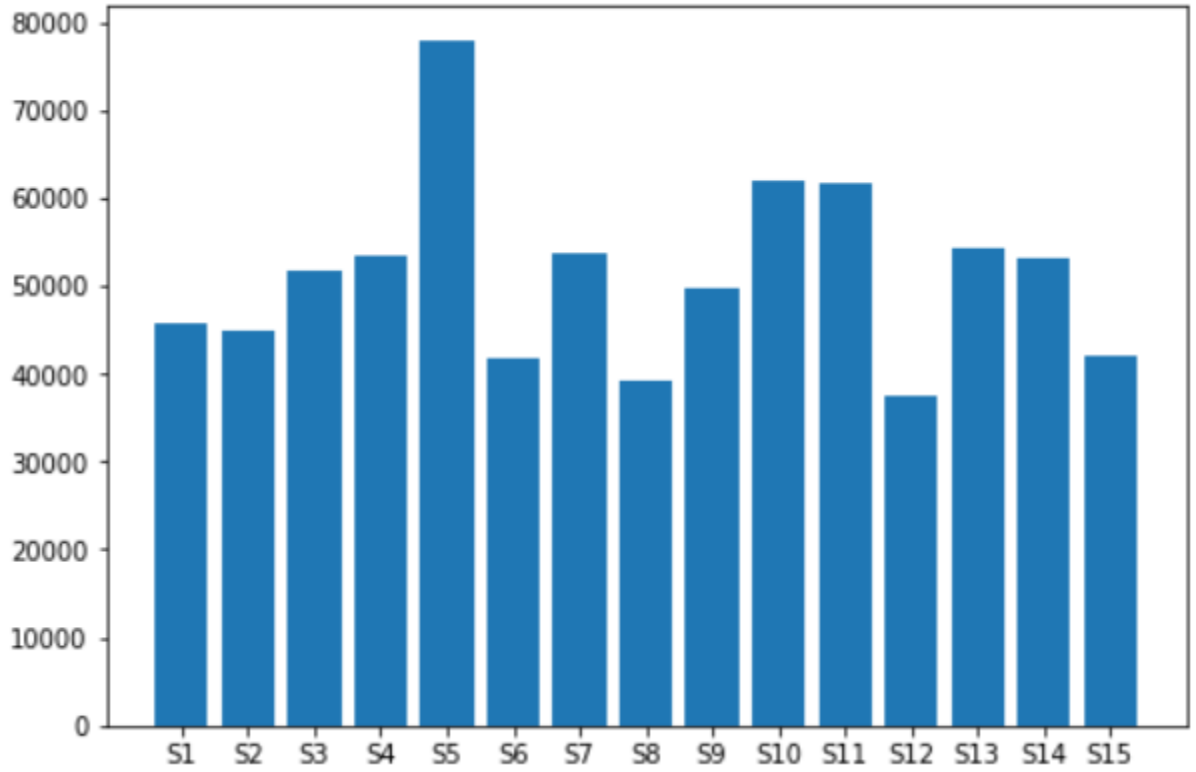
Présentation des variables

- Variables extraites des fichiers .pkl (suite):
  - Données multiples (suite):
    - wrist_ACC_X : position du Empatica4 sur l'axe x
    - wrist_ACC_Y : position du Empatica4 sur l'axe y
    - wrist_ACC_Z : position du Empatica4 sur l'axe z
    - wrist_BVP : Photoplethysmographie captée par le Empatica4
    - wrist_EDA : conductance captée par le Empatica4
    - wrist_TEMP : température captée par le Empatica4

