

Titre professionnel

**Développeur en intelligence
artificielle**

RNCP 34757

**Rapport E1 compétences C1 à C7,
C9 à C13, C15 à C18**

**Prévention des maladies
cardiovasculaires**

Table des matières

1 Présentation du projet.....	3
a Contexte du projet.....	3
2 Qualification des données.....	4
a Étude du rapport des données.....	4
i Rapport généré par ProfileReport.....	4
ii Age et genre.....	4
iii Taille poids et IMC.....	4
iv Pression systolique et diastolique.....	6
v Autres colonnes.....	6
vi Recherche de doublons.....	6
b Conception d'une base de données analytique à partir du dataset filtré.....	7
c Conception d'une base de données de production.....	8
i Création et remplissage des tables.....	9
3 Choix et entraînement d'un modèle.....	10
a Chargement des données depuis une base SQL.....	10
b Analyse et préparation des données.....	11
i Variable cible.....	11
ii Variables qualitatives.....	11
iii Variables quantitatives.....	12
iv Visualisation de la répartition des données.....	13
c Test d'algorithmes.....	13
i Optimisation des algorithmes.....	15
ii Le voting classifier.....	15
iii Technique de stacking.....	16
iv Un mot sur la technique de boosting.....	16
v Un mot sur la technique de bagging.....	16
d Conclusion.....	16
4 Réalisation d'un site web.....	17
a Maquette du site.....	18
b Cahier des charges :	18
c Utilisation du programme.....	19
d Effectuer une prédiction.....	21
e La fonction alerte.....	22
5 Gestion de projet.....	22
a Gestion de projet dans l'entreprise :	22
b Gestion de projet méthode "agile".....	23
6 Annexes.....	23
a Jupyter notebook :	23
b Programme python de l'application web :	23
c Plan du site web de l'application.....	24
d Utilitaire de création de base et d'utilisateurs.....	24
e Gestion de projet – méthode "agile".....	24

1 Présentation du projet.

Ce projet reprend le brief 12 dont voici l'énoncé :

Vous travaillez dans le domaine de la médecine préventive. Votre métier est donc de donner des conseils d'hygiène de vie (propreté, mais aussi diététique, encouragement à un sport ou une activité physique, ergonomie et manière de faire des efforts, prévention des comportements à risque, etc.) ainsi que de proposer un accompagnement dans le dépistage de maladies et plus spécifiquement dans la prévention des risques cardiovasculaires.

a Contexte du projet

Entre 300 000 et 400 000 accidents cardiovasculaires surviennent chaque année en France, dont un tiers sont mortels. Comment mieux prédire le risque cardiovasculaire ? Si plusieurs facteurs de risque sont identifiés, quelles sont les interactions entre ces facteurs ? Les maladies cardiovasculaires, principalement les accidents vasculaires cérébraux (AVC) et les infarctus du myocarde, sont la deuxième cause de mortalité en France. La liste des facteurs de risque cardiovasculaire est malheureusement longue. Les 12 facteurs de risque constituent ainsi un véritable réseau de facteurs de risque cardiovasculaire. En fonction de ces interactions, des chercheurs ont pu mettre en évidence 4 groupes de facteurs de risque : Des «facteurs non modifiables» (le sexe, l'âge et les antécédents familiaux) : ils prédisent d'autres facteurs, mais ne peuvent pas être prédits par d'autres facteurs. Des «facteurs liés au mode de vie» (le tabagisme, la sédentarité, l'alcoolisme) : ils prédisent beaucoup d'autres facteurs (sauf les facteurs non modifiables), mais sont très peu prédits par d'autres facteurs. Des «facteurs cliniques en amont» (les troubles du sommeil, l'obésité, la dépression) : ils prédisent beaucoup d'autres facteurs et sont eux-mêmes prédits par de nombreux facteurs. Des «facteurs cliniques en aval» (l'hypertension artérielle, les dyslipidémies, le diabète) : ils prédisent très peu de facteurs, mais sont en revanche prédits par beaucoup de facteurs.

Pour vous accompagner au mieux dans votre démarche de prévention de ces risques cardiovasculaires, vous avez décidé de développer un outil permettant de poser un diagnostic rapide de risques cardiovasculaire. Cet outil mettra en œuvre un algorithme de machine learning (de classification binaire : prédition binaire : 0 ou 1) permettant de prédire s'il y a un risque cardiovasculaire ou s'il n'y en a pas.

Présentation des données : Pour pouvoir entraîner votre algorithme, vous avez monté un partenariat avec des médecins généralistes de votre ville et ainsi pu récolter des données de patients. Ces données sont stockées dans un fichier .csv.

Ce fichier comporte 12 colonnes :

- AGE : integer (number of days)
- HEIGHT : integer (cm)
- WEIGHT : integer (kg)
- GENDER : categorical (1: female, 2: male)
- AP_HIGH : systolic blood pressure, integer
- AP_LOW : diastolic blood pressure, integer
- CHOLESTEROL : categorical (1: normal, 2: above normal, 3: well above normal)
- GLUCOSE : categorical (1: normal, 2: above normal, 3: well above normal)
- SMOKE : categorical (0: no, 1: yes)

- ALCOHOL : categorical (0: no, 1: yes)
- PHYSICAL_ACTIVITY : categorical (0: no, 1: yes)

Et la variable cible :

- CARDIO_DISEASE : categorical (0: no, 1: yes)

2 Qualification des données.

Référentiel : C1. Qualifier les données grâce à des outils d'analyse et de visualisation de données en vue de vérifier leur adéquation avec le projet.

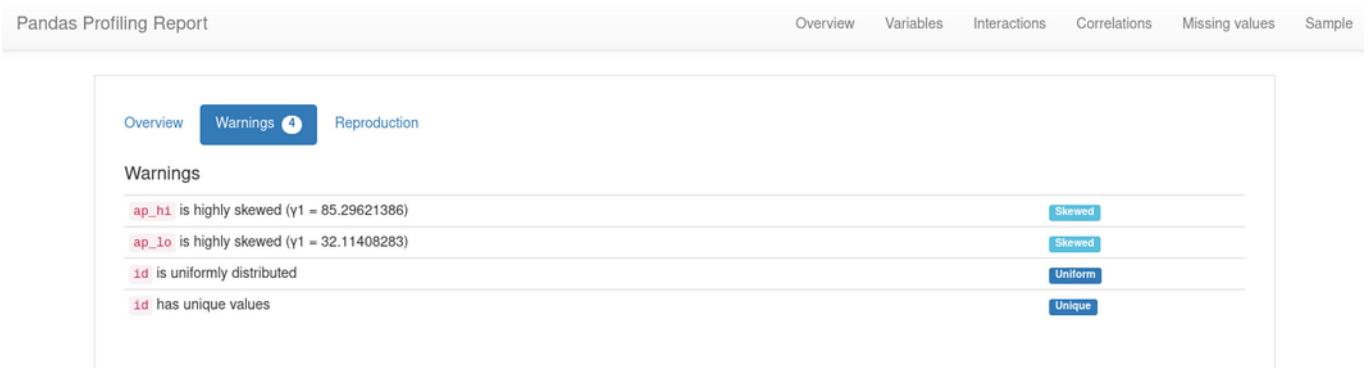
Référentiel : C3. Programmer l'import de données initiales nécessaires au projet en base de données, afin de les rendre exploitable par un tiers, dans un langage de programmation adapté et à partir de la stratégie de nettoyage des données préalablement définie.

Le dataset est livré sous forme d'un fichier csv et, pour en étudier le contenu on va supprimer la colonne id (qui nous servira plus tard de clé primaire) et rajouter une colonne IMC pour mieux traquer les valeurs aberrantes concernant le rapport taille/poids. Nous utiliserons ensuite la méthode de pandas "ProfileReport" et nous générerons le rapport sous forme "html".

a Étude du rapport des données.

i Rapport généré par ProfileReport.

Ce rapport est sauvegardé dans le fichier "cardio.html" ; 4 avertissements sont émis :



Les avertissements concernant les anomalies sur les pressions artérielles sont expliqués plus bas. Le rapport nous indique que l'id est unique ce qui nous assure, en principe, une absence de doublon. Ceci étant, une petite vérification sera faite.

ii Age et genre.

La première chose à noter est l'âge exprimé en jour et correspondant à une plage allant de 29 à 64 ans. Il n'y a pas d'anomalie visible dans cette colonne.

La deuxième colonne est le genre. Comme on peut le voir il y a majoritairement des hommes (45 530) contre 24 470 femmes. Ce déséquilibre peut être source de problèmes surtout si le genre est une variable importante dans la prédiction.

iii Taille poids et IMC.

La taille et le poids des 2 colonnes suivantes donnent un minimum et un maximum de 50 à 250 cm et 10 à 200 kg. Sans être impossible, ces valeurs méritent d'être mieux étudiées pour savoir s'il y a, ou non, des valeurs aberrantes. Pour cela nous avons créé l'IMC qui donne des valeurs comprises entre 3,47 et 298 !

On rappelle que selon wikipédia la signification de l'IMC est la suivante :

Interprétation de l'IMC (kg m ⁻²)	Interprétation
moins de 16,5	dénutrition
16,5 à 18,5	maigreur
18,5 à 25	poids normal
25 à 30	surpoids
30 à 35	obésité modérée
35 à 40	obésité sévère
40 et plus	obésité morbide ou massive

Avec une répartition dans la population qui est :

IMC dans la population française

IMC (kg m ⁻²)	Proportions						
	1997	2000	2003	2006	2009	2012	2020
- de 18,5	4,2 %	3,8 %	3,9 %	3,9 %	3,6 %	3,5 %	4,5 %
18,5 à 24,9	57,5 %	55,5 %	52,7 %	52,4 %	50,0 %	49,2 %	48,2 %
25 à 29,9	29,8 %	30,6 %	31,5 %	30,6 %	31,9 %	32,3 %	30,3 %
30 à 39,9	8,2 %	9,7 %	11,2 %	12,3 %	13,4 %	13,8 %	15,0 %
Plus de 40	0,3 %	0,4 %	0,7 %	0,8 %	1,1 %	1,2 %	2,0 %

Au vu des données de ces deux tableaux et de ce que l'on sait sur les risques de maladies cardiovasculaires liés au sur/sous poids on recherche dans le dataset les valeurs aberrantes avec les critères suivants : IMC > 40 ou < 15 sans activité physique et avec risque cardiovasculaire nul. A l'évidence, il s'agit de données présentant au moins une erreur de saisie. Voici les résultats :

```
1 df[ (df.IMC>40)&(df.cardio== 0)&(df.active== 0) ]
```

		age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	IMC
435	16765	1	186	200.0	130	70		1	1	0	0	0	0	57.810151
591	14705	1	164	125.0	130	90		1	1	0	0	0	0	46.475312
992	21897	1	154	110.0	110	70		1	1	0	0	0	0	46.382189
1230	15126	2	173	125.0	120	80		1	1	1	0	0	0	41.765512
1399	22644	1	166	131.0	120	80		3	3	0	0	0	0	47.539556
...
61503	17375	1	162	110.0	110	60		1	3	0	0	0	0	41.914342
62938	17325	1	165	115.0	120	80		1	1	0	0	0	0	42.240588
64115	18426	1	59	57.6	125	67		1	1	0	0	0	0	165.469693
66751	21182	1	157	126.0	140	90		3	3	0	0	0	0	51.117692
67887	21199	1	159	112.0	130	90		1	3	0	0	0	0	44.302045

113 rows × 13 columns

```
1 df[ (df.IMC<15)&(df.cardio== 0)&(df.active== 0) ]
```

		age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	IMC
22307	21228	1	169	42.0	150	90		2	2	0	0	0	0	14.705367
63154	22192	2	198	58.0	110	70		1	1	0	0	0	0	14.794409

On décidera de supprimer ces données.

iv Pression systolique et diastolique.

Viennent ensuite les pressions artérielles systoliques et diastoliques dont le rapport généré par ProfilReport nous indique qu'elles sont "skewed", c'est à dire comportant un biais. En effet, les valeurs vont de -150 à 16020 pour l'une et de -70 à 11000 pour l'autre et ces valeurs extrêmes sont très rares (d'où le biais détecté par ProfilReport).

L'hypertension sévère étant au dessus de 200, on ne gardera que les valeurs comprises entre 5 et 300 pour la première (49 lignes supprimées), et entre 2 et 300 pour la seconde. Inutile de préciser que les valeurs négatives n'ont aucun sens ! Un professionnel de santé aurait dû être consulté pour une meilleure analyse.

```
1 df[ (df.ap_hi<5)|(df.ap_hi>300) ]
```

		age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	IMC
1876	15116	1	160	60.0	902	60		1	1	0	0	1	0	23.437500
2014	22712	2	167	59.0	906	0		1	1	0	0	1	0	21.155294
4607	15281	1	165	78.0	-100	80		2	1	0	0	1	0	28.650138
4817	14425	1	168	63.0	909	60		2	1	0	0	1	0	22.321429

```
1 df[ (df.ap_lo<2)|(df.ap_lo>300) ]
```

		age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	IMC
228	17489	2	183	98.0	160	1100		1	2	1	0	1	1	29.263340
241	21932	2	157	60.0	160	1000		2	1	0	0	0	1	24.341758
260	18217	1	150	83.0	140	800		1	1	0	0	1	1	36.888889
329	23407	1	176	63.0	160	1000		2	2	0	0	0	1	20.338326
345	18704	1	154	81.0	140	1000		2	1	0	0	1	1	34.154158
...
69771	23330	1	167	81.0	160	1000		1	1	0	0	1	1	29.043709
69872	21808	1	152	56.0	160	1000		1	1	0	0	1	1	24.238227
69878	21239	2	168	95.0	160	1000		1	1	0	0	1	1	33.659297
69885	22417	2	166	78.0	170	1000		1	1	0	0	0	0	28.305995
69967	21416	2	168	63.0	140	1000		1	1	0	0	1	1	22.321429

976 rows × 13 columns

v Autres colonnes.

Les autres colonnes ne présentent pas de problèmes particuliers et sont sous forme de classe et non de chiffres représentants des grandeurs réelles. S'il y a eu des erreurs de saisie, elles ne sont pas détectables.

On notera enfin que les données ne sont pas également réparties (très peu de fumeurs, de consommateurs d'alcool, d'inactifs,...). Dans le dataset, par contre, pour ce qui est de la variable cible, les valeurs sont comparables. Attention donc aux corrélations qui pourraient exister sans être très visibles.

vi Recherche de doublons.

En utilisant la méthode "duplicated" de pandas on s'aperçoit qu'il existe bel et bien des doublons ! 24 pour être précis.

```
: 1 len(df[df.duplicated(subset=['age', 'gender','height', 'weight', 'ap_hi', 'ap_lo',
2                               'cholesterol','gluc', 'smoke', 'alco', 'active','cardio'])]== True])
```

24

En voici un échantillon :

```
1 df[df.duplicated(subset=['age', 'gender','height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol','gluc',
2                           'smoke', 'alco', 'active','cardio'], keep= False)== True].sort_values(by = ['age'])
```

		age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	IMC
	id													
57690	14552	1	158	64.0	120	80		1	1	0	0	1	0	25.636917
9004	14552	1	158	64.0	120	80		1	1	0	0	1	0	25.636917
91592	16160	1	168	65.0	120	80		1	1	0	0	1	1	23.030045
24435	16160	1	168	65.0	120	80		1	1	0	0	1	1	23.030045
1685	16793	1	165	68.0	120	80		1	1	0	0	1	0	24.977043
31110	16793	1	165	68.0	120	80		1	1	0	0	1	0	24.977043
40450	16805	1	157	67.0	120	80		1	1	0	0	1	0	27.181630
86345	16805	1	157	67.0	120	80		1	1	0	0	1	0	27.181630

Bien sûr il n'est pas possible d'affirmer qu'il s'agisse de doublons. Mais l'hypothèse que deux personnes du même sexe, ayant le même âge le jour de l'examen et présentant les mêmes données cliniques est proche, en termes de vraisemblance, de l'hypothèse impliquant des marmottes dans l'emballage du chocolat dans le papier alu...

Il me semble donc juste d'éliminer ces supposés doublons même si leur présence n'aurait pas beaucoup d'impact dans l'apprentissage d'un modèle de machine learning.

```
1 # Suppression des doublons
2 df.drop_duplicates(subset= ['age', 'gender','height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol','gluc',
3 'smoke', 'alco', 'active','cardio'], keep= 'first', inplace= True)|
```

b Conception d'une base de données analytique à partir du dataset filtré.