Présentation des variables

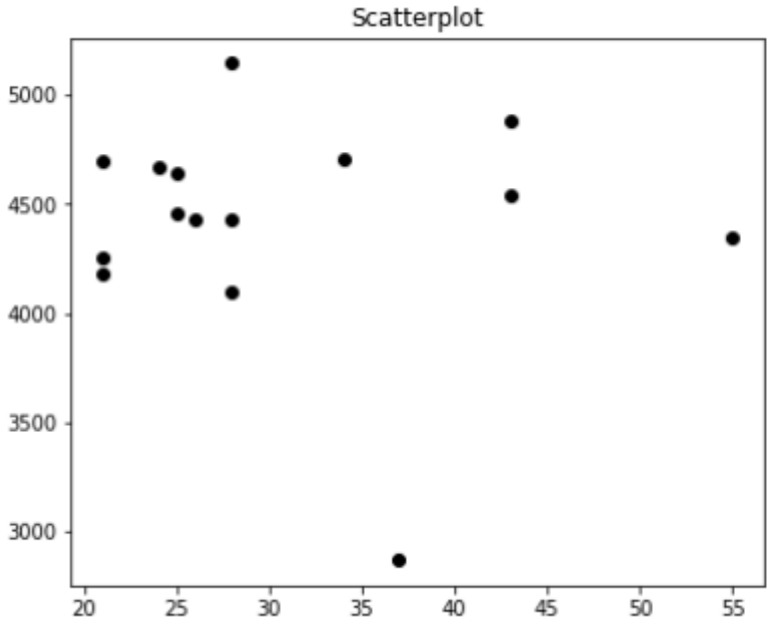
- Variables créées à partir des données extraites
  - Pour le RespiBAN :
    - SUM_chest_ACC_X : somme des mouvements du sujet sur l'axe x
    - SUM_chest_ACC_Y : somme des mouvements du sujet sur l'axe y
    - SUM_chest_ACC_Z : somme des mouvements du sujet sur l'axe z
    - rpeaks_per_second_4Hz : nombre de pics R pour chaque seconde de l'expérience
    - rpeaks_4Hz_Weight : poids accordé à chaque valeur de rpeaks_per_second_4Hz
  - Pour le Empatica4 :
    - SUM_wrist_ACC_X : somme des mouvements du sujet sur l'axe x
    - SUM_wrist_ACC_Y : somme des mouvements du sujet sur l'axe y
    - SUM_wrist_ACC_Z : somme des mouvements du sujet sur l'axe z

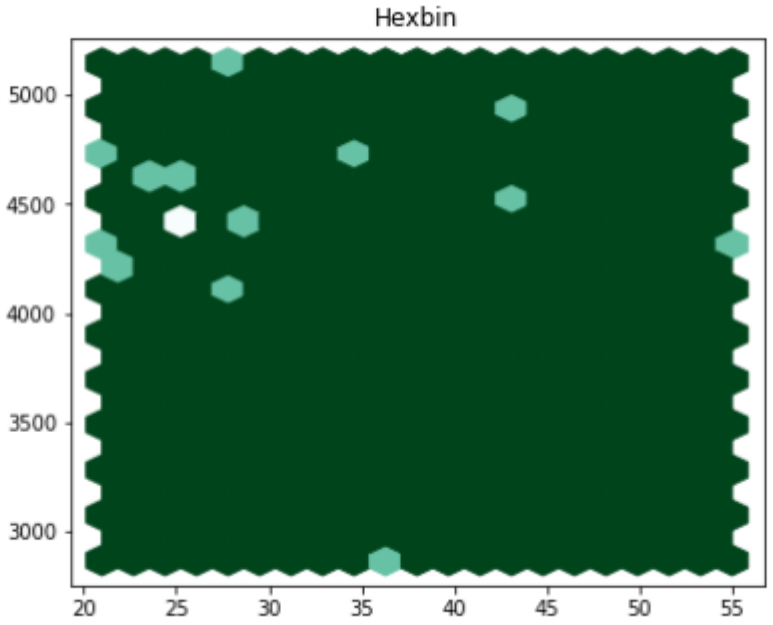
Data Visualisation

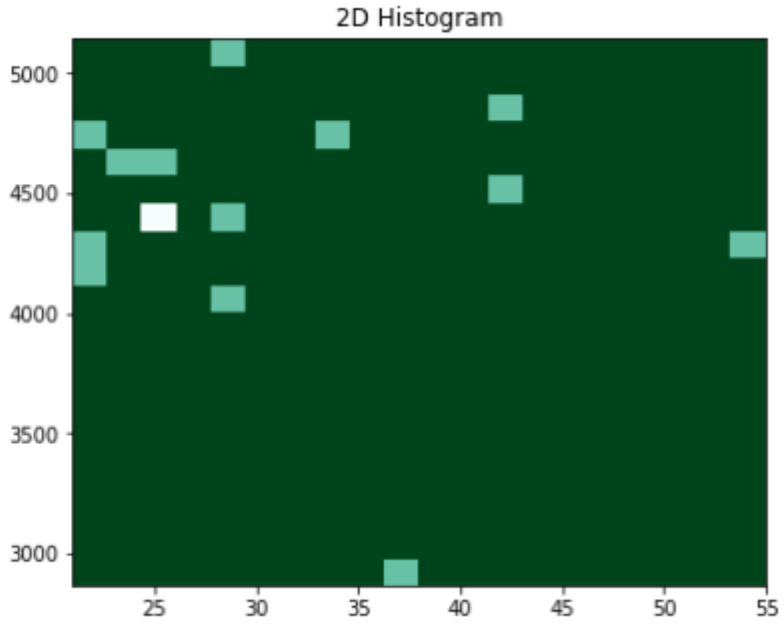


Barplot représentant le nombre total de pics R sur toute l'expérience pour chaque sujet
Cette représentation permet de noter la diversité des profils au sein des sujets

Data Visualisation

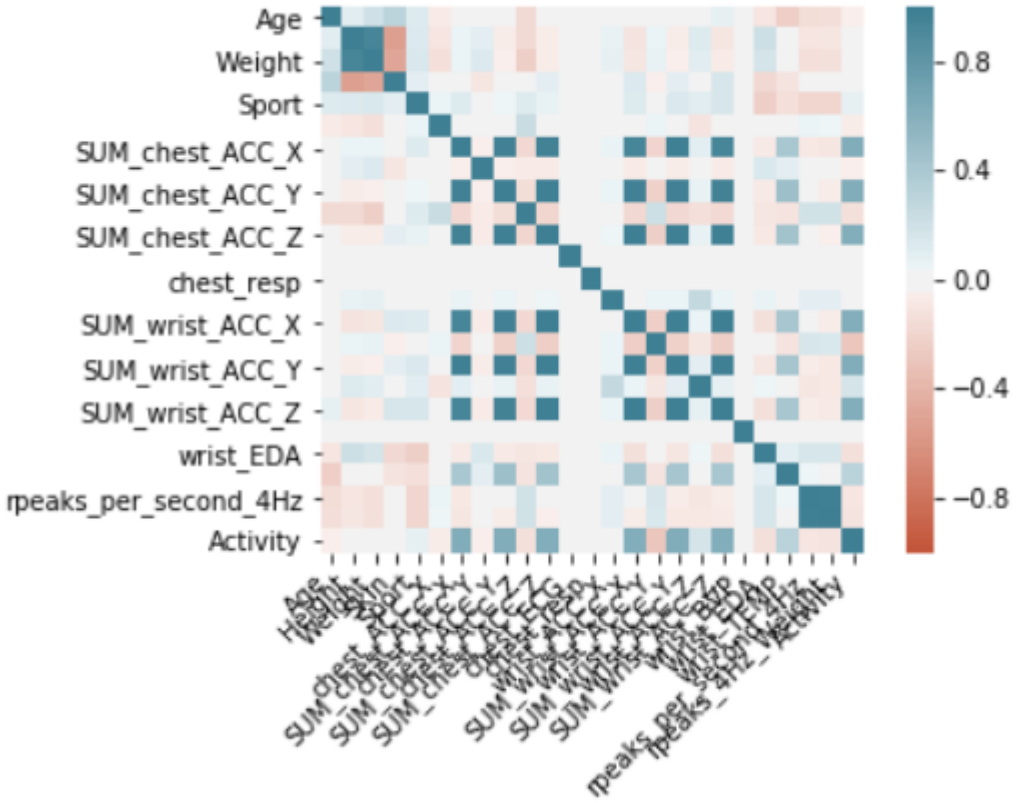






Ensemble de scatterplots mettant en relation l'âge des sujets à la quantité de mouvement du RespiBAN sur l'axe x.
On remarque que l'âge n'a pas d'influence sur les mouvements des sujets.