Référentiel : C2. Concevoir une base de données analytique avec l'approche orientée requêtes en vue de la mise à disposition des données pour un traitement analytique ou d'intelligence artificielle.

```
1 bd= 'base_E1_train.db'
2 # Ne pas ré-exécuter ce code une fois la base créée
3 """
4 try:
5     # Création de la base de donnée
6     conn = sqlite3.connect(bd)
7     print(f"Création de la base {bd} réussi")
8 except sqlite3.Error as error:
9     print("Erreur lors de la connexion à SQLite", error)
10 """
```

Création de la base base_E1_train.db réussi

Cette base ne contiendra qu'une seule table contenant le jeu de données précédemment filtré.

```

1 # Ne pas ré-exécuter ce code une fois la base créée - Création de la table entrainement
2 """
3 try:
4     conn = sqlite3.connect(bd)
5     cur = conn.cursor()
6
7     sql = "CREATE TABLE entrainement ( "+\
8         "id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "+\
9         "age INTEGER NOT NULL, "+\
10        "gender INTEGER, "+\
11        "height INTEGER NOT NULL, "+\
12        "weight INTEGER NOT NULL, "+\
13        "ap_hi INTEGER NOT NULL, "+\
14        "ap_lo INTEGER NOT NULL, "+\
15        "cholesterol INTEGER, "+\
16        "gluc INTEGER, "+\
17        "smoke INTEGER, "+\
18        "alco INTEGER, "+\
19        "active INTEGER, "+\
20        "cardio INTEGER );"
21     cur.execute(sql)
22     conn.commit()
23     cur.close()
24     conn.close()
25     print("Table créée.")
26
27 except sqlite3.Error as error:
28     print("Erreur lors de la connexion à SQLite", error)
29 """

```

Table créée.

```

1 # Ne pas ré-exécuter ce code une fois la table remplie
2 """
3 try:
4     # Création de la base de donnée
5     conn = sqlite3.connect(bd)
6     cur = conn.cursor()
7
8     sql = "INSERT INTO entrainement (age, gender, height, weight, ap_hi, ap_lo, cholesterol, "+\
9         "gluc, smoke, alco, active, cardio) VALUES(?,?,?,?,?,?,?,?,?,?)"
10
11    for i, row in df.iterrows():
12        cur.execute(sql, tuple(row))
13        conn.commit()
14
15    cur.close()
16    conn.close()
17    print("La connexion SQLite est fermée")
18 except sqlite3.Error as error:
19     print("Erreur lors de la connexion à SQLite", error)
20 """

```

La connexion SQLite est fermée

c Conception d'une base de données de production.

Référentiel : C10. Concevoir une base de données relationnelle à l'aide de méthodes standards de modélisation de données.

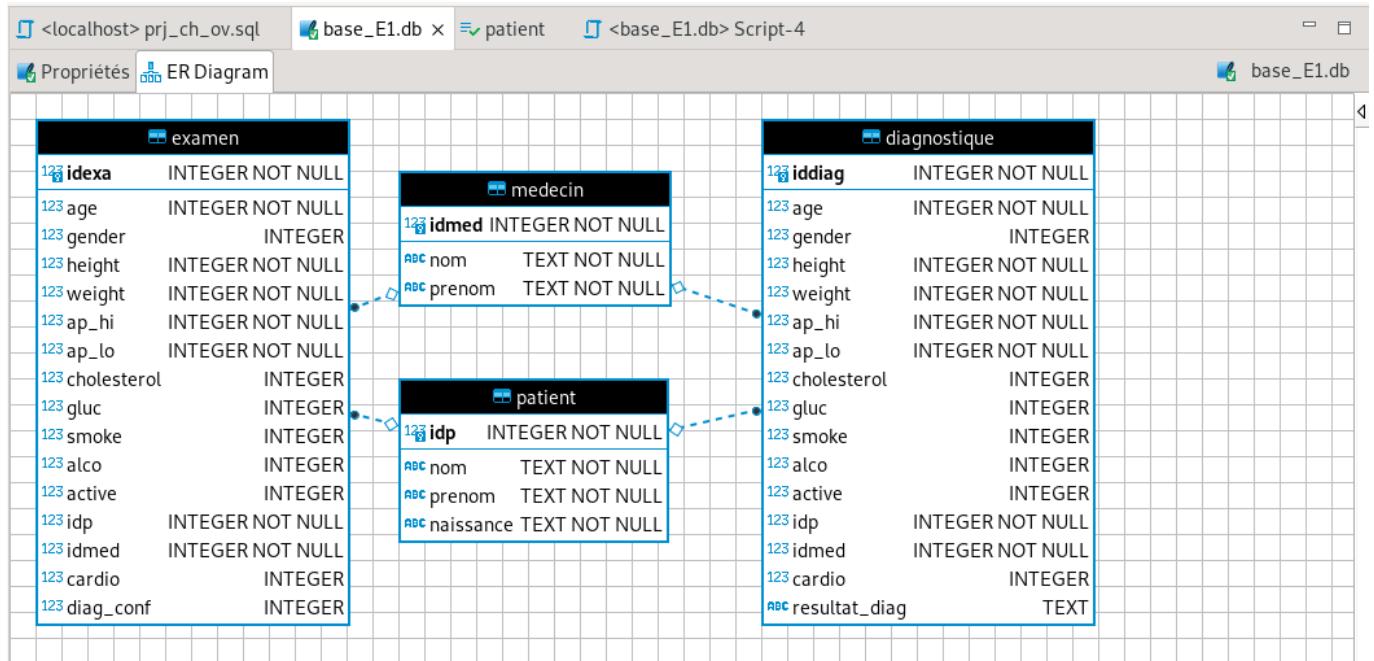
Pour rendre cette étude plus proche de la réalité, nous allons créer une base de données regroupant un certains nombres de tables :

La table patients : On va créer de toute pièce cette table à partir des datasets prénoms et patronymes que l'on trouve sur le site ".gouv" concernant le développement du machine learning. Une date de naissance sera attribuée en tenant compte de l'âge donné dans le dataset d'origine auquel on ajoute un nombre aléatoire pour plus de réalisme.

La table médecins : Idem pour cette table.

La table examen : Sera l'image de notre jeu de données en intégrant deux clés étrangères l'une étant la clé primaire de la table patient, l'autre la clé primaire de la table médecin. Une colonne supplémentaire contiendra le champs diagnostique confirmé qui sera égal à 1 si c'est le cas, 0 sinon.

La table diagnostique : Sera sur le modèle de la table examen avec un champ supplémentaire de type texte qui sera le résumé des résultats produits par le ou les algorithmes utilisés pour réaliser le diagnostique et un champ en moins, le champ diagnostique confirmé.



i Création et remplissage des tables.

Toutes les tables ont été créées et remplies en utilisant les commandes sql via la librairie sqlite3 de python (comme précédemment pour la table "entraînement" de la base base_E1_train.db).

Voici, par exemple, la création de la table examen et le remplissage de celle-ci.

```

# Ne pas ré-exécuter ce code une fois la base créée - Création de la table examen
"""
try:
    conn = sqlite3.connect(bdp)
    cur = conn.cursor()

    sql = "CREATE TABLE examen ( "+\
        "idexa INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "+\
        "age INTEGER NOT NULL, "+\
        "gender INTEGER, "+\
        "height INTEGER NOT NULL, "+\
        "weight INTEGER NOT NULL, "+\
        "ap_hi INTEGER NOT NULL, "+\
        "ap_lo INTEGER NOT NULL, "+\
        "cholesterol INTEGER, "+\
        "gluc INTEGER, "+\
        "smoke INTEGER, "+\
        "alco INTEGER, "+\
        "active INTEGER, "+\
        "idp INTEGER NOT NULL, "+\
        "idmed INTEGER NOT NULL, "+\
        "cardio INTEGER , "+\
        "diag_conf INTEGER , "+\
        "FOREIGN KEY(idp) REFERENCES patient(idp), " +\
        "FOREIGN KEY (idmed) REFERENCES medecin (idmed));" #+|"

    cur.execute(sql)
    conn.commit()
    cur.close()
    conn.close()
    print("Table examen créée.")

except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

```

Table examen créée.

```

# Ne pas ré-exécuter ce code une fois la base créée - Remplissage de la table examen
"""
try:
    # Connexion à la base de donnée
    conn = sqlite3.connect(bdp)
    cur = conn.cursor()

    sql = "INSERT INTO examen (age, gender, height, weight, ap_hi, ap_lo, cholesterol, "+\
        "gluc, smoke, alco, active, cardio, idp, idmed, diag_conf) VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
    for i, row in df_examen.iterrows():
        cur.execute(sql, tuple(row))
        conn.commit()

    cur.close()
    conn.close()
    print("Table examen remplie")
except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

```

Table examen remplie

3 Choix et entraînement d'un modèle.

a Chargement des données depuis une base SQL.

Référentiel : C4. Préparer les données disponibles depuis la base de données analytique en vue de leur utilisation par les algorithmes d'intelligence artificielle

Les données une fois filtrées ont été mises dans la table "entraînement" de la base de données base_E1_train.db. Ces données serviront à l'entraînement des différents modèles que nous allons tester.

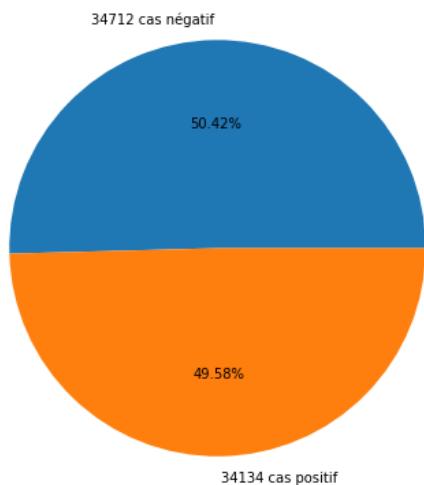
```
1 bdd= 'base_E1_train.db'
2
3 conn = sqlite3.connect(bdd)
4
5 dfbase= pd.read_sql_query("SELECT * FROM entraînement", conn)
6
7 conn.close()
8
9 # La colonne id ne servant pas, on la supprime
10 dfbase.drop(["id"], axis= 1, inplace= True)
11
```

b Analyse et préparation des données.

i Variable cible.

Au passage, on regarde la répartition cas positifs / cas négatifs. Cette répartition est tout à fait équilibrée.

```
1 # Répartition du dataset
2 ze= dfbase["cardio"][dfbase["cardio"]== 0].count()
3 un= dfbase["cardio"][dfbase["cardio"]== 1].count()
4 plt.figure(figsize=(7,7))
5 plt.pie([ze,un], labels= [f"{ze} cas négatif",f"{un} cas positif"], autopct= '%1.2f%%')
6 plt.show()
```



ii Variables qualitatives.

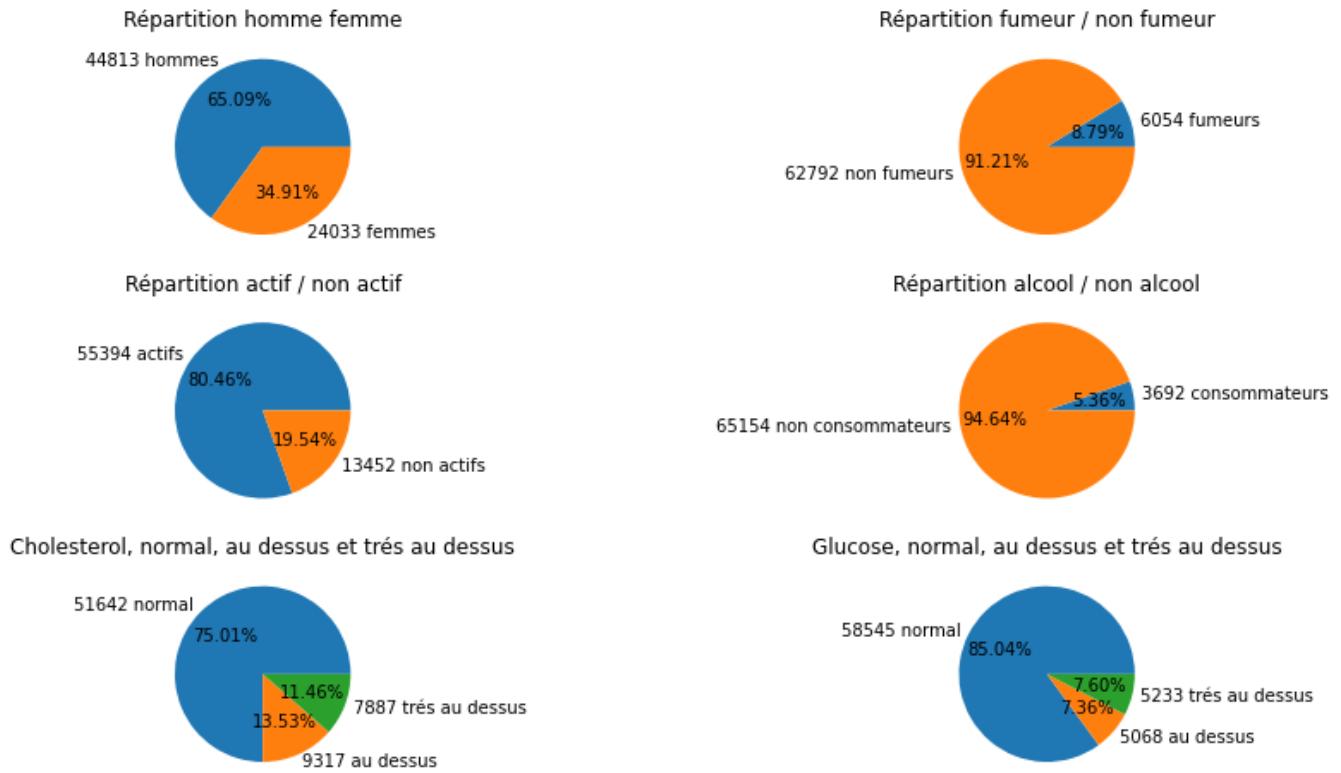
On regardera ensuite les différentes variables de classifications.

```
# Contenu des variables de classification.

for el in ("gender", "cholesterol", "gluc", "smoke", "alco", "active", "cardio"):
    print(f"Valeur possible de la colonne '{el}': {np.sort(df[el].unique())}")

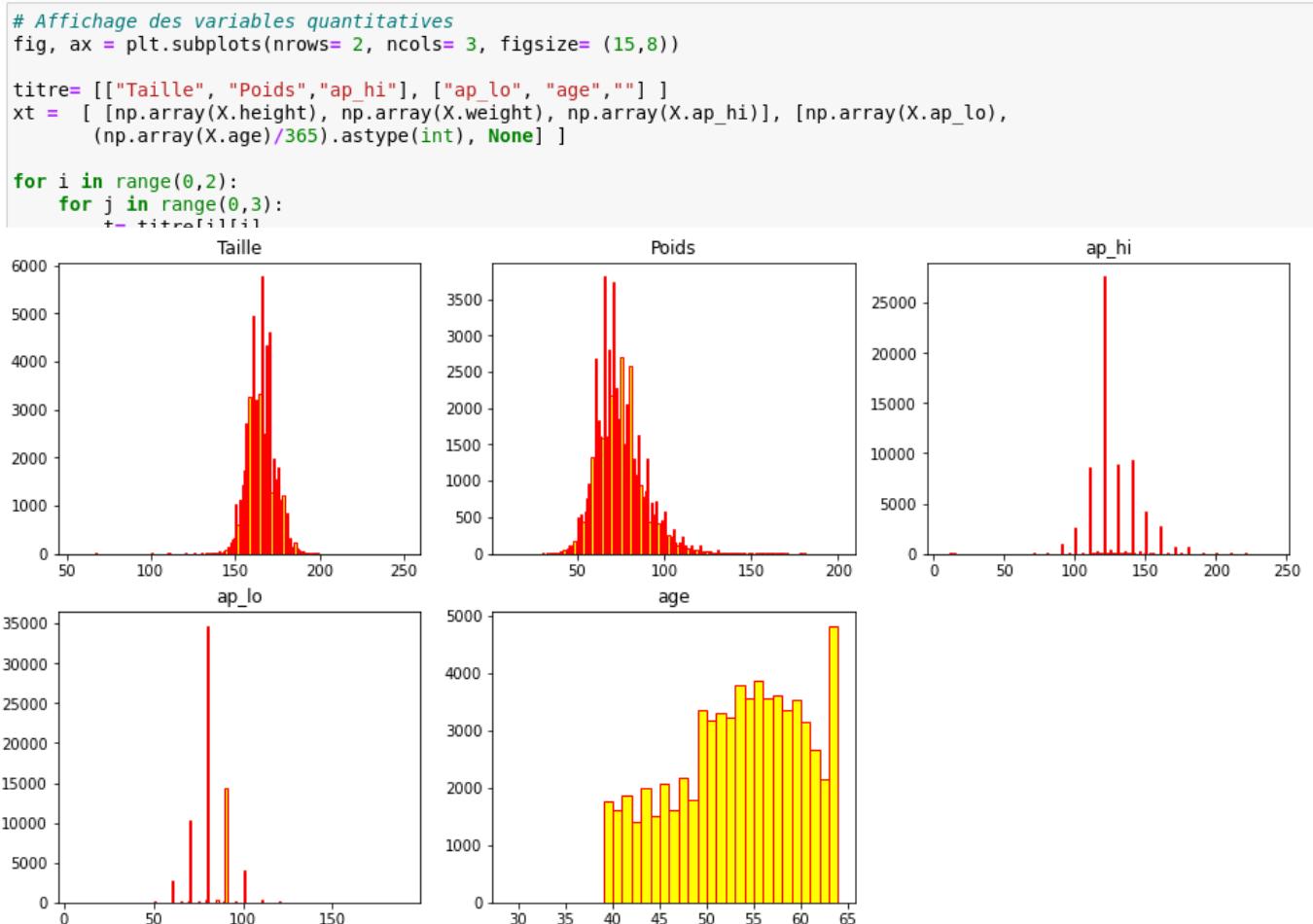
Valeur possible de la colonne 'gender': [1 2]
Valeur possible de la colonne 'cholesterol': [1 2 3]
Valeur possible de la colonne 'gluc': [1 2 3]
Valeur possible de la colonne 'smoke': [0 1]
Valeur possible de la colonne 'alco': [0 1]
Valeur possible de la colonne 'active': [0 1]
Valeur possible de la colonne 'cardio': [0 1]
```

On note que le jeu de données n'est pas équilibré. Beaucoup plus de non fumeurs que de fumeurs, beaucoup plus d'hommes que de femmes, beaucoup de non consommateurs d'alcool, beaucoup d'actifs, peu de taux de glucose et de taux de cholestérol anormaux.



iii Variables quantitatives.

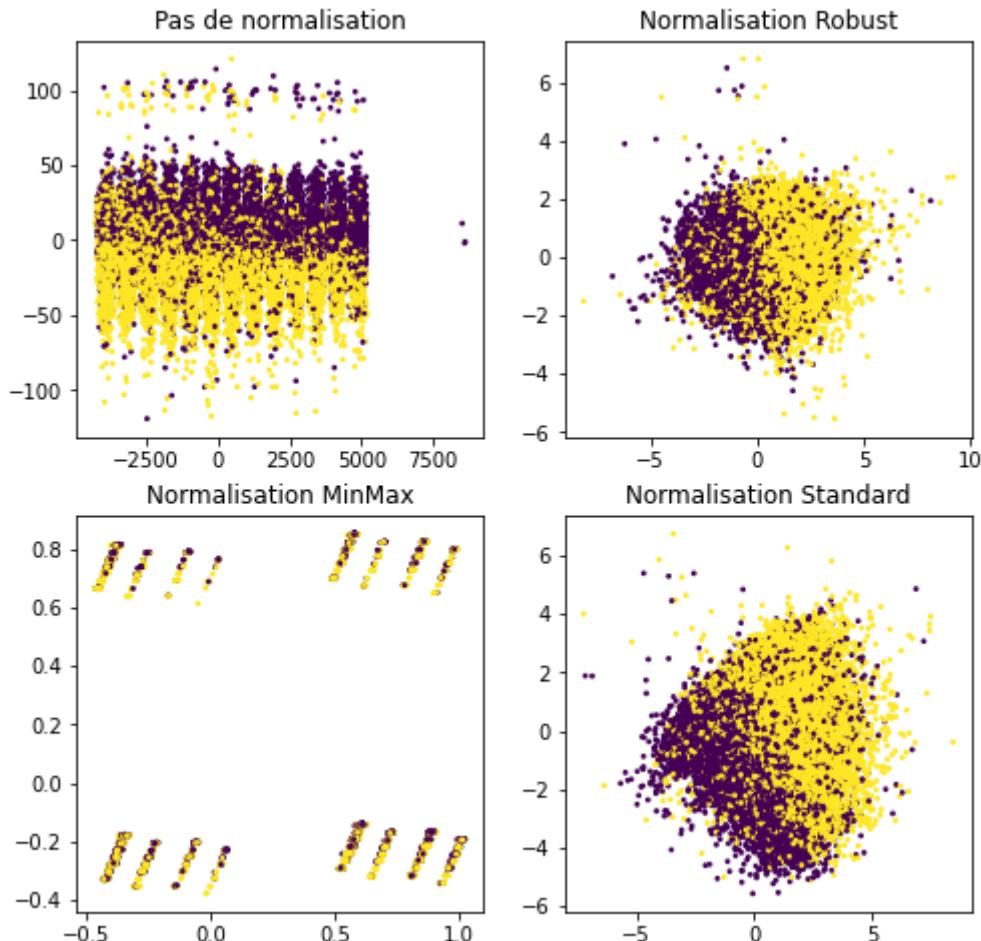
Voici la répartition des variables quantitatives. L'âge a été ramené en année pour une meilleure lecture.



iv Visualisation de la répartition des données.

Il peut être intéressant de visualiser le jeu de données. Pour ce faire, on peut utiliser l'analyse en composante principale en réduisant le nombre de variable d'entrées à 2. On peut regarder aussi l'effet de la standardisation sur le regroupement des données.

Comme on peut le constater, une normalisation utilisant le min/max n'est pas, à priori, un choix judicieux. Au vu des graphiques, on préférera la normalisation standard. On peut aussi relever la présence d'un petit groupe (sur le graphique sans normalisation) qui pourrait s'apparenter à des "outliers", c'est à dire des données anormales. Cependant ils sont très peu (donc l'influence sera minime) et je n'ai pas la compétence métier pour aller plus avant dans cette analyse.



c Test d'algorithmes

Référentiel : C5. Concevoir le programme d'intelligence artificielle adapté aux données disponibles afin de répondre aux objectifs fonctionnels du projet, à l'aide des algorithmes, outils et méthodes standards, notamment de machine learning et de deep learning

Référentiel : C6. Développer le programme d'intelligence artificielle selon les données du projet et les éléments de conception définis, en exploitant les algorithmes et les outils standards couramment utilisés dans le domaine

Ce petit bout de code va nous permettre de tester quelques classificateurs associés aux principaux algorithmes de standardisation.

```

1 for mod in [KNeighborsClassifier(), AdaBoostClassifier(), BaggingClassifier(),
2             RandomForestClassifier(), SGDClassifier(), GaussianNB(), BernoulliNB(), SVC()]:
3     print(f"Pour le modèle {mod}:")
4     print(f"Standardisation\tprécision\ttemps d'exécution")
5     for std in [None, StandardScaler(), RobustScaler(), MinMaxScaler()]:
6         deb= time.time()
7         pre= test(modele= mod, standardisation= std, Xtrain= X_train, Ytrain= Y_train, Xtest= X_test,
8                   Ytest= Y_test)
9         fin= time.time()
10        #if std== None: écrit= "None"
11        #else: écrit= str(std)
12        print(f"{std}\t{100*pre:.2f} %\t{fin-deb:.2f} s")
13

```

Certains algorithmes sont peu (voire pas du tout) sensibles à la normalisation contrairement à d'autres. On note toutefois que les scores de précision sont toujours meilleurs avec une normalisation (à l'exception du Gaussian ou les précisions sont quasi identiques).

Les résultats des différentes normalisations sont à peu près équivalents à l'exception du MinMax qui ne fonctionne pas avec le Bernoulli Naive Bayes.

On notera aussi que le SVM est particulièrement lent comparé aux autres algorithmes.

Enfin les algorithmes incluant plusieurs modèles (adaboost, random forest,...) sont performants. On s'intéressera donc aux techniques de baggins, boostings et stacking. On pourra alors faire travailler ensemble nos différents modèles que l'on aura optimisé.

KNeighborsClassifier():

	Standardisation	précision	temps
None	68.25 %	1.05 s	
StandardScaler()	69.47 %	5.44 s	
RobustScaler()	69.81 %	4.06 s	
MinMaxScaler()	68.88 %	5.87 s	

AdaBoostClassifier():

	Standardisation	précision	temps
None	73.11 %	1.39 s	
StandardScaler()	73.11 %	1.43 s	
RobustScaler()	73.11 %	1.44 s	
MinMaxScaler()	73.11 %	1.43 s	

BaggingClassifier():

	Standardisation	précision	temps
None	69.11 %	1.69 s	
StandardScaler()	68.85 %	1.67 s	
RobustScaler()	69.14 %	1.69 s	
MinMaxScaler()	69.04 %	1.68 s	

RandomForestClassifier():

	Standardisation	précision	temps
None	71.08 %	6.39 s	
StandardScaler()	71.08 %	6.31 s	
RobustScaler()	71.09 %	6.56 s	
MinMaxScaler()	71.25 %	9.87 s	

SGDClassifier():

	Standardisation	précision	temps
None	60.28 %	3.42 s	
StandardScaler()	71.15 %	0.31 s	
RobustScaler()	72.37 %	0.33 s	
MinMaxScaler()	71.14 %	0.22 s	

GaussianNB():			SVC():		
Standardisation	précision	temps	Standardisation	précision	temps
None	70.70 %	0.03 s	None	60.38 %	226.38 s
StandardScaler()	70.62 %	0.08 s	StandardScaler()	72.98 %	190.70 s
RobustScaler()	70.62 %	0.05 s	RobustScaler()	72.96 %	197.17 s
MinMaxScaler()	70.62 %	0.03 s	MinMaxScaler()	73.02 %	168.04 s
BernoulliNB():					
Standardisation	précision	temps			
None	51.89 %	0.04 s			
StandardScaler()	71.24 %	0.07 s			
RobustScaler()	71.16 %	0.12 s			
MinMaxScaler()	58.87 %	0.07 s			

i Optimisation des algorithmes.

Référentiel : C8. Modifier les paramètres et composants de l'intelligence artificielle afin d'ajuster aux objectifs du projet les capacités fonctionnelles de l'algorithme à l'aide de techniques d'optimisation

Pour le KNN, on a lancé une série de GridSearchCV. On s'aperçoit qu'au delà de 35 voisins les améliorations sont peu sensibles. On arrive à une précision de 72,5 % contre 69,5 % précédemment. Le modèle occupe 9,5 Mo.

Pour le random forest (qui est un algorithme de bagging), les résultats de l'optimisation sont décevants (on reste un peu au dessus des 71 %). Le modèle est assez volumineux, 282 Mo.

Le modèle SGD est particulièrement léger (2,2 Ko.... C'en est presque suspect !) et atteint les 72,5 % de précision. A noter que la méthode predict_proba ne fonctionne pas avec le paramètre loss= 'hinge'. De ce fait, nous ne pourrions pas utiliser voting=soft dans le voting classifier. Nous avons alors modifié ce paramètre à "log".

Pour les autres algorithmes, l'optimisation est aussi plutôt décevante. Nous avons entraîné 7 modèles (knn, descente de gradient stochastique, forêt aléatoire, gaussien, bernoulli, adaboost et svm) et nous allons essayé d'améliorer les performances en les faisant travailler ensemble. Le SVD a été écarté en raison de sa lenteur.

ii Le voting classifier.

La première méthode est une méthode de vote. Le principe est d'interroger chacun des algorithmes puis de procéder à un vote. 2 types de vote sont proposés, un vote à la majorité simple (utilisation de la méthode predict), et un vote tenant compte du degré de certitude avec laquelle les algorithmes répondent (utilisation de la méthode predict_proba).

```

rf= load("modeles/random-forest_rf160.h5")
knn= load("modeles/knn_knn35.h5")
sgd= load("modeles/sgd_sgd.h5")
gs= load("modeles/gaussianNB_gs00015199.h5")
bn= load("./modeles/bernoulli_bn.h5")

eclf1 = VotingClassifier(estimators=[('rf', rf), ('knn', knn), ('sgd', sgd), ('gs', gs), ('bn', bn)],
                         voting='hard')
eclf1 = eclf1.fit(X_train, Y_train)
#print(eclf1.predict(X_))

print(f"{100 * eclf1.score(X_test, Y_test):.2f} %")

```

72.78 %

Pour le "voting soft", les résultats sont identiques :

```

eclf2 = VotingClassifier(estimators=[('rf', rf), ('knn', knn), ('sgd', sgd), ('gs', gs), ('bn', bn)],
                         voting='soft')
eclf2 = eclf2.fit(X_train, Y_train)

print(f"{100*eclf2.score(X_test, Y_test):.2f} %")

```

72.78 %

Les résultats obtenus ne montrent pas une amélioration sensible du résultat. Quant au temps de calcul, il a été rallongé de façon significative. Cette méthode sera donc abandonnée.

iii Technique de stacking

Dans cette technique il s'agit d'entraîner plusieurs modèles sur le jeu de données puis, au lieu de pratiquer un vote, nous allons entraîner un modèle final qui recevra en entrée les prédictions des différents modèles et qui choisira le meilleur estimateur.

```

eclf3 = StackingClassifier(estimators=[('rf', rf), ('knn', knn), ('sgd', sgd), ('gs', gs), ('bn', bn)],
                           final_estimator= make_pipeline(StandardScaler(), RandomForestClassifier()))
eclf3 = eclf3.fit(X_train, Y_train)
print(f"{100*eclf3.score(X_test, Y_test):.2f} %")

```

71.79 %

Les résultats sont, là aussi, décevants.

iv Un mot sur la technique de boosting

Cette technique consiste à entraîner un algorithme sur la totalité du jeu de données puis créer un autre algorithme de même type qui sera entraîné sur les résultats provenant de son prédécesseur. Chaque algorithme est plutôt en under-fitting. En mettant en série un même type d'algorithme, cela permet d'améliorer les performances.

Il existe un algorithme de boosting dans scikit learn (Adaboost) qui utilise par défaut l'algorithme de classification "DecisionTree". Les tests montrent que sans optimisation particulière cet algorithme se montre plutôt performant (voir tableau ci-dessus).

v Un mot sur la technique de bagging

Cette technique consiste à entraîner plusieurs algorithmes de même type en parallèle sur une partie du jeu de données. Chaque algorithme est plutôt en over-fitting. En réunissant les différentes prévisions on peut espérer améliorer les performances.

Il existe un algorithme de bagging dans scikit learn (RandomForest) qui utilise l'algorithme de classification "DecisionTree". Si l'on souhaite utiliser un autre algorithme de classification en utilisant la technique du bagging, on utilisera "BaggingClassifier".

d Conclusion.

Que ce soit par les techniques classiques ou par les ensembles nous ne parvenons pas à franchir les 73 % de précision. Peut être parce que les données ne nous permettent pas d'aller au-delà. Faudrait il créer de nouvelles données comme, par exemple, l'IMC ?

Nous avons vu que le jeu de données était déséquilibré entre hommes et femmes et, par ailleurs, il semblerait que le risque de maladies cardiovasculaires soit différent entre les hommes et les femmes. On pourrait alors décider d'entraîner des algorithmes spécifiquement sur les hommes et sur les femmes.

Il se pourrait aussi que l'absence de données plus pertinentes (tenir compte de la masse graisseuse,...) nous empêche, quelque soit les algorithmes que nous pourrions utiliser, de dépasser cette limite (sauf au prix d'un over-fitting qui rendrait les prédictions inutilisables en pratique).

Peut être aussi que nous n'avons pas tenu suffisamment compte du fait que les données ne sont pas équilibrées ? Peu de fumeurs, de consommateurs d'alcool, de personnes ayant un taux de glucose ou de cholestérol au dessus de la normale,... ?

Enfin le jeu de données peut comporter plus d'anomalies que celles qui ont été révélées lors de l'analyse qui en a été faite.

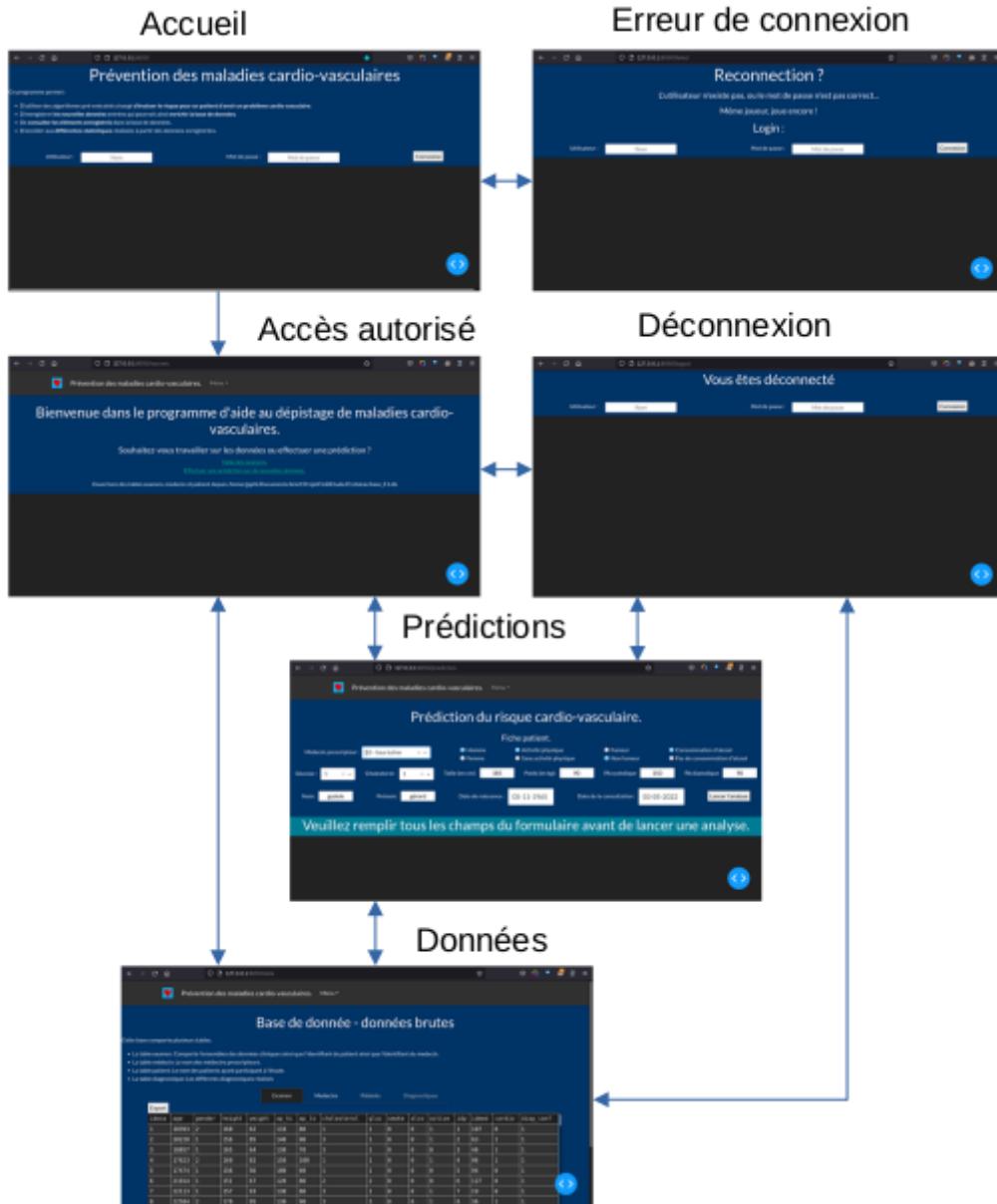
Dans tous les cas, une compétence métier serait nécessaire pour aller au-delà.