Data Visualisation



Matrice de corrélation des variables utilisées dans les modèles

Organisation des données

- Pour créer les modèles, les données ont été rassemblées dans un dataframe

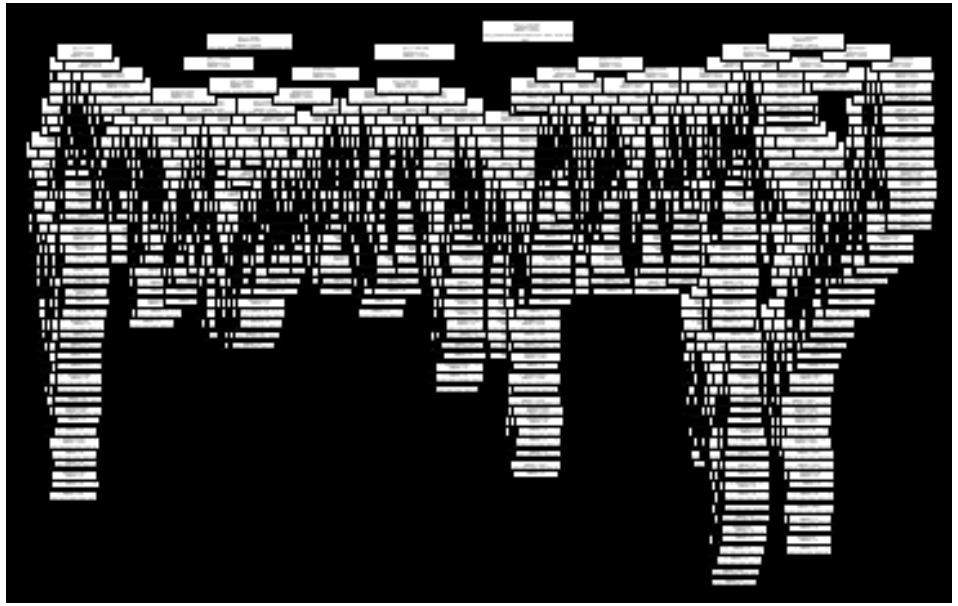
- Les données ont été séparées avec 80% des données pour l'entraînement et 20% des données pour les tests

Split Datas

```
X_train, X_testing, y_train, y_testing = train_test_split(X, y, test_size=0.20, random_state=42)
```

Modèles créés

- Decision Tree
  - Arbre de décision de classification



Arbre obtenu

Modèles créés

- Decision Tree (suite)
  - Précision obtenue

```
Entrée [68]: tree_prediction = dec_tree.predict(X_testing)
             tree_accuracy = accuracy_score(y_testing, tree_prediction)
             tree_accuracy

 Out[68]: 0.9941597806780447
```

  - Rappel obtenu

```
Entrée [73]: tree_recall = recall_score(y_testing, tree_prediction, average='macro')
             tree_recall

 Out[73]: 0.9939843563473727
```

Modèles créés

- Decision Tree
  - Matrice de consfusion

```
Entrée [74]: confusion_matrix(y_testing, tree_prediction)

Out[74]: array([[27550,    28,     9,    35,    89,    56,    25,    18,    38],
                [   16,  7316,     0,     0,     0,     0,     0,     0,     0],
                [   11,     0,  5226,     4,     0,     0,     0,     0,     0],
                [   35,     0,     4,  3555,     0,     0,     0,     0,     0],
                [   78,     0,     0,     0,  5542,     1,     0,     0,     0],
                [   66,     0,     0,     0,     0, 11041,     0,     0,     0],
                [   20,     0,     0,     0,     0,     0, 21659,    15,     0],
                [    9,     0,     0,     0,     0,     0,    14,  7425,     6],
                [   27,     0,     0,     0,     0,     0,     0,     1, 13673]],
               dtype=int64)
```