4 Réalisation d'un site web.

Référentiel : C7. Développer l'interaction entre les fonctionnalités de l'application et l'intelligence artificielle dans le respect des objectifs visés et des bonnes pratiques du domaine.

Voici un diagramme des pages du site et de leur inter connexion. Une des contraintes liée à ce genre de site est qu'il doit garantir une certaine confidentialité. Il est nécessaire de protéger tout ou partie de ce type de site par un accès sécurisé.

a Maquette du site.



b Cahier des charges :

Référentiel : C9. Analyser un besoin en développement d'application mettant en œuvre des techniques d'intelligence artificielle afin de produire les éléments de réponses techniques

Le programme permettra d'utiliser le résultat de la précédente étude pour réaliser des prédictions sur les risques de maladies cardiovasculaires en fonction des paramètres correspondant aux données cliniques de patients. Pour ce faire, il devra fournir une interface graphique permettant l'enregistrement des données, la lecture des résultats, l'enregistrement de nouvelles données dans la base, d'enregistrer les résultats du test d'un patient particulier sous forme de fichier log par exemple et de toutes autres fonctions qui pourraient améliorer l'expérience utilisateur.

L'interface graphique devra être protégée en accès par un système de login/mot de passe.

L'interface utilisateur devra comporter plusieurs pages implémentant les diverses fonctionnalités.

Une page sera consacrée à l'entrée des données d'un patient dont on cherche à déterminer s'il a un risque cardiovasculaire.

Une autre page sera consacrée à la visualisation des données déjà enregistrées.

D'autres pages pourraient être rajoutées facilement élargissant les fonctionnalités de ce site. Par exemple une page pourrait être consacrée aux statistiques sur les données enregistrées, une autre dédiée à la gestion de la base de données,....

c Utilisation du programme.

Référentiel : C11. Développer les requêtes et les composants d'accès aux données dans un langage adapté afin de persister et mettre à jour les données issues de l'application en base de données

Référentiel : C12. Développer le back-end de l'application d'intelligence artificielle dans le respect des spécifications fonctionnelles et des bonnes pratiques du domaine

Référentiel : C13. Développer le front-end de l'application d'intelligence artificielle à partir de maquettes et du parcours utilisateur, dans le respect des objectifs visés et des bonnes pratiques du domaine

La première page du site décrit rapidement l'utilité de cette application et demande un nom d'utilisateur et son mot de passe.

Prévention des maladies cardio-vasculaires

Ce programme permet :

- D'utiliser des algorithmes pré-entraînés chargé d'évaluer le risque pour un patient d'avoir un problème cardio vasculaire.
- D'enregistrer les nouvelles données entrées qui pourrait ainsi enrichir la base de données.
- De consulter les éléments enregistrés dans la base de données.
- D'accéder aux différentes statistiques réalisées à partir des données enregistrées.

Utilisateur : Nom Mot de passe : Mot de passe Connexion

Une fois ces données renseignées, une page d'accueil permettant l'accès à toutes les fonctionnalités du site... au nombre de 2 à l'heure où j'écris ce rapport ! On note la présence d'un menu déroulant qui permet d'accéder à toutes les fonctionnalités de la partie privée du site, ainsi qu'à la possibilité de se déconnecter. Le menu n'est pas la seule façon d'accéder aux autres parties privées du site, les liens au milieu de la page le permettent également.

C'est également ici que l'on charge les données nécessaires au fonctionnement de cette application, comme l'indique la dernière ligne de la page.

Prévention des maladies cardio-vasculaires. Menu ▾

Bienvenue dans le programme d'aide au dépistage de maladies cardio-vasculaires.

Souhaitez-vous travailler sur les données ou effectuer une prédiction ?

[Table des examens.](#)
[Effectuer une prédiction sur de nouvelles données.](#)

Ouverture des tables examen, medecin et patient depuis /home/jpphi/Documents/brief/ProjetFinDEtude/E1/datas/base_E1.db

Depuis cette page, on accède aux 2 fonctionnalités pour l'instant implémentées. Comme précédemment ces pages inclus un menu déroulant permettant une navigation aisée.

La page prédiction permet de soumettre les données d'un patient à une série d'algorithmes préalablement entraînée.

Prévention des maladies cardio-vasculaires. Menu ▾

Prédiction du risque cardio-vasculaire.

Fiche patient.

Médecin prescripteur:	Select...	<input checked="" type="radio"/> Homme	<input checked="" type="radio"/> Activité physique	<input checked="" type="radio"/> Fumeur	<input checked="" type="radio"/> Consommation d'alcool			
		<input type="radio"/> Femme	<input type="radio"/> Sans activité physique	<input type="radio"/> Non fumeur	<input type="radio"/> Pas de consommation d'alcool			
Glucose :	1	Cholesterol :	1	Taille (en cm): 185	Poids (en kg): 90	PA systolique : 150	PA diastolique : 90	
Nom:	Nom	Prénom:	Prénom	Date de naissance:	05-11-1965	Date de la consultation:	03-05-2022	Lancer l'analyse

Veuillez remplir tous les champs du formulaire avant de lancer une analyse.

La page des données permet de consulter les différentes tables.

Prévention des maladies cardio-vasculaires. Menu ▾

Base de donnée - données brutes

Cette base comporte plusieurs tables.

- La table examen: Comporte l'ensemble des données cliniques ainsi que l'identifiant du patient ainsi que l'identifiant du medecin.
- La table medecin: Le nom des médecins prescripteurs.
- La table patient: Le nom des patients ayant participé à l'étude.
- La table diagnostique: Les différents diagnostics réalisés

	Examen	Medecins	Patients	Diagnostiques											
Export															
indexa	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	idp	idmed	cardio	diag_conf
1	18393	2	168	62	110	80	1	1	0	0	1	1	107	0	1
2	20228	1	156	85	140	90	3	1	0	0	1	2	63	1	1
3	18857	1	165	64	130	70	3	1	0	0	0	3	46	1	1
4	17623	2	169	82	150	100	1	1	0	0	1	4	96	1	1
5	17474	1	156	56	100	60	1	1	0	0	0	5	94	0	1
6	21914	1	151	67	120	80	2	2	0	0	0	6	127	0	1
7	22113	1	157	93	130	80	3	1	0	0	1	7	19	0	1
8	22584	2	178	95	130	90	3	3	0	0	1	8	36	1	1

Enfin la fonction de déconnexion permet de... se déconnecter !

Vous êtes déconnecté

Utilisateur:	Nom	Mot de passe:	Mot de passe	Connexion
--------------	-----	---------------	--------------	-----------

Une erreur dans le couple nom utilisateur mot de passe, ou une tentative d'accéder aux pages protégées sans être "authentifié" vous invitera à saisir à nouveau vos identifiant/mot de passe.

127.0.0.1:8050/failed

Reconnection ?

L'utilisateur n'existe pas, ou le mot de passe n'est pas correct...

Même joueur, joue encore !

Login :

Utilisateur : Nom

Mot de passe : Mot de passe

d Effectuer une prédition.

C'est sur la page prédition que l'on va pouvoir remplir tous les champs relatifs au patient. Le médecin ayant fait la prescription médicale, l'ensemble des données cliniques, son nom, prénom et date de naissance ainsi que le jour de l'examen. Tous ces champs sont actuellement pré-remplis pour faciliter la démonstration de l'application. Mais il est bien évident qu'aucun de ces champs ne devra avoir une valeur par défaut pour s'assurer qu'ils seront bien saisis.

Lorsque tous les champs ont été saisis, il suffit de cliquer sur le bouton "lancer l'analyse".

Les données sont alors transmises aux différents algorithmes de prédition préalablement entraînés. La réponse donnée est la probabilité d'avoir une maladie cardiovasculaire. On considère qu'en dessous de 50 % la prédition sera de 0, 1 au-delà.

Prédiction du risque cardio-vasculaire.

Fiche patient.

Médecin prescripteur: <input type="text" value="1 - maffre lucas"/>	<input type="radio"/> Homme	<input type="radio"/> Activité physique	<input type="radio"/> Fumeur	<input type="radio"/> Consommation d'alcool	
	<input type="radio"/> Femme	<input type="radio"/> Sans activité physique	<input type="radio"/> Non fumeur	<input type="radio"/> Pas de consommation d'alcool	
Glucose : <input type="text" value="1"/>	Cholesterol : <input type="text" value="1"/>	Taille (en cm): <input type="text" value="185"/>	Poids (en kg): <input type="text" value="85"/>	PA systolique : <input type="text" value="130"/>	PA diastolique : <input type="text" value="80"/>
Nom: <input type="text" value="Neymar"/>	Prénom: <input type="text" value="Jean"/>	Date de naissance: <input type="text" value="05-11-1965"/>	Date de la consultation: <input type="text" value="03-05-2022"/>	<input type="button" value="Lancer l'analyse"/>	

Veuillez remplir tous les champs du formulaire avant de lancer une analyse.

Compte rendu d'analyse:

Homme, pratiquant une activité physique, non fumeur(se), consommant de l'alcool, avec un taux de glucose normal, avec un taux de cholesterol normal, taille : 185 cm, poids : 85 kg, pression systolique : 130, pression diastolique : 80, né(e) le : 1965-11-05, date de la consultation : 2022-05-03.
Mme/M (Nom prénom) Neymar Jean, adressé par (numéro d'enregistrement, nom, prénom): 1 - maffre lucas.

Résultats des analyses:

Le nom des algorithmes est suivi de la probabilité d'avoir un risque, pour ce patient, de maladie cardio-vasculaire.

- random-forest_rf160.h5 : 38.75 %
- gaussianNB_gs00015199.h5 : 25.60 %
- adaboost_absgd.h5 : 49.96 %
- knn_knn35.h5 : 48.57 %
- bernoulli_bn.h5 : 72.18 %
- sgd_sgd.h5 : 48.69 %

Les données seront ensuite ensuite enregistrées dans une base. Pour le patient, s'il existe déjà, son identifiant sera enregistré dans la table "diagnostique" avec l'identifiant du médecin et les résultats de cette étude. S'il n'existe pas déjà, il sera enregistré dans la table patient de la base. On récupérera ainsi son identifiant. Le tout sera enregistré dans la table diagnostique.

e La fonction alerte.

Référentiel : C15. Maintenir l'application d'intelligence artificielle à l'aide des techniques de monitorage afin de détecter et corriger les éventuels dysfonctionnements

Lorsque les choses se passent mal, l'erreur est affichée dans la zone compte rendu de la fenêtre. Ici, on teste cette fonction en rentrant la date du jour comme date de naissance. Ceci nous permet de tester cette fonction alerte.

The screenshot shows a web browser window with the URL `127.0.0.1:8050/prediction`. The page title is "Prévention des maladies cardio-vasculaires." and the main heading is "Prédiction du risque cardio-vasculaire.". The form includes fields for "Médecin prescripteur" (1 - maffre lucas), gender (Homme), height (185 cm), weight (90 kg), systolic blood pressure (150), diastolic blood pressure (90), glucose (1), cholesterol (1), smoking status (Fumeur), alcohol consumption (Consommation d'alcool), name (toto), first name (tutu), birth date (03-05-2022), and consultation date (03-05-2022). A button labeled "Lancer l'analyse" is visible. A prominent red banner at the bottom states: "Veuillez remplir tous les champs du formulaire avant de lancer une analyse." Below this, a message says: "Une erreur est survenue !" followed by "Le message envoyé par le code est: Message dans prediction: âge du patient= 0". A link "Envoyer le rapport d'erreur par mail" is present, along with a blue circular icon containing a double arrow symbol.

Il y a 3 cas où la fonction alerte est appelée depuis la page prédition :

- ✗ La recherche ou la création du patient dans la table patient s'est mal passée.
- ✗ L'analyse des données a produit un résultat inattendu.
- ✗ L'enregistrement des données dans la table diagnostique s'est mal passé.

L'utilisation de cette fonctionnalité pourrait être étendue.

5 Gestion de projet

Référentiel : C18. Communiquer avec les parties prenantes afin de rendre compte de l'avancement du projet en mettant en œuvre les canaux de communication nécessaires.

Référentiel : C16. Planifier les actions du projet à l'aide d'un outil adapté afin de prévoir la complétion du projet dans les temps impartis

a Gestion de projet dans l'entreprise :

Dans l'entreprise qui m'accueille en alternance, la gestion de la communication autour des différents projets est organisée autour d'un document que l'on remplit au fil de l'eau. Il reprend tous les sujets que nous avons abordé ainsi que leur état d'avancement et les points de blocage éventuels.

Tous les lundis matins nous avons une réunion d'une à deux heures où chacun présente son document. Voici un extrait de mon document pour la semaine 15.... particulièrement chargée !

Dashboard WK 15

15TH APRIL, 2022

1 Détection de mouvement.

Project	Type	Description
Optical flow	Rédaction	Finalisation des documents d'études. <ul style="list-style-type: none"> - optical_flow_rapport_JPM.docx: étude sur les différentes solutions pour extraire un avant plan sur une caméra en mouvement. - "Optical flow STV0991 - correction mouvement camera.docx": Document d'étude sur la façon dont la STV0991 retourne le flux optique. Pistes pour "corriger" les vecteurs du vecteur du mouvement caméra.
Optical flow	Programmation	Programmes d'étude pour optical flow. <ul style="list-style-type: none"> - Le programme <code>extract_avant-plan_orb.py</code> reprend les solutions de la STV0991 pour extraire l'avant plan. Sur une vidéo 720p format 4/3 on peut travailler à plus de 20 fps tout en intégrant la correction du mouvement caméra. A ce stade, on peut envisager l'utilisation d'optical flow en temps réel. - Le programme <code>extract_avant-plan_sift.py</code> a permis de comparer les solutions d'extraction d'avant-plan temps réel à partir d'algorithme de reconnaissance de points. Avec "ORB" (solution proche de celle de la STV0991) on tourne à 20/25 fps sur une image 4/3 720p, avec SIFT on ne passe pas les 2 fps. On notera que SURF, version rapide de SIFT n'est pas dans la version gratuite d'openCV.

2 Sujet d'étude: Optical flow et avant-plan rapide.

Project	Type	Description
		Optical flow et avant-plan à vitesse rapide. <ul style="list-style-type: none"> - Il faudra tester la capacité de capter des mouvements haute vitesse. Dans un premier temps sur la simulation logiciel mise au point en trouvant des vidéos avec mouvement rapide de l'avant-plan. - Par la suite, comment récupérer la tables des vecteurs depuis la STV0991

b Gestion de projet méthode "agile"

Lors de notre formation, nous avons eu l'occasion de travailler en équipe en utilisant la méthode agile. Je joins, en annexe, les différentes étapes de l'élaboration du projet "chat-bot" reprenant les différents rituels.

6 Annexes.

a Jupyter notebook :

C'est le fichier avec lequel a été réalisé la mise au point, l'optimisation ainsi que l'ensemble des tests des différents modèles de machine learning.

b Programme python de l'application web :

Ce programme est une interface web réalisée grâce à la librairie dash. Il comprend :

- 2 fichiers pythons devant se situer à la racine (app.py et cv.py).
- Une base de donnée sqlite contenant les utilisateurs ayant le droit d'utiliser l'application. Cette base doit être au même niveau que les 2 fichiers précédents. Cette base comprend les champs id, username, email et password (crypté).
- Un sous-répertoire apps contenant les fichiers : sucess.py, prediction.py, home.py, login.py, data.py, failed.py, logout.py, glob.py et navbar2.py

c Plan du site web de l'application.

d Utilitaire de création de base et d'utilisateurs.

C'est un petit utilitaire sous forme d'application web et qui permet de créer une base de données ainsi que les utilisateurs qui pourront accéder à l'application de prédition des maladies cardiovasculaires.

Un champs "type d'utilisateur" dans la table utilisateur (sous forme de clé étrangère) et une table type utilisateur permettant de mieux filtrer les accès aux différentes parties du site est créé mais n'est, pour l'instant, pas utilisé.

e Gestion de projet – méthode "agile".

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        fichier lançant l'application. Doit se situer à la racine du site

Ressources:
    https://ichi.pro/fr/comment-configurer-l-authentification-utilisateur-pour-les-
applications-
dash-a-l-aide-de-python-et-flask-118945949211473
    https://dash.plotly.com/
    https://dash-bootstrap-components.opensource.faculty.ai/docs/components/

@auteur: jpphi
"""

from dash import dcc
from dash import html
from dash.dependencies import Input, Output, State
import dash_bootstrap_components as dbc

from sqlalchemy import Table, create_engine
from sqlalchemy.sql import select
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
import warnings
from flask_login import login_user, logout_user, current_user, LoginManager, UserMixin
import configparser

import os

from app import app, server
from apps import home, login, success, failed, data, logout, prediction, home

warnings.filterwarnings("ignore")
#engine = create_engine('sqlite:///data.sqlite')
engine = create_engine('sqlite:///utilisateurs.sqlite')
db = SQLAlchemy()
config = configparser.ConfigParser()

class Users(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(15), unique=True, nullable = False)
    email = db.Column(db.String(50), unique=True)
    password = db.Column(db.String(80))
    #clair = db.Column(db.String(80))

Users_tbl = Table('users', Users.metadata)
Type_users_tbl = Table('typeusers', Users.metadata)

#server = app.server
#app.config.suppress_callback_exceptions = True

# config
server.config.update(
    SECRET_KEY=os.urandom(12),
    SQLALCHEMY_DATABASE_URI='sqlite:///data.sqlite',
    SQLALCHEMY_TRACK_MODIFICATIONS=False
)
db.init_app(server)

# Setup the LoginManager for the server
login_manager = LoginManager()
login_manager.init_app(server)

```

```

login_manager.login_view = '/login'

#User as base
# Create User class with UserMixin
# Sans cette classe, une erreur est générée empêchant la connexion à la base
# de donnée (User n'a pas d'attribut is_active)
class Users(UserMixin, User):
    pass

----- Layout de l'application -----

app.layout= html.Div([
    html.Div(id='page-content', className='content'),
    dcc.Location(id='url', refresh=False),
])
----- Callback -----


# callback to reload the user object
@login_manager.user_loader
def load_user(user_id):
    return Users.query.get(int(user_id))

@app.callback(
    Output('page-content', 'children')
    , [Input('url', 'pathname')])
def display_page(pathname):
    if pathname== '/':
        return home.layout

    elif pathname== '/login':
        return login.layout

    elif pathname== '/success':
        if current_user.is_authenticated:
            return success.layout
        else:
            return failed.layout

    elif pathname== '/data':
        if current_user.is_authenticated:
            return data.layout
        else:
            return failed.layout

    elif pathname== '/failed':
        return failed.layout
        # if current_user.is_authenticated:
        #     return success.layout
        # else:
        #     return failed.layout
    elif pathname== '/logout':
        if current_user.is_authenticated:
            logout_user()
            return logout.layout
        else:
            return login.layout

    elif pathname== '/prediction':
        if current_user.is_authenticated:
            return prediction.layout
        else:
            return failed.layout
    else:
        return '404'

@app.callback(Output('url_login', 'pathname')
    , [Input('login-button', 'n_clicks')])
    , [State('uname-box', 'value'), State('pwd-box', 'value')])
```

```
def successful(n_clicks, input1, input2):
    user = Users.query.filter_by(username=input1).first()
    if user:
        if check_password_hash(user.password, input2):
            login_user(user)
            return '/success'
        else:
            return '/failed'
    else:
        pass

#----- Main -----
if __name__ == '__main__':
    app.run_server(debug=True)
    #serve(app, host="0.0.0.0", port=8080)
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé le 25 décembre 2021

Projet de fin d'étude Simplon
Serveur

@author: jpphi
"""

import dash
import dash_bootstrap_components as dbc

# bootstrap theme
external_stylesheets = [dbc.themes.DARKLY]

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
app.title = 'Prévention des maladies cardio-vasculaire'

app._favicon = "favicon.ico"

server = app.server
app.config.suppress_callback_exceptions = True
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        home.py; page home. Ce fichier doit se situer dans le sous-répertoire /apps

@auteur: jpphi
"""

from dash import dcc, html
import dash_bootstrap_components as dbc

from apps import login, glob

layout= html.Div([
    html.H1("Prévention des maladies cardio-vasculaires", id='h1', style= {"text-align": "center"}),
    dbc.Row([dcc.Markdown('''
Ce programme permet :
- D'utiliser des algorithmes pré-entraînés chargé **d'évaluer le risque pour un patient d'avoir un problème cardio vasculaire**.
- D'enregistrer **les nouvelles données** entrées qui pourrait ainsi **enrichir la base de données**.
- De **consulter les éléments enregistrés** dans la base de données.
- D'accéder aux **différentes statistiques** réalisées à partir des données enregistrées.''), 
        html.P(),
        html.Div(login.layout),
    ],
    style={'background-color': glob.fond_ecran_formulaire, "display": "flex",
    "flex-direction": "column", "justify-content": "space-between", })
```

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        login.py; page login. Ce fichier doit se situer dans le sous-répertoire /apps

@auteur: jpphi
"""

from dash import dcc, html
#import dash_bootstrap_components as dbc

from app import app
from apps import glob

layout= html.Div([
    dcc.Location(id='url_login', refresh=True),

    html.H2("Login : ",style={"text-align": "center"}),
    html.P(""),
    html.Div([
        html.Div([
            html.P(children= "Utilisateur :"),
            dcc.Input(placeholder="Nom", type='text', id='uname-box',
                      style= {"text-align": "center", "height": "30px"}, ),
        ],
        style= {"display": "flex", "text-align": "center",
                "align-content": "center", "gap": "20px", "justify-content": "space-around"},),

        html.Div([
            html.P(children= "Mot de passe :"),
            dcc.Input(placeholder="Mot de passe", type='password', id='pwd-
box',
                      style= {"text-align": "center", "height": "30px"}, #,
value="toto"
        ],
        style= {"display": "flex", "text-align": "center",
                "align-content": "center", "gap": "20px", "justify-content": "space-around"},),

        #html.Div([
        #    html.Button(children='Connexion', n_clicks=0, type='submit',
        #                  id='login-button', style= {"text-align": "center",
        #                                         "height": "30px"}),
        #],
        #style= {"display": "flex", "text-align": "center",
        #        "#      "align-content": "center", "gap": "20px", "justify-content": "space-around"},),

        html.Div(children='', id='output-state',
                  style={ "display": "flex", 'background-color': glob.fond_ecran_formulaire,}),


    ],
    style= {"display": "flex", "background-color": glob.fond_ecran_formulaire,
            "justify-content": "space-around", "align-items": "baseline"})

],style= {"background-color": glob.fond_ecran_formulaire, })

```

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        navbar2.py; layout de la barre de navigation.
            Ce fichier doit se situer dans le sous-répertoire /apps

@auteur: jpphi
"""

from dash.dependencies import Input, Output, State
import dash_bootstrap_components as dbc
from dash import html

from app import app

dropdown = dbc.DropdownMenu(
    children=[
        dbc.DropdownMenuItem("Consultation données", href="/data"),
        dbc.DropdownMenuItem("Prédiction", href="/prediction"),
        dbc.DropdownMenuItem("logout", href="/logout"),
    ],
    nav = True,
    in_navbar = True,
    label = "Menu",
)

layout = dbc.Navbar(
    dbc.Container(
        [
            dbc.Row(
                [
                    dbc.Col(html.Img(src="../assets/stethoscope-coeur2.png",
height="30px")),
                    dbc.Col(
                        dbc.NavbarBrand("Prévention des maladies cardio-
vasculaires."),
                    ),
                ],
                #no_gutters=True,
            ),
            dbc.NavbarToggler(id="navbar-toggler2"),
            dbc.Collapse(
                dbc.Nav(
                    [dropdown], className="ml-auto", navbar=True
                ),
                id="navbar-collapse2",
                navbar=True,
            ),
        ],
        style={"display": "flex", "justify-content": "space-around"}
    ),
    color="dark",
    dark=True,
    className="mb-4",
)

def toggle-navbarCollapse(n, is_open):
    if n:
        return not is_open
    return is_open

for i in [2]:
    app.callback(

```

```
Output(f"navbar-collapse{i}", "is_open"),
[Input(f"navbar-toggler{i}", "n_clicks")],
[State(f"navbar-collapse{i}", "is_open")],
)(toggle-navbarCollapse)
```

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        page success; ce fichier doit se situer dans le sous-répertoire /apps

@author: jpphi
"""

import pandas as pd
import sqlite3

from dash import dcc, html
from dash.dependencies import Input, Output, State
import dash_bootstrap_components as dbc

from app import app
from apps import navbar2, glob

#----- Ouverture des tables -----

if glob.df_examen== None or glob.df_medecin== None or glob.df_patient== None \
    or glob.df_diagnostique== None:
    try:
        with sqlite3.connect(glob.bdd) as conn:
            glob.df_examen= pd.read_sql("SELECT * from examen",conn)
        with sqlite3.connect(glob.bdd) as conn:
            glob.df_medecin= pd.read_sql("SELECT * from medecin",conn)
        with sqlite3.connect(glob.bdd) as conn:
            glob.df_patient= pd.read_sql("SELECT * from patient",conn)
        with sqlite3.connect(glob.bdd) as conn:
            glob.df_diagnostique= pd.read_sql("SELECT * from diagnostique",conn)
        cpt_rendu= f'Ouverture des tables examen, medecin et patient depuis {glob.bdd}'
    except:
        cpt_rendu= f"Une des table de la base {glob.bdd} n'a pu être chargé"

layout = html.Div([
    dcc.Location(id='url_login_success', refresh=True),
    navbar2.layout,
    html.Div([
        html.H2("Bienvenue dans le programme d'aide au dépistage de maladies "+\
            "cardio-vasculaires.", style={"text-align": "center"}),
        html.P(),
        html.H4('Souhaitez-vous travailler sur les données ou effectuer une
    prédition ?',
            style= {"text-align": "center"}),
        dcc.Link('Table des examens.', href = '/data'),
        dcc.Link('Effectuer une prédition sur de nouvelles données.', href = '/-
    prediction'),
        ],style={"display": "flex", 'background-color': glob.fond_ecran_formulaire,
        "flex-direction": "column", "align-items": "center"},),
        html.P(),
        html.Div([
            html.P(cpt_rendu,style={"text-align": "center"}),
        ]),
    ],style={"display": "flex", 'background-color': glob.fond_ecran_formulaire,
    "flex-direction": "column"}) #end div

```

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        page prediction; ce fichier doit se situer dans le sous-répertoire /apps

@auteur: jpphi
"""

from dash import dcc, html
import dash_bootstrap_components as dbc
from dash.dependencies import Input, Output, State

import datetime
import ast

from apps import navbar2, glob
from app import app

layout = html.Div([navbar2.layout,

    html.H2("Prédiction du risque cardio-vasculaire.", style={"text-align":"center"}),

    html.P(children= ""),

    html.H4("Fiche patient.", style={"text-align":"center"}),

    # Boutons radios. 1er groupe de 4 entrées
    html.Div([
        html.Div([
            html.P(children= "Médecin prescripteur:"),  

            dcc.Dropdown(id='dd-medecin',
                options= [str(row.idmed) + " - " + str(row.nom).lower() + " " +  

                    str(row.prenom) for index, row in glob.df_medecin.iterrows()],
                style= {"width": "200px", "color": glob.fond_ecran_formulaire,},  

                value= ""),
            ], style={"display": "flex", "align-items": "baseline",
                "justify-content": "space-around", "gap": "10px"}),
        html.Div([
            #html.P(children= "Genre :"),
            dcc.RadioItems(id="ri-genre", options=[{'label': 'Homme', 'value': 'h'},
            {'label': 'Femme', 'value': 'f'}], value= "h",
            labelStyle={'display': 'flex', "gap": "5px"},),
            ], style={"display": "flex", "align-items": "flex-start", "gap": "10px",
                "justify-content": "space-between"},),
        html.Div([
            #html.P(children= "Activité physique :"),
            dcc.RadioItems(id="ri-activite", options=[{'label': 'Activité physique', 'value': 'act'},
            {'label': 'Sans activité physique', 'value': 'inact'}], value= "act",
            labelStyle={'display': 'flex', "gap": "5px"},),
            ], style={"display": "flex", "align-items": "flex-start", "gap": "10px",
                "justify-content": "space-around"},),
        html.Div([
            #html.P(children= "Tabagisme :"),

```

```

        dcc.RadioItems(id= "ri-tabac", options=[{'label': ' Fumeur', 'value': 'fum'},
            {'label': ' Non fumeur', 'value': 'nfum'}, ], value= "nfum",
            labelStyle={'display': 'flex', "gap": "5px"}),
        ], style={"display": "flex", "align-items": "flex-start",
            "justify-content": "space-around", "gap": "10px"}),

        html.Div([
            #html.P(children= "Alcool :"),
            #html.P("    "),
            dcc.RadioItems(id= "ri-alcool", options=[{'label': "Consommation
d'alcool", 'value': 'alc'}, {'label': "Pas de consommation d'alcool", 'value': 'nalc'}], value= "alc",
                labelStyle={'display': 'flex', "gap": "5px"}),
            ], style={"display": "flex", "align-items": "flex-start",
                "justify-content": "space-around", "gap": "10px"}),
            ], style={'background-color': glob.fond_ecran_formulaire, "display": "flex",
"align-items": "baseline",
"justify-content": "space-around",}),

        html.P(children= ""),

    # 2 dropdown 2 input. 2ème groupe de 4 entrées
    html.Div([
        html.Div([
            html.P(children= "Glucose :"),
            dcc.Dropdown(id='dd-glucose', options= [1,2,3], placeholder= '1, 2 ou
3',
                style= {"width": "100px", "color": glob.fond_ecran_formulaire}, value= 1, ),
            ], style={"display": "flex", "align-items": "baseline",
                "justify-content": "space-around", "gap": "10px"}),

        html.Div([
            html.P(children= "Cholesterol :"),
            dcc.Dropdown(id='dd-cholesterol', options= [1,2,3], placeholder= '1,
2 ou 3',
                style= {"width": "100px", "color": glob.fond_ecran_formulaire}, value= 1, ),
            ], style={"display": "flex", "align-items": "baseline",
                "justify-content": "space-around", "gap": "10px", "width": "50
%"}),

        html.Div([
            html.P(children= "Taille (en cm):"),
            dcc.Input(placeholder= "[50;280]", type='text', id='i-taille',
                min= 50, max= 280, style= {"text-align": "center",
                "width": "100px"}, value= 185),
            ], style={"display": "flex", "align-items": "baseline", "gap": "10px",
                "justify-content": "space-around",}),

        html.Div([
            html.P(children= "Poids (en kg):"),
            dcc.Input(placeholder= "[10;650]", type='text', id='i-poids',
                min= 10, max= 650, style= {"text-align": "center",
                "width": "100px"}, value= 90),
            ], style={"display": "flex", "align-items": "baseline", "gap": "10px",
                "justify-content": "space-around",}),

        html.Div([
            html.P(children= "PA systolique"),
            dcc.Input(placeholder= "[10;300]", type='text', id='i-pasys',
                min= 10, max= 300, style= {"text-align": "center",
                "width": "100px"}, value= 120),
            ], style={"display": "flex", "align-items": "baseline", "gap": "10px",
                "justify-content": "space-around",}),
    
```

```

        value= 150,),
    ], style={"display": "flex", "align-items": "baseline",
               "justify-content": "space-around", "gap": "10px"}),

    html.Div([
        html.P(children= "PA diastolique"),
        dcc.Input(placeholder="[10;300]", type='text', id='i-padia',
                  min= 10, max= 300, style= {"text-align": "center",
"width": "100px"}, value= 90),
    ], style={"display": "flex", "align-items": "baseline",
               "justify-content": "space-around", "gap": "10px"}),

], style={'background-color': glob.fond_ecran_formulaire, "display": "flex",
"justify-content": "space-around", "align-items": "center"}),

html.P(children= ""),

# 2 date 2 input. 3ème groupe de 4 entrées
html.Div([
    html.Div([
        html.P(children= "Nom:"), 
        dcc.Input(placeholder="Nom", type='text', id='i-nom',
                  style= {"text-align": "center", "width": "100px"}, value= "toto",),
    ], style={"display": "flex", "align-items": "baseline", "gap": "10px",
               "justify-content": "space-around"},),

    html.Div([
        html.P(children= "Prénom:"), 
        dcc.Input(placeholder="Prénom", type='text', id='i-prenom',
                  style= {"text-align": "center", "width": "100px"}, value= "tutu",),
    ], style={"display": "flex", "align-items": "baseline", "gap": "10px",
               "justify-content": "space-around"},),

    html.Div([
        html.P(children= "Date de naissance :"),
        dcc.DatePickerSingle(
            id='dt-naissance',
            min_date_allowed= datetime.date(1900, 1, 1),
            max_date_allowed= datetime.date.today(), #(2017, 9, 19)
            initial_visible_month= datetime.date(1965, 11, 5),
            date= datetime.date(1965, 11, 5),
            display_format= "DD-MM-Y"
        ),
    ], style={"display": "flex", "align-items": "baseline",
               "justify-content": "space-around", "gap": "10px"}),

    html.Div([
        html.P(children= "Date de la consultation :"),
        dcc.DatePickerSingle(
            id='dt-consultation',
            min_date_allowed= datetime.date(1900, 1, 1),
            max_date_allowed= datetime.date.today(), #(2017, 9, 19)
            #initial_visible_month= datetime.date.today(),
            date= datetime.date.today(),
            display_format= "DD-MM-Y",
            style= {"display": "flex", "height": "20px"},),
    ], style={"display": "flex", "align-items": "baseline", "height": "25px",
               "justify-content": "space-around", "gap": "10px"}),

    html.Button(id='analyse-button', children= "Lancer l'analyse",
n_clicks=0,),

], style={'background-color': glob.fond_ecran_formulaire, "display": "flex",
"justify-content": "space-around", "align-items": "baseline"}),

```