Modèles créés

- Random Forest
  - Création d'un RandomForestClassifier

```
Entrée [20]: rf = RandomForestClassifier(random_state=42, n_estimators=100)
             rf.fit(X_train, y_train)

Out[20]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=42, verbose=0,
                                warm_start=False)
```

Modèles créés

- Random Forest
  - Précision obtenue

```
Entrée [21]: prediction = rf.predict(X_testing)

Entrée [22]: accuracy = accuracy_score(y_testing, prediction)
             accuracy

    Out[22]: 0.9992373928488686
```

  - Rappel obtenu

```
Entrée [75]: forest_recall = recall_score(y_testing, prediction, average='macro')
             forest_recall

    Out[75]: 0.9991658610056393
```

Modèles créés

- Random Forest
  - Matrice de confusion

```
Entrée [28]: confusion_matrix(y_testing, prediction)

Out[28]: array([[27807,     3,     4,     2,    11,     4,     9,     6,     2],
                [    4,  7328,     0,     0,     0,     0,     0,     0,     0],
                [    4,     0,  5237,     0,     0,     0,     0,     0,     0],
                [    5,     0,     0,  3589,     0,     0,     0,     0,     0],
                [   14,     0,     0,     0,  5607,     0,     0,     0,     0],
                [    2,     0,     0,     0,     0, 11105,     0,     0,     0],
                [    2,     0,     0,     0,     0,     0, 21692,     0,     0],
                [    1,     0,     0,     0,     0,     0,     0,  7453,     0],
                [    6,     0,     0,     0,     0,     0,     0,     0, 13695]],
               dtype=int64)
```

Modèles créés

- XGBoost
  - Création d'un XGBoost Classifier

```
Entrée [79]: xg_class = XGBClassifier(objective ='reg:linear', eta = 0.3,
                                      colsample_bytree = 0.3, learning_rate = 0.1,
                                      max_depth = 10, alpha = 10, n_estimators = 100,
                                      subsample=0.7)
             xg_class.fit(X_train,y_train)

  Out[79]: XGBClassifier(alpha=10, base_score=0.5, booster='gbtree', colsample_bylevel=1,
                         colsample_bynode=1, colsample_bytree=0.3, eta=0.3, gamma=0,
                         learning_rate=0.1, max_delta_step=0, max_depth=10,
                         min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                         nthread=None, objective='multi:softprob', random_state=0,
                         reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                         silent=None, subsample=0.7, verbosity=1)
```

Modèles créés

- XGBoost
  - Précision obtenue

```
Entrée [81]: accuracy_xgb = accuracy_score(y_testing, prediction_xgb)
             accuracy_xgb

    Out[81]: 0.9992953123793343
```

  - Rappel obtenu

```
Entrée [82]: xgb_recall = recall_score(y_testing, prediction_xgb, average='macro')
             xgb_recall

    Out[82]: 0.9992932864082934
```

Modèles créés

- XGBoost
  - Matrice de confusion

```
Entrée [83]: confusion_matrix(y_testing, prediction_xgb)

    Out[83]: array([[27805,     3,     6,     2,     6,     4,    11,     9,     2],
                    [    2,  7330,     0,     0,     0,     0,     0,     0,     0],
                    [    6,     0,  5235,     0,     0,     0,     0,     0,     0],
                    [    7,     0,     0,  3587,     0,     0,     0,     0,     0],
                    [    3,     0,     0,     0,  5618,     0,     0,     0,     0],
                    [    2,     0,     0,     0,     0, 11105,     0,     0,     0],
                    [    2,     0,     0,     0,     0,     0, 21692,     0,     0],
                    [    1,     0,     0,     0,     0,     0,     0,  7453,     0],
                    [    7,     0,     0,     0,     0,     0,     0,     0, 13694]],
                   dtype=int64)
```