```

    "#"flex-direction": "raw",
    "#"flex-wrap": "wrap", }

    html.P(children= ""),
    #html.P(children= ""),
    #html.Div(id='output-container-date-picker-single'),

    # Zone d'analyse
    html.Div([
        html.H2(children= "Veuillez remplir tous les champs du formulaire "+\
            "avant de lancer une analyse.", style= {"text-align" : "center"}),

        html.Div(id='container-button-basic', children= ""),
    ], style= {'background-color': glob.fond_ecran_reponse, "display": "flex",
    "flex-direction": "column", "justify-content": "space-between"})

], style= {'background-color': glob.fond_ecran_formulaire, "display": "flex",
    "flex-direction": "column", "justify-content": "space-between"}) #end div

#----- Callback -----
@app.callback(
    Output(component_id= 'container-button-basic', component_property= 'children'),
    Input('analyse-button', 'n_clicks'),
    State('ri-genre', 'value'), State("ri-activite", 'value'), State("ri-tabac", "value"),
    State("ri-alcool", "value"), State("dd-glucose", "value"), State("dd-cholesterol",
"value"),
    State("i-taille", "value"), State("i-poids", "value"), State("i-pasys", "value"),
    State("i-padia", "value"), State("dt-consultation", "date"), State("dt-naissance",
"date"),
    State("i-nom", "value"), State("i-prenom", "value"), State("dd-medecin", "value")
)
def update_output(n_clicks, rigenre, riactivite, ritabac, rialcool, ddglucose,
ddcholesterol, itaille, ipoids, ipasys, ipadia, dtconsultation, dtnaissance,
inom, iprenom, ddmedecin):
    cr= """**Compte rendu d'analyse:** \n"""
    if n_clicks > 0:
        if rigenre== None:
            return "Erreur: Veuillez renseigner le genre."
        else:
            cr+= "Homme, " if rigenre== 'h' else "Femme, "
            rigenre = 1 if rigenre== 'h' else 2

        if riactivite== None:
            return "Erreur: Veuillez renseigner l'activité."
        else:
            cr+= "pratiquant une activité physique, " if riactivite== 'act' \
                else "ne pratiquant pas d'activité physique, "
            riactivite = 1 if riactivite== 'act' else 0

        if ritabac== None:
            return "Erreur: Veuillez renseigner tabagisme."
        else:
            cr+= "fumeur(se), " if ritabac== 'fum' else "non fumeur(se), "
            ritabac = 1 if ritabac== 'fum' else 0

        if rialcool== None:
            return "Erreur: Veuillez renseigner la consommation d'alcool."
        else:
            cr+= "consommant de l'alcool, " if rialcool== 'alc' \
                else "ne consommant pas d'alcool, "
            rialcool = 1 if rialcool== 'alc' else 0

        if ddglucose== None:
            return "Erreur: Veuillez renseigner le niveau de glucose."
        else:
            if ddglucose== 1 :
                cr+= "avec un taux de glucose normal, "

```

```

elif ddglucose== 2 :
    cr+= "avec un taux de glucose au dessus de la normal, "
elif ddglucose== 3 :
    cr+= "avec un taux de glucose très au dessus de la normal, "
else:
    cr+= "TAUX DE GLUCOSE INNATENDU"

if ddcholesterol== None:
    return "Erreur: Veuillez renseigner le taux de cholesterol."
else:
    if ddcholesterol== 1 :
        cr+= "avec un taux de cholesterol normal, "
    elif ddcholesterol== 2 :
        cr+= "avec un taux de cholesterol au dessus de la normal, "
    elif ddcholesterol== 3 :
        cr+= "avec un taux de cholesterol très au dessus de la normal, "
    else:
        cr+= "TAUX DE CHOLESTEROL INNATENDU"

if itaille== None or itaille== "":
    return "Erreur: Veuillez renseigner la taille."
else:
    cr+= f"taille : {itaille} cm, "

if ipoids== None or ipoids== "":
    return "Erreur: Veuillez renseigner le poids."
else:
    cr+= f"poids : {ipoids} kg, "

if ipasys== None or ipasys== "":
    return "Erreur: Veuillez renseigner PA systolique."
else:
    cr+= f"pression systolique : {ipasys}, "

if ipadia== None or ipadia== "":
    return "Erreur: Veuillez renseigner PA diastolique."
else:
    cr+= f"pression diastolique : {ipadia}, "

if dtnaissance== None:
    return "Erreur: Veuillez renseigner la date de naissance."
else:
    cr+= f"né(e) le : {dtnaissance}, "

if dtconsultation== None:
    return "Erreur: Veuillez renseigner la date de consultation."
else:
    cr+= f"date de la consultation : {dtconsultation}. \n"

if inom== None:
    return "Erreur: Veuillez renseigner le nom du patient."
else:
    cr+= f"Mme/M (Nom prénom) {inom} "

if iprenom== None:
    return "Erreur: Veuillez renseigner le prénom du patient."
else:
    cr+= f"{iprenom}, "

if ddmedecin== None:
    return "Erreur: Veuillez renseigner le champ médecin prescripteur."
else:
    cr+= f"adressé par (numéro d'enregistrement, nom, prénom): {ddmedecin}. \n"

d1= datetime.datetime.strptime(dtnaissance, '%Y-%m-%d')
d2= datetime.datetime.strptime(dtconsultation, '%Y-%m-%d')

if (d2-d1).days== 0: # Test de la fonction alerte
    glob.message_erreur= "Test de la fonction d'alerte:"

```

```

msg_alerte= glob.alerte("Message dans prediction: âge du patient= 0")
return msg_alerte
else:
    donnees_patient= [(d2-d1).days, rigenre, itaille, ipoids, ipasys, ipadia,
                      ddcholesterol, ddglucose, ritabac, rialcool, riactivite]
    identite_patient= [inom, iprenom, d1]

#print(ddmedecin, " - ", type(ddmedecin))

# Le patient est-il dans la base?
idp= glob.patient(identite_patient= identite_patient)
if idp== None:
    msg_alerte = glob.alerte(message= "Erreur dans prediction. idp== None")
    return msg_alerte

# On récupère l'id du medecin ?
idmed= eval(ddmedecin.split('-')[0])

retour_analyse= glob.analyse(donnees_patient= donnees_patient)

if retour_analyse== None:
    msg_alerte = glob.alerte(message= "Erreur dans prediction.
retour_analyse== None")
    return msg_alerte
else: #Enregistrement en base
    #print("Retour analyse:", retour_analyse)
    cr+= "#### Résultats des analyses: \n"
    cr+= "Le nom des algorithmes est suivie de la probabilité d'avoir un
risque, "+\
          "pour ce patient, de maladie cardio-vasculaire.\n"

    # extraction du dictionnaire le la liste
    d= retour_analyse[11]
    d= d.replace("array","",)
    d= d.replace(")", "")
    d= ast.literal_eval(d)
    tab_score=[]
    for c,v in d.items():
        if c== "score": continue
        tab_score.append(v[1])
        cr+= f"- {c} : {100*v[1]:.2f} %\n"
    moyenne= sum(tab_score)/len(tab_score)
    cardio= 0 if moyenne < 0.5 else 1
    ret= glob.enregistrement(retour_analyse= retour_analyse, idpatient= idp,
                           idmedecin= idmed, cardio= cardio)
    #ret= None # Test fonction alerte
    if ret== None:
        msg_alerte = glob.alerte(message= "Erreur retour enregistrement.
ret== None")
        return msg_alerte

    # cr= f"{glob.message_erreur} {n_clicks}, {rigenre}, {riactivite}, {ritabac},
{rialcool}, {ddglucose}, "+\
        "#     f'{ddcholesterol}, {itaille}, {ipoids}, {ipasys}, {ipadia}, "
{dtconsultation}, "+\
        "#     f'{dtnaissance} - {(d2-d1).days}, {idp}, {idmed}"

return dcc.Markdown(cr)

```

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        data.py; page data. Ce fichier doit se situer dans le sous-répertoire /apps

@auteur: jpphi
"""

from dash import dcc, html, dash_table
import dash_bootstrap_components as dbc

from apps import navbar2, glob

largeur= '1200px'

dfex= glob.df_examen[0:1000]
dfpat= glob.df_patient[0:1000]

tab_examen= dash_table.DataTable(id='container-button-timestamp',
    data= dfex.to_dict('records'),
    columns=[{'id': c, 'name': c} for c in glob.df_examen.columns],
    export_format='csv',
    style_header={'backgroundColor': 'rgb(30, 30, 30)', "color" : glob.couleur_ecrit},
    style_table={'overflowX': 'auto',
                 'width' : largeur,
                 'height': '400px'},
    style_cell={
        'backgroundColor': 'rgb(50, 50, 50)',
        'color': 'white',
        'textAlign':'left',
        'padding-left': '5px'
    }
)

tab_medecin= dash_table.DataTable(id='container-button-timestamp',
    data= glob.df_medecin.to_dict('records'),
    columns=[{'id': c, 'name': c} for c in glob.df_medecin.columns],
    export_format='csv',
    style_header={'backgroundColor': 'rgb(30, 30, 30)', "color" : glob.couleur_ecrit},
    style_table={'overflowX': 'auto',
                 'width' : largeur,
                 'height': '400px'},
    style_cell={
        'backgroundColor': 'rgb(50, 50, 50)',
        'color': 'white',
        'textAlign':'left',
        'padding-left': '5px'
    }
)

tab_patient= dash_table.DataTable(id='container-button-timestamp',
    data= dfpat.to_dict('records'),
    columns=[{'id': c, 'name': c} for c in glob.df_patient.columns],
    export_format='csv',
    page_size= 10,
    style_header={'backgroundColor': 'rgb(30, 30, 30)', "color" : glob.couleur_ecrit},
    style_table={'overflowX': 'auto',
                 'width' : largeur,
                 'height': '400px'},
    style_cell={
        'backgroundColor': 'rgb(50, 50, 50)',
        'color': 'white',
        'textAlign':'left',
    }
)

```

```

        'padding-left': '5px'
    }
)

tab_diag= dash_table.DataTable(id='container-button-timestamp',
    data= glob.df_diagnostique.to_dict('records'),
    columns=[{'id': c, 'name': c} for c in glob.df_diagnostique.columns],
    export_format='csv',
    page_size= 10,
    style_header={'backgroundColor': 'rgb(30, 30, 30)', "color" : glob.couleur_ecrit},
    style_table={'overflowX': 'auto',
                 'width' : largeur,
                 'height': '400px'},
    style_cell={
        'backgroundColor': 'rgb(50, 50, 50)',
        'color': 'white',
        'textAlign':'left',
        'padding-left':'5px'
    }
)

layout = html.Div([
    navbar2.layout,
    html.Div([
        html.H2(children= "Base de donnée - données brutes", style= {"text-align": "center"}),
        html.P(""),
        dcc.Markdown(children= """Cette base comporte plusieurs tables.  

- La table examen: Comporte l'ensembles des données cliniques ainsi que l'identifiant du patient ainsi que l'identifiant du medecin.  

- La table medecin: Le nom des médecins prescripteurs.  

- La table patient: Le nom des patients ayant participant à l'étude.  

- La table diagnostique: Les différents diagnostiques réalisés"""),
        html.P(""),
        html.Div([
            dbc.Tabs([
                dbc.Tab(tab_examen, label="Examen", label_style={"color": "#e5e7e9"}),
                dbc.Tab(tab_medecin, label="Medecins", label_style={"color": "#fefefe"}),
                dbc.Tab(tab_patient, label="Patients", label_style={"color": "#dbdbdb"}),
                dbc.Tab(tab_diag, label="Diagnostiques", label_style={"color": "#b9b9b9"}),
            ], style={"display": "flex", "align-items": "center", "align-content": "center"},),
            ], style={"display": "flex", "flex-direction": "column", "justify-content": "center", "align-items": "center", "align-content": "center"},),
        html.P(),
        dcc.Link("Page d'accueil", href = '/success'),
    ]),
], style={'background-color': glob.fond_ecran_formulaire, "display": "flex", "flex-direction": "column"}) #end div

```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        failed.py; page failed. Ce fichier doit se situer dans le sous-répertoire /apps

@auteur: jpphi
"""

from dash import dcc, html
import dash_bootstrap_components as dbc

from apps import login, glob

layout = html.Div([dcc.Location(id='url_login_df', refresh=True),
    html.Div([
        html.H1(children= "Reconnection ?", style= {"text-align": "center"}),
        html.P(""),

        html.H4("L'utilisateur n'existe pas, ou le mot de passe n'est pas "+\
            "correct...", style= {"text-align": "center"}),
        html.P(""),

        html.H4("Même joueur, joue encore !", style= {"text-align": "center"}),
        html.P(""),

        html.Div([login.layout]),

    ]) #end div
], style= {'background-color': glob.fond_ecran_formulaire, "display": "flex",
    "flex-direction": "column"}) #end div
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        logout.py; page logout. Ce fichier doit se situer dans le sous-répertoire /apps

@auteur: jpphi
"""

from dash import html, dcc
from apps import login, glob

layout = html.Div([
    dcc.Location(id='logout', refresh=True),

    html.Div(html.H2('Vous êtes déconnecté', style= {"text-align": "center"})),

    html.P(""),

    html.Div(login.layout),
], style={'background-color': glob.fond_ecran_formulaire, "display": "flex",
"flex-direction": "column","justify-content": "space-between", })
```

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Serveur support de l'application d'aide au diagnostique sur les maladies cardio-
    vasculaires
        glob.py; Fichier contenant les fonctions et variables globales.
            Ce fichier doit se situer dans le sous-répertoire /apps

@auteur: jpphi
"""

import os
from joblib import load
import sqlite3
import pandas as pd

import dash_bootstrap_components as dbc
from dash import html
from dash.dependencies import Input, Output, State

from app import app

# Couleurs...
Bleu_de_minuit= "#003366"
Bleu_marine= "#03224C"
Bleu_nuit= "#0F055B"
Bleu_outremer= "#15019B"
Bleu_outremer2= "#2B009A"

fond_ecran_reponse= "#007B94"
fond_ecran_formulaire= Bleu_de_minuit

couleur_ecrit= "#e5e7e9"
# base de données
bdd= "/home/jpphi/Documents/brief/ProjetFinDEtude/E1/datas/base_E1.db"
#bdb= "/home/jpphi/Documents/brief/ProjetFinDEtude/E1/datas/base_E1.db"

df_examen= None
df_medecin= None
df_patient= None
df_diagnostique= None

repertoire= "/home/jpphi/Documents/brief/ProjetFinDEtude/E1/modeles/"

message_erreur= "" #TEST DE LA FONCTIONNALITE ALERTE

def alerte(message= None):
    global message_erreur
    #print(message, " - ",message_erreur)

    aff_alerte= dbc.Alert([
        html.H4(children= "Une erreur est survenu !",
                style= {"text-align": "center"}),
        html.H5(children= f"Le message envoyé par le code est:
{message}"),
        html.P(),
        html.H5(children= f"Dans glob.py on a le message:
{message_erreur}"),
        html.Div([
            html.Button(id='email-button', children= "Envoyer le
rapport d'erreur par mail"
                        , n_clicks=0,style= {"text-align": "center",
"width": "300px"}),
            ], style={"display": "flex", "justify-content": "center",
"flex-direction": "row"}),
    ],