API créée

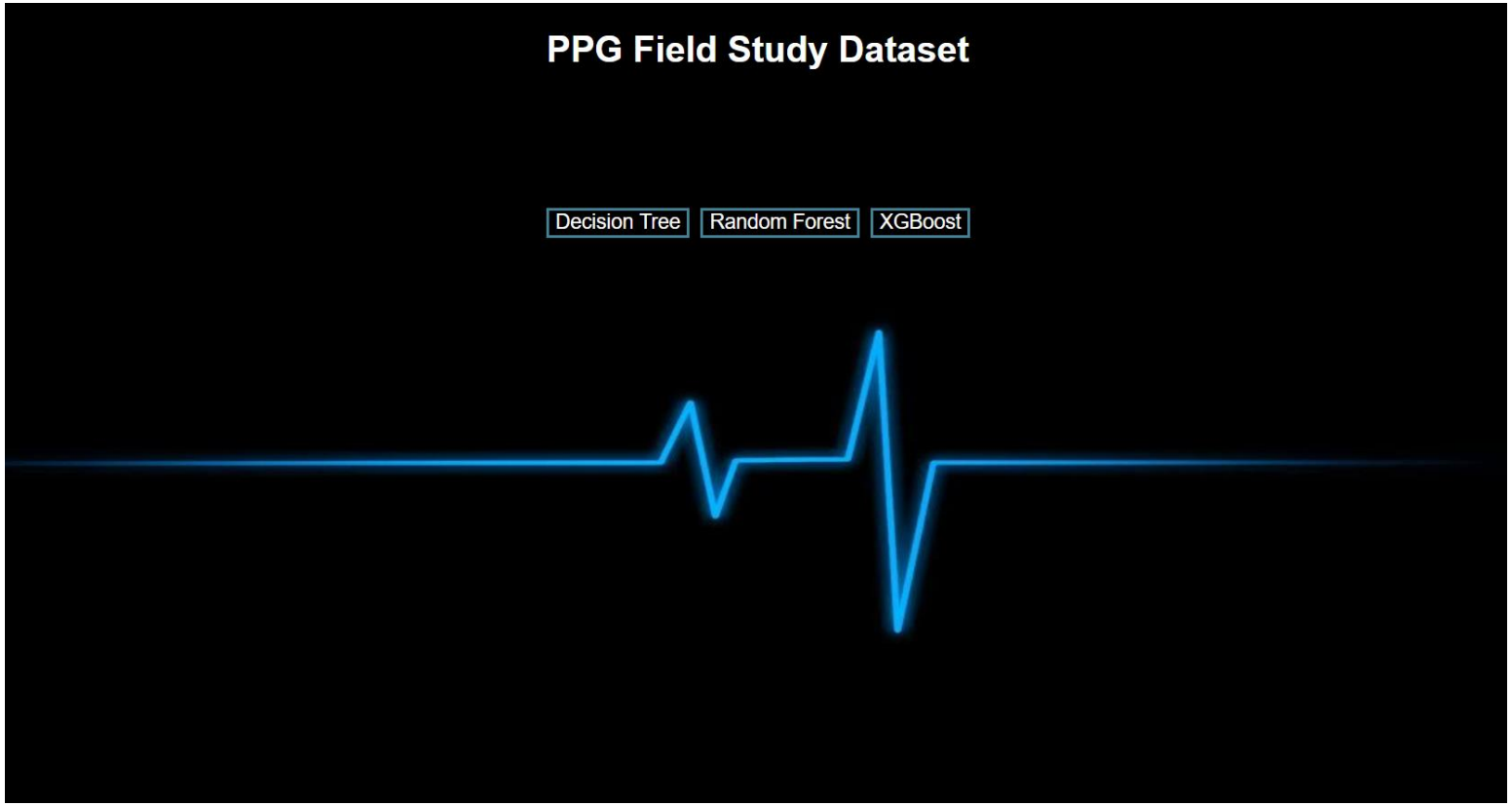
- Création d’une API avec Django pour mettre en forme les résultats

- L’API s’organise en 4 interfaces html :
  - Une interface d’accueil
  - Une interface pour la prédiction du Decision Tree
  - Une interface pour la prédiction du Random Forest
  - Une interface pour la prédiction du XGBoost