```

```

        ], color="danger" )
    return aff_alerte

def analyse(donnees_patient= None):
    global message_erreur

    # Analyse
    try:
        dict_algo_fichier = charge_modeles(repertoire= repertoire)
        algos = list(dict_algo_fichier.keys())
        fichiers= list(dict_algo_fichier.values())

        ch= {}
        score=0
        nb=0
        for f in fichiers:
            nb+= 1
            mod= load(repertoire+f"{f}")
            score+= mod.predict([donnees_patient])[0]

            prediction = mod.predict_proba([donnees_patient])
            ch[f]= prediction[0]
        ch["score"]= f"{score}/{nb}"
        #print(f"dp: {donnees_patient}, ch: {ch}")
        donnees_patient.append(str(ch))
        #print(f"dp + ch: {donnees_patient}")
        return donnees_patient
    except:
        message_erreur= "Fonction analyse: except"
        return None

def charge_modeles(repertoire= None):
    """
    Retourne un dictionnaire dont la clé est le nom de l'algorithme et la valeur
    associée à la clé le nom du modèle binaire à charger.
    Entrée:
        nom du répertoire contenant les fichiers modeles
    Sortie:
        le dictionnaire{algo:nom_du_fichier}
    """
    assert type(repertoire)== str, "Le paramètre repertoire doit être de "+\
        "type chaîne de caractères"
    liste_fichiers= os.listdir(repertoire)
    # On ne récupère que les fichiers d'extensions .h5
    liste_fichiers = [f for f in liste_fichiers if ".h5" in f]
    # Recherche du nom de l'algorithme contenue dans le nom fichier
    # Les fichiers doivent être écrit avec la syntaxe: algo_caractéristique.h5
    algo= [a.split("_")[0] for i,a in enumerate(liste_fichiers)]
    return {cle: valeur for cle, valeur in zip(algo, liste_fichiers)}

def creation_patient(identite_patient= None):
    global message_erreur

    #print(identite_patient)
    try:
        # Connexion à la base de donnée
        conn = sqlite3.connect(bdd)
        cur = conn.cursor()

        sql = "INSERT INTO patient (nom, prenom, naissance) VALUES(?, ?, ?)"
        #print(f"identite_patient: {identite_patient}")
        cur.execute(sql, (identite_patient[0].upper(),
        identite_patient[1].lower(),
                    identite_patient[2]) )
        conn.commit()

        cur.close()
        conn.close()
    
```

```

# Le patient est créé, on récupère son id
idpatient= recherche_id(identite_patient= identite_patient)

        return idpatient
except sqlite3.Error as error:
    message_erreur= f"Erreur dans la fonction creation_patient"
    return None

def enregistrement(retour_analyse= None, idpatient= None, idmedecin= None, cardio= None):
    global message_erreur

    try:
        # Connexion à la base de donnée
        conn = sqlite3.connect(bdd)
        cur = conn.cursor()

        sql = "INSERT INTO diagnostique (age, gender, height, weight, ap_hi,
ap_lo, "+\
                "cholesterol, gluc, smoke, alco, active, resultat_diag, idp,
idmed, cardio) "+\
                "VALUES(?,?,?,?,?,?,?,?,?, ?,?, ?, ?, ?, ?, ?)"

        ligne= tuple(el for el in retour_analyse)

        cur.execute(sql, (ligne[0], ligne[1], ligne[2], ligne[3], ligne[4],
ligne[5],
                ligne[6], ligne[7], ligne[8], ligne[9], ligne[10], ligne[11],
idpatient, idmedecin, cardio))
        conn.commit()

        cur.close()
        conn.close()
        return 1
    except sqlite3.Error as error:
        message_erreur= f"Erreur dans enregistrement lors de la connexion à
SQLite: {error}"
        return None

def ouverture_base_medecin():
    global message_erreur

    try:
        conn = sqlite3.connect(bdd)
        cur = conn.cursor()

        query = conn.execute("SELECT * FROM medecin")
        cols = [column[0] for column in query.description]

        df_medecin= pd.DataFrame.from_records(data = query.fetchall(), columns =
cols)
        return 1
    except:
        message_erreur="Fonction ouverture_base_medecin: Impossible de lire la
table"
        return None

def patient(identite_patient= None):
    global message_erreur

    #print("patient ", identite_patient)

    # Recherche id patient. S'il n'existe pas, crée le patient
    idp= recherche_id(identite_patient= identite_patient)
    if idp== None:
        message_erreur= "Problème dans la fonction recherche_id"
        return None
    elif idp== 0: # création d'un nouveau patient

```

```

idp= creation_patient(identite_patient= identite_patient)
    if idp== None:
        message_erreur= "fonction patient, échec création patient."
        return None
    return idp

def recherche_id(identite_patient= None):
    global message_erreur

    #print(identite_patient)
    try:
        # Connexion à la base de donnée
        conn = sqlite3.connect(bdd)
        cur = conn.cursor()

        cur.execute("SELECT idp FROM patient WHERE nom= ? AND prenom= ? AND
naissance= ?",
                    ((identite_patient[0]).upper(),identite_patient[1].lower(),
                     identite_patient[2]))

        idp= cur.fetchall()
        n= len(idp)
        cur.close()
        conn.close()
    except sqlite3.Error as error:
        message_erreur= f"Erreur dans la fonction recherche_id lors de la "+\
                      f"connexion à SQLite: {error}"
        return None

    if n== 0: # le patient n'existe pas
        idp= []
    elif n > 1: # plusieurs patients avec même nom, prénom et date de naissance
        message_erreur= "Erreur dans la fonction recherche_id. Patient enregistré
"+\
                      f"plusieur fois: {idp}"
        return None
    else:
        return int( (idp[0])[0] )

@app.callback(
    Output(component_id= 'container-email', component_property= 'children'),
    Input('email-button', 'n_clicks'))
def update_output(n_clicks):
    if n_clicks > 0:
        return

```

Rapport E1, Étude

Fonctions et import

```
Entrée [1]: import numpy as np
import pandas as pd
from pandas_profiling import ProfileReport

import sqlite3

import random
import os

from joblib import dump, load

import datetime
from datetime import date, timedelta
import time

from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB, BernoulliNB
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, BaggingClassifier, AdaBoostClassifier
from sklearn.ensemble import VotingClassifier, StackingClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.tree import DecisionTreeClassifier

#from sklearn.metrics import accuracy_score, classification_report, r2_score
#from tensorflow import keras

import seaborn as sn
import matplotlib.pyplot as plt
# Pour utiliser le module inline 'module://matplotlib_inline.backend_inline'
# D'autre backend peuvent être utilisée
%matplotlib inline
```

```
Entrée [2]: class Faussaire:
    _prenoms_csv= "datas/Prenoms.csv"
    _filtre_prenoms= 1
    _noms_csv= "datas/patronymes.csv"
    _freq_nom= 120
    _liste_prenoms_courant_homme= None
    _liste_prenoms_courant_femme= None
    _liste_noms_filtre= None
    _prenoms= None
    _noms= None
    _long_liste_fem= 0
    _long_liste_hom= 0
    _long_liste_nom= 0

    def __init__(self):
        # Prénoms h et f
        self._prenoms= pd.read_csv(self._prenoms_csv, sep= ';', encoding="latin1") #, encoding="latin1"
        prenoms_courant= self._prenoms[self._prenoms["04_fréquence"] > self._filtre_prenoms]

        prenoms_courant_femme= prenoms_courant[ (prenoms_courant["02_genre"]=="f") |
                                                (prenoms_courant["02_genre"]=="m,f") |
                                                (prenoms_courant["02_genre"]=="f,m")]
        self._liste_prenoms_courant_femme= list(prenoms_courant_femme["01_prenom"])
        self._long_liste_fem= len(self._liste_prenoms_courant_femme)

        prenoms_courant_homme= prenoms_courant[ (prenoms_courant["02_genre"]=="m") |
                                                (prenoms_courant["02_genre"]=="m,f") |
                                                (prenoms_courant["02_genre"]=="f,m")]
        self._liste_prenoms_courant_homme= list(prenoms_courant_homme["01_prenom"])
        self._long_liste_hom= len(self._liste_prenoms_courant_homme)

        # Noms
        self._noms= pd.read_csv(self._noms_csv)
        noms_filtre= self._noms[self._noms["count"]>self._freq_nom]
        self._liste_noms_filtre= list(noms_filtre["patronyme"])
        self._long_liste_nom= len(self._liste_noms_filtre)

    def nom_prenom(self, genre= None):
        assert genre== 'f' or genre== 'h', "ERREUR de valeur: Le genre doit être 'f' ou 'h'"

        #try:
        i= random.randint(0, self._long_liste_nom - 1)
        nom= self._liste_noms_filtre[i]

        if genre== 'f':
            j= random.randint(0, self._long_liste_fem - 1)
            prenom= self._liste_prenoms_courant_femme[j]
        else:
            j= random.randint(0, self._long_liste_hom - 1)
            prenom= self._liste_prenoms_courant_homme[j]

        return nom, prenom#, len(self._liste_noms_filtre)
```

```

    def voir_nom_i(self, i):
        return self.__liste_noms_filtre[i]

    def voir_df_noms(self):
        return self.__noms

    def voir_df_prenoms(self):
        return self.__prenoms

Entrée [3]: def erreur_prediction(valeurs_predites= None, valeurs_reelle= None):
    """
    Vérifie que les valeurs prédites sont les mêmes que les valeurs réelles.

    paramètres en entrée:
    valeurs_predites
    valeurs_reelle

    Paramètres de sortie:
    erreur: nombre d'erreur de prédiction
    precision: 100*(1-erreur/(longueur des tableaux))
    liste_erreur: Liste de toutes les erreurs détecté sous forme de dictionnaire
        clé valeur réelle: valeur prédite
    """

    assert len(valeurs_predites) == len(valeurs_reelle), "Erreur de dimension: la dimension des tableaux "+\
        "f doit être identique. {len(valeurs_predites)} # de {len(valeurs_reelle)}"

    erreur= 0
    d= {}
    longueur= len(valeurs_reelle)
    for i in range(longueur):
        if valeurs_reelle[i]!= valeurs_predites[i]:
            if valeurs_reelle[i] not in d.keys():
                d[valeurs_reelle[i]]= [valeurs_predites[i]]
            else:
                d[valeurs_reelle[i]].append(valeurs_predites[i])
            erreur+= 1

    return erreur, 100*(1-erreur/longueur), d

```

```

Entrée [4]: def test_algo(modele= None, standardisation= None, Xtrain= None, Ytrain= None, Xtest= None, Ytest= None):
    model= modele
    if standardisation!= None:
        Xtrain_normalise= standardisation.fit_transform(X_train)
        X_test_normalise= standardisation.transform(X_test)
    else:
        Xtrain_normalise= Xtrain
        X_test_normalise= Xtest

    model.fit(Xtrain_normalise, Ytrain)

    return model.score(X_test_normalise,Ytest)

```

```

Entrée [5]: def charge_modeles(repertoire= None):
    """
    Retourne un dictionnaire dont la clé est le nom de l'algorithme et la valeur
        associée à la clé le nom du modèle binaire à charger.

    Entrée:
        nom du répertoire contenant les fichiers modeles
    Sortie:
        le dictionnaire{algo:nom_du_fichier}
    """
    assert type(repertoire)== str, "Le paramètre repertoire doit être de "+\
        "type chaîne de caractères"

    liste_fichiers= os.listdir(repertoire)
    # On ne récupère que les fichiers d'extensions .h5
    liste_fichiers = [f for f in liste_fichiers if ".h5" in f]
    # Recherche du nom de l'algorithme contenu dans le nom fichier
    # Les fichiers doivent être écrit avec la syntaxe: algo_caractéristique.h5
    algo= [a.split("_")[0] for i,a in enumerate(liste_fichiers)]
    return {cle: valeur for cle, valeur in zip(algo, liste_fichiers)}

```

Importation des données

Pour étudier le contenu du dataset, on va supprimer la colonne id et rajouter une colonne IMC pour traquer les valeur abérantes.

```

Entrée [8]: df= pd.read_csv("./datas/cardio_train.csv",sep= ";", index_col= "id")

# Création de l'IMC qui nous servira pour la détection de valeurs aberrantes
df["IMC"]= df.weight/((df.height/100)**2)

df.head(10)

```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	IMC
id													
0	18393	2	168	62.0	110	80	1	1	0	0	1	0	21.967120
1	20228	1	156	85.0	140	90	3	1	0	0	1	1	34.927679
2	18857	1	165	64.0	130	70	3	1	0	0	0	1	23.507805
3	17623	2	169	82.0	150	100	1	1	0	0	1	1	28.710479
4	17474	1	156	56.0	100	60	1	1	0	0	0	0	23.011177

```

age gender height weight ap_hi ap_lo cholesterol gluc smoke alco active cardio      IMC
id
8 21914     1   151   67.0   120   80       2   2   0   0   0   0 29.384676
9 22113     1   157   93.0   130   80       3   1   0   0   1   0 37.729725
12 22584    2   178   95.0   130   90       3   3   0   0   1   1 29.983588

```

Entrée [11]: `analyse=ProfileReport(df)`
`analyse.to_file(output_file="cardio2.html")`

Recherche d'éventuel doublon

Entrée [12]: `len(df[df.duplicated(subset=['age', 'gender','height', 'weight', 'ap_hi', 'ap_lo',
 'cholesterol','gluc', 'smoke', 'alco', 'active','cardio'])]== True])`

Out[12]: 24

Entrée [13]: `df[df.duplicated(subset=['age', 'gender','height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol','gluc',
 'smoke', 'alco', 'active','cardio'], keep=False)== True].sort_values(by = ['age'])`

Out[13]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	IMC
	id												
57690	14552	1	158	64.0	120	80		1	1	0	0	1	0 25.636917
9004	14552	1	158	64.0	120	80		1	1	0	0	1	0 25.636917
91592	16160	1	168	65.0	120	80		1	1	0	0	1	1 23.030045
24435	16160	1	168	65.0	120	80		1	1	0	0	1	1 23.030045
1685	16793	1	165	68.0	120	80		1	1	0	0	1	0 24.977043
31110	16793	1	165	68.0	120	80		1	1	0	0	1	0 24.977043
40450	16805	1	157	67.0	120	80		1	1	0	0	1	0 27.181630
86345	16805	1	157	67.0	120	80		1	1	0	0	1	0 27.181630
63776	16937	2	170	70.0	120	80		1	1	0	0	0	0 24.221453
14974	16937	2	170	70.0	120	80		1	1	0	0	0	0 24.221453
1585	17493	2	169	74.0	120	80		1	1	0	0	1	1 25.909457
71971	17493	2	169	74.0	120	80		1	1	0	0	1	1 25.909457
80859	17535	2	165	65.0	120	80		1	1	0	0	1	0 23.875115
46667	17535	2	165	65.0	120	80		1	1	0	0	1	0 23.875115
92891	18210	1	160	60.0	120	80		1	1	0	0	1	0 23.437500
41436	18210	1	160	60.0	120	80		1	1	0	0	1	0 23.437500
32416	18353	1	169	67.0	120	80		1	1	0	0	1	0 23.458562
60642	18353	1	169	67.0	120	80		1	1	0	0	1	0 23.458562
78064	18955	1	165	75.0	120	80		1	1	0	0	1	1 27.548209
82698	18955	1	165	75.0	120	80		1	1	0	0	1	1 27.548209
68156	18979	1	165	65.0	120	80		1	1	0	0	0	0 23.875115
88987	18979	1	165	65.0	120	80		1	1	0	0	0	0 23.875115
15414	18988	1	164	65.0	120	80		1	1	0	0	1	0 24.167162
54977	18988	1	164	65.0	120	80		1	1	0	0	1	0 24.167162
28624	19059	1	165	65.0	120	80		1	1	0	0	1	1 23.875115
94486	19059	1	165	65.0	120	80		1	1	0	0	1	1 23.875115
83812	19858	1	165	68.0	120	80		1	1	0	0	1	0 24.977043
23334	19858	1	165	68.0	120	80		1	1	0	0	1	0 24.977043
2283	20293	1	162	70.0	110	70		1	1	0	0	1	0 26.672763
81232	20293	1	162	70.0	110	70		1	1	0	0	1	0 26.672763
3247	20495	1	165	70.0	120	80		1	1	0	0	1	0 25.711662
15094	20495	1	165	70.0	120	80		1	1	0	0	1	0 25.711662
34798	20516	1	164	66.0	120	80		1	1	0	0	0	0 24.538965
86312	20516	1	164	66.0	120	80		1	1	0	0	0	0 24.538965
55102	21119	1	160	60.0	120	80		1	1	0	0	0	1 23.437500
97521	21119	1	160	60.0	120	80		1	1	0	0	0	1 23.437500
65438	21230	1	164	62.0	120	80		1	1	0	0	1	0 23.051755
24539	21230	1	164	62.0	120	80		1	1	0	0	1	0 23.051755
64445	21280	1	165	65.0	120	80		1	1	0	0	1	0 23.875115
31244	21280	1	165	65.0	120	80		1	1	0	0	1	0 23.875115
11684	21778	1	160	58.0	120	80		1	1	0	0	1	0 22.656250
93659	21778	1	160	58.0	120	80		1	1	0	0	1	0 22.656250
57602	21943	1	165	65.0	120	80		1	1	0	0	1	1 23.875115
74969	21943	1	165	65.0	120	80		1	1	0	0	1	1 23.875115
69842	21945	1	165	60.0	120	80		1	1	0	0	1	0 22.038567
2223	21945	1	165	60.0	120	80		1	1	0	0	1	0 22.038567

```
age gender height weight ap_hi ap_lo cholesterol gluc smoke alco active cardio IMC
```

```
Entrée [14]: # Suppression des doublons
df.drop_duplicates(subset= ['age', 'gender','height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol','gluc',
'smoke', 'alco', 'active','cardio'], keep='first', inplace= True)
```

```
Entrée [15]: cte_age_an_jour= 365.242199
print(f"Dans ce jeu de donnée, le patient le plus jeune à {int(10798/cte_age_an_jour)} ans et le "+\
f"plus âgé {int(23713/cte_age_an_jour)} ans avec une moyenne d'âge de {int(19468/cte_age_an_jour)} ans.")
```

Dans ce jeu de donnée, le patient le plus jeune à 29 ans et le plus âgé 64 ans avec une moyenne d'âge de 53 ans.

Recherche de valeurs abbérantes:

Il y a un certains nombre de valeurs abbérantes qu'il convient de traiter.

Exemple:

- Enregistrement 9223, âge 21220, genre 1, taille 250
- Enregistrement 82567, âge 18804, genre 2, poids 10.0 pression artérielle 180/1100
- Enregistrement 11089, pression artérielle 11500
- Enregistrement 22108, pression artérielle négative (-115)

On va supprimer ces enregistrements

Pour supprimer les valeur abbérantes poids / taille on créer la colonne IMC.