- Pour lancer l’API, entrer : python manage.py runserver

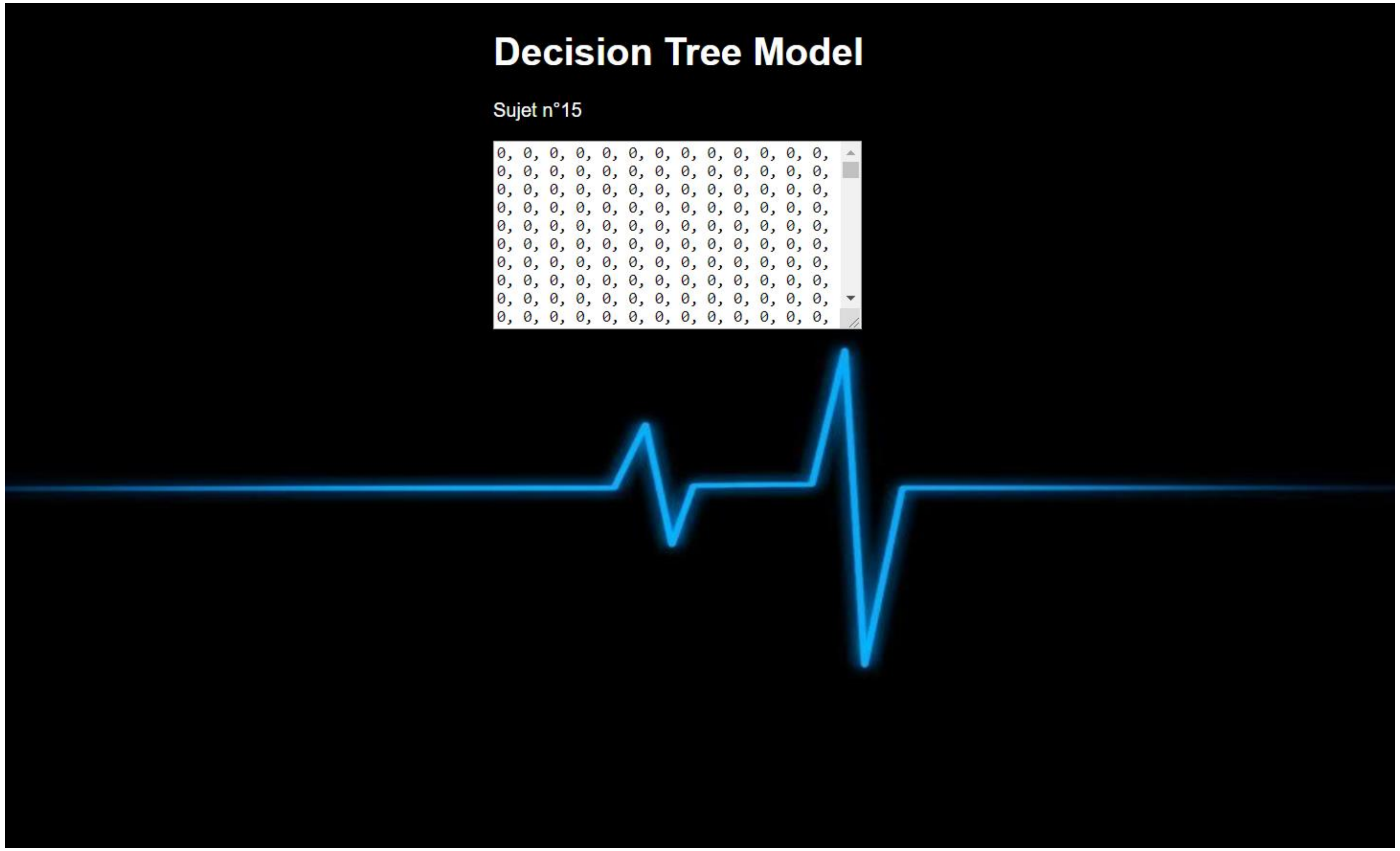
API créée

- http://127.0.0.1:8000/accueil



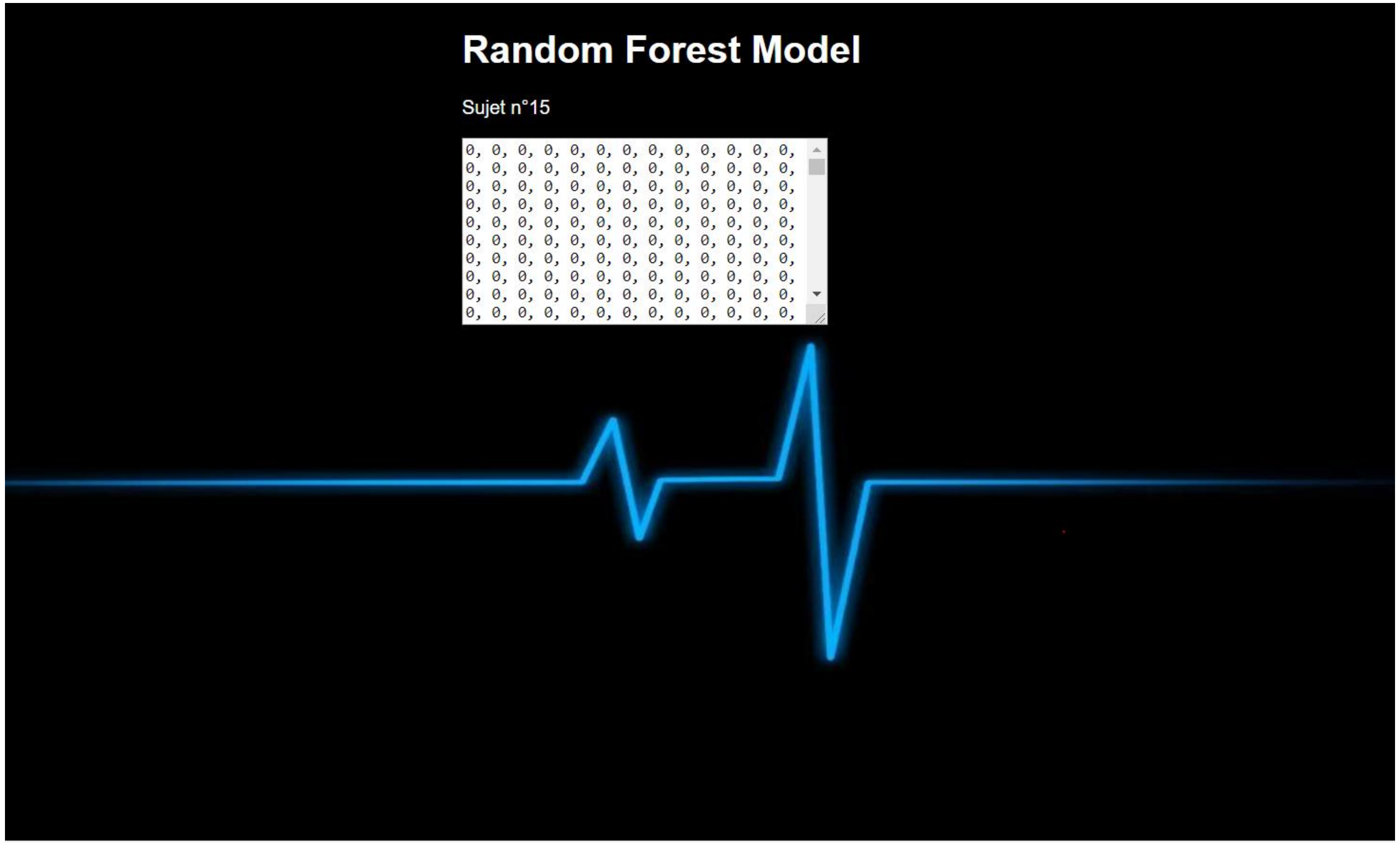
API créée

- http://127.0.0.1:8000/decision_tree



API créée

- http://127.0.0.1:8000/random_forest



API créée

- http://127.0.0.1:8000/xgboost