IMC	Interprétation
+ de 40	obésité morbide ou massive
35 à 40	obésité sévère
30 à 35	obésité modérée
25 à 30	surpoids
18.5 à 25	corpulence normale
16.5 à 18.5	maigreur
- de 16.5	famine

```
Entrée [16]: df[ (df.IMC>40)&(df.cardio== 0)&(df.active== 0)]
```

```
Out[16]:   age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  active  cardio  IMC
      id
  618  16765     1    186  200.0    130     70          1     1     0     0     0     0     0  57.810151
  846  14705     1    164  125.0    130     90          1     1     0     0     0     0     0  46.475312
 1378  21897     1    154  110.0    110     70          1     1     0     0     0     0     0  46.382189
 1721  15126     2    173  125.0    120     80          1     1     1     0     0     0     0  41.765512
 1965  22644     1    166  131.0    120     80          3     3     0     0     0     0     0  47.539556
...
 87816 17375     1    162  110.0    110     60          1     3     0     0     0     0     0  41.914342
 89887 17325     1    165  115.0    120     80          1     1     0     0     0     0     0  42.240588
 91523 18426     1     59   57.6    125     67          1     1     0     0     0     0     0  165.469693
 95311 21182     1    157  126.0    140     90          3     3     0     0     0     0     0  51.117692
 96955 21199     1    159  112.0    130     90          1     3     0     0     0     0     0  44.302045
```

113 rows × 13 columns

```
Entrée [17]: df[ (df.IMC<15)&(df.cardio== 0)&(df.active== 0) ]
```

```
Out[17]:   age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  active  cardio  IMC
      id
 31869  21228     1    169   42.0    150     90          2     2     0     0     0     0     0  14.705367
 90183  22192     2    198   58.0    110     70          1     1     0     0     0     0     0  14.794409
```

```
Entrée [18]: df[ (df.ap_hi<5)|(df.ap_hi>300) ]
```

```
Out[18]:   age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  active  cardio  IMC
      id
 2654  15116     1    160   60.0    902     60          1     1     0     0     1     0  23.437500
 2845  22712     2    167   59.0    906     0          1     1     0     0     1     0  21.155294
 6525  15281     1    165   78.0   -100     80          2     1     0     0     1     0  28.650138
 6822  14425     1    168   63.0    909     60          2     1     0     0     1     0  22.321429
 11089 21032     1    175   80.0   11500    90          1     1     0     0     1     1  26.122449
 12494 16905     2    163   63.0     1  2088          1     1     1     0     1     0  23.711845
 12710 18870     1    164   75.0   1420     80          2     1     0     0     1     1  27.885187
 13616 22659     1    155   87.0    701    110          1     1     0     0     1     1  36.212279
 19827 15996     1    168   72.0   1500     80          1     1     0     0     1     1  25.510204
 22881 22108     2    161   90.0   -115     70          1     1     0     0     1     0  34.720883
```

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	IMC
id													
25314	22398	2	163	50.0	907	70	3	3	0	0	1	1	18.818924
29313	15581	1	153	54.0	-100	70	1	1	0	0	1	0	23.068051
34120	16131	1	161	92.0	906	0	2	1	0	0	1	1	35.492458
34295	18301	1	162	74.0	-140	90	1	1	0	0	1	1	28.196921
36025	14711	2	168	50.0	-120	80	2	1	0	0	0	1	17.715420
36339	15835	2	169	75.0	14020	80	2	1	0	0	1	1	26.259585
36414	21361	1	169	71.0	14020	80	3	3	0	0	1	1	24.859074
36793	18304	1	157	83.0	1400	80	1	1	0	0	1	1	33.672766
40239	19700	2	175	87.0	1620	80	2	1	0	0	1	1	28.408163
42410	17548	1	154	65.0	907	70	1	1	0	0	1	0	27.407657
45400	16070	1	170	64.0	907	0	1	1	0	0	1	0	22.145329
50055	23325	2	168	59.0	-150	80	1	1	0	0	1	1	20.904195
52725	20612	2	175	78.0	1130	90	1	1	0	0	1	1	25.469388
57646	20322	1	162	50.0	309	0	1	1	0	0	1	0	19.051974
58349	19806	1	162	67.0	401	80	1	3	0	0	1	1	25.529645
58374	17438	1	169	70.0	16020	80	1	1	0	0	0	1	24.508946
58728	21117	1	160	60.0	1202	80	1	1	0	0	1	1	23.437500
59301	20970	1	154	41.0	806	0	1	1	0	0	1	0	17.287907
60477	18716	1	171	80.0	1	1088	1	1	0	0	1	1	27.358845
60565	17988	2	176	69.0	906	0	1	1	0	0	1	0	22.275310
60948	20456	2	182	80.0	906	60	1	1	0	0	1	1	24.151673
61618	20961	2	170	78.0	1400	90	2	1	0	0	1	0	26.989619
61725	23418	1	165	67.0	1420	80	2	1	0	0	1	1	24.609734
62154	19534	1	158	62.0	1300	80	3	1	0	1	1	1	24.835763
66571	23646	2	160	59.0	-120	80	1	1	0	0	0	0	23.046875
66998	16910	2	180	78.0	14020	90	1	1	0	0	1	1	24.074074
67502	19731	1	160	65.0	14020	90	1	1	0	0	1	0	25.390625
69672	21867	1	156	76.0	1400	90	1	1	0	0	1	1	31.229454
72539	16918	2	164	66.0	1409	90	1	1	0	0	1	1	24.538965
73356	18784	2	168	65.0	11020	80	1	1	0	0	1	1	23.030045
77010	18511	1	164	54.0	960	60	1	1	0	0	1	0	20.077335
79116	18307	1	152	76.0	13010	80	2	2	0	0	1	1	32.894737
79679	23182	1	161	105.0	13010	80	1	1	0	0	0	0	40.507696
81769	21948	2	166	73.0	1300	90	1	1	0	1	1	0	26.491508
82660	23264	1	153	63.0	1110	80	1	1	0	0	0	1	26.912726
91364	16929	1	168	69.0	1205	90	1	1	0	0	0	1	24.447279
92655	16674	1	157	78.0	906	60	2	1	0	0	1	0	31.644286

Entrée [19]: df[(df.ap_lo<2) |(df.ap_lo>300)]

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	IMC
id													
314	17489	2	183	98.0	160	1100	1	2	1	0	1	1	29.263340
334	21932	2	157	60.0	160	1000	2	1	0	0	0	1	24.341758
357	18217	1	150	83.0	140	800	1	1	0	0	1	1	36.888889
458	23407	1	176	63.0	160	1000	2	2	0	0	0	1	20.338326
482	18704	1	154	81.0	140	1000	2	1	0	0	1	1	34.154158
...
99659	23330	1	167	81.0	160	1000	1	1	0	0	1	1	29.043709
99798	21808	1	152	56.0	160	1000	1	1	0	0	1	1	24.238227
99807	21239	2	168	95.0	160	1000	1	1	0	0	1	1	33.659297
99816	22417	2	166	78.0	170	1000	1	1	0	0	0	0	28.305995
99955	21416	2	168	63.0	140	1000	1	1	0	0	1	1	22.321429

976 rows × 13 columns

Enregistrement du dataset sans les doublons

Entrée [20]: df.to_csv("prov.csv", encoding="utf-8", sep=",", index=False)

Création d'une base de donnée analytique

On va créer une base de données qui contiendra toute les données filtrées du dataset

```
Entrée [21]: bd= 'base_E1_train.db'
# Ne pas ré-exécuter ce code une fois la base créée
"""
try:
    # Création de la base de donnée
    conn = sqlite3.connect(bd)
    print(f"Création de la base {bd} réussie")
except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

Création de la base base_E1_train.db réussie
```

```
Entrée [24]: # Ne pas ré-exécuter ce code une fois la base créée - Création de la table entraînement
"""
try:
    conn = sqlite3.connect(bd)
    cur = conn.cursor()

    sql = "CREATE TABLE entraînement ( "+\
        "id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "+\
        "age INTEGER NOT NULL, "+\
        "gender INTEGER, "+\
        "height INTEGER NOT NULL, "+\
        "weight INTEGER NOT NULL, "+\
        "ap_hi INTEGER NOT NULL, "+\
        "ap_lo INTEGER NOT NULL, "+\
        "cholesterol INTEGER, "+\
        "gluc INTEGER, "+\
        "smoke INTEGER, "+\
        "alco INTEGER, "+\
        "active INTEGER, "+\
        "cardio INTEGER );"
    cur.execute(sql)
    conn.commit()
    cur.close()
    conn.close()
    print("Table créée.")

except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

Table créée.
```

Filtrage des données

```
Entrée [25]: df= pd.read_csv("prov.csv")

# Suppression de valeur aberrante IMC
# On supprime les IMC<40 qui ne font pas de sport et noté sans risque cardio
df= df[ (df.IMC<40)|(df.cardio== 1)|(df.active== 1) ]
# On supprime les IMC>15 qui ne font pas de sport et noté sans risque cardio
df= df[ (df.IMC>15)|(df.cardio== 1)|(df.active== 1) ]

# suppression valeurs aberrantes des pressions artérielles
df= df[ (df.ap_hi>5)&(df.ap_hi<300) ]
df= df[ (df.ap_lo>2)&(df.ap_lo<300) ]

# suppression de la colonne IMC qui n'appartient pas au dataset original
df.drop("IMC", axis= 1, inplace= True)
```

Intégrations des données dans la table entraînement de la base créée

```
Entrée [26]: # Ne pas ré-exécuter ce code une fois la table remplie
"""
try:
    # Création de la base de donnée
    conn = sqlite3.connect(bd)
    cur = conn.cursor()

    sql = "INSERT INTO entraînement (age, gender, height, weight, ap_hi, ap_lo, cholesterol, "+\
        "gluc, smoke, alco, active, cardio) VALUES(?,?,?,?,?,?,?,?,?,?,?,?)"

    for i, row in df.iterrows():
        cur.execute(sql, tuple(row))
        conn.commit()

    cur.close()
    conn.close()
    print("La connexion SQLite est fermée")
except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

La connexion SQLite est fermée
```

Création d'une base de donnée de production

On va créer une base de données qui contiendra plusieurs tables

- Table patient

- Table médecin
- Table examens
- Table diagnostique

Entrée [8]: `bdp= './datas/base_E1.db'`

```
Entrée [28]: # Ne pas ré-exécuter ce code une fois la base créée
"""
try:
    # Création de la base de donnée
    conn = sqlite3.connect(bdp)
    print(f"Création de la base {bdp} réussi")
except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

Création de la base ./datas/base_E1.db réussi
```

Table des patients

Entrée [29]: # Ne pas ré-exécuter ce code une fois la base créée - Crédation de la table patient

```
"""
try:
    conn = sqlite3.connect(bdp)
    cur = conn.cursor()

    sql = "CREATE TABLE patient ( "+\
        "idp INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "+\
        "nom TEXT NOT NULL, "+\
        "prenom TEXT NOT NULL, "+\
        "naissance TEXT NOT NULL);"

    cur.execute(sql)
    conn.commit()
    cur.close()
    conn.close()
    print("Table patient créée.")

except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

Table patient créée.
```

Entrée [30]: # Crédation d'une liste de médecin

```
faux_et_usage_de_faux= Faussaire()

liste_medecin=[]
hf=['h','f']
liste_medecin.append(("MAFFRE","lucas"))
for i in range(200):
    nom, prenom= faux_et_usage_de_faux.nom_prenom(hf[random.randint(0,1)])
    l= nom, prenom
    liste_medecin.append(l)
```

Entrée [31]: # On inclu dans le dataframe les patients, les id de médecin, et la colonne diag_conf

```
df=df.reset_index(drop= True)

df["nom"]= ""
df["prenom"]= ""
df["idp"]= 0
df["idmed"]= 0
df["diag_conf"]= 1 # On suppose que tout les diagnostiques dans la base sont confirmées

for idx, row in df.iterrows():
    #Seul l'id du médecin figurera dans la table examen
    df.loc[idx,"idmed"]= random.randint(0, len(liste_medecin)-1)

    # future clé primaire de la table patient
    df.loc[idx,"idp"]= idx +1

    # Calcul de la date de naissance. On suppose que l'âge donnée dans la table
    # correspond à l'âge du patient au jour de l'examen. On ajoute à ce nombre de
    # jours un nombre aléatoire et on en "déduit" une date de naissance...
    # Même si seul "idp" appartient à la table d'examen, on est obligé d'inclure
    # les noms prénoms et date de naissance de façon provisoire dans cette table
    ecart= random.randint(10,1000)
    age= int(df.loc[idx,"age"])
    dt= date.today()-timedelta(days= age + ecart)
    df.loc[idx,"naissance"]= pd.to_datetime(dt)

    # Nom et prénom (en fonction du sexe) du patient
    if df.loc[idx,"gender"]== 1:
        nom, prenom= faux_et_usage_de_faux.nom_prenom('h')
        df.loc[idx,"nom"]= nom
        df.loc[idx,"prenom"]= prenom
    else:
        nom, prenom= faux_et_usage_de_faux.nom_prenom('f')
        df.loc[idx,"nom"]= nom
        df.loc[idx,"prenom"]= prenom
```

```
Entrée [32]: # df pour créer la table patient
df_patient= df.drop(['age', 'gender', 'height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol',
'gluc', 'smoke', 'alco', 'active', 'cardio', 'idmed', 'diag_conf'],axis= 1)
```

```
Entrée [33]: # Ne pas ré-exécuter ce code une fois la base créée - Remplissage de la table patient
"""
try:
    # Connexion à la base de donnée
    conn = sqlite3.connect(bdp)
    cur = conn.cursor()

    sql = "INSERT INTO patient(nom,prenom,idp,naissance) VALUES(?, ?, ?, ?)"

    for i, row in df_patient.iterrows():
        l= tuple(row)
        cur.execute(sql, (l[0],l[1],l[2],str(l[3])))
    conn.commit()

    cur.close()
    conn.close()
    print("Table patient rempli.")
except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

Table patient rempli.
```

Table des médecins

```
Entrée [34]: # Ne pas ré-exécuter ce code une fois la table créée - Création de la table medecin
"""
try:
    conn = sqlite3.connect(bdp)
    cur = conn.cursor()

    sql = "CREATE TABLE medecin ( "+\
          "idmed INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "+\
          "nom TEXT NOT NULL, "+\
          "prenom TEXT NOT NULL);"

    cur.execute(sql)
    conn.commit()
    cur.close()
    conn.close()
    print("Table medecin créée.")

except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

Table medecin créée.
```

```
Entrée [35]: # Ne pas ré-exécuter ce code une fois la table rempli - Remplissage de la table médecins
"""
try:
    # Connexion à la base de donnée
    conn = sqlite3.connect(bdp)
    cur = conn.cursor()

    sql = "INSERT INTO medecin(nom, prenom) VALUES(?,?)"

    for nom, prenom in liste_medecin:
        cur.execute(sql, (nom,prenom))
    conn.commit()

    cur.close()
    conn.close()
    print("Table medecin rempli.")
except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

Table medecin rempli.
```

Table examens

```
Entrée [36]: # Ne pas ré-exécuter ce code une fois la base créée - Création de la table examen
"""
try:
    conn = sqlite3.connect(bdp)
    cur = conn.cursor()

    sql = "CREATE TABLE examen ( "+\
          "idexa INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "+\
          "age_INTEGER NOT NULL, "+\
          "gender INTEGER, "+\
          "height INTEGER NOT NULL, "+\
          "weight INTEGER NOT NULL, "+\
          "ap_hi INTEGER NOT NULL, "+\
          "ap_lo INTEGER NOT NULL, "+\
          "cholesterol INTEGER, "+\
          "gluc INTEGER, "+\
          "smoke INTEGER, "+\
          "alco INTEGER, "+\


```

```

    "active INTEGER, "+\
    "idp INTEGER NOT NULL, "+\
    "idmed INTEGER NOT NULL, "+\
    "cardio INTEGER , "+\
    "diag_conf INTEGER , "+\
    "FOREIGN KEY(idp) REFERENCES patient(idp),"+\
    "FOREIGN KEY (idmed) REFERENCES medecin (idmed));" #+\

    cur.execute(sql)
    conn.commit()
    cur.close()
    conn.close()
    print("Table examen créée.")

except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

```

Table examen créée.

Entrée [37]: # df pour créer la table examens
`df_examens = df.drop(['nom', 'prenom', 'naissance'], axis=1)`

Entrée [38]: df_examens

Out[38]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	idp	idmed	diag_conf
0	18393	2	168	62.0	110	80	1	1	0	0	1	0	1	78	1
1	20228	1	156	85.0	140	90	3	1	0	0	1	1	2	132	1
2	18857	1	165	64.0	130	70	3	1	0	0	0	1	3	104	1
3	17623	2	169	82.0	150	100	1	1	0	0	1	1	4	179	1
4	17474	1	156	56.0	100	60	1	1	0	0	0	0	5	107	1
...
68841	19240	2	168	76.0	120	80	1	1	1	0	1	0	68842	118	1
68842	22601	1	158	126.0	140	90	2	2	0	0	1	1	68843	43	1
68843	19066	2	183	105.0	180	90	3	1	0	1	0	1	68844	51	1
68844	22431	1	163	72.0	135	80	1	2	0	0	0	1	68845	36	1
68845	20540	1	170	72.0	120	80	2	1	0	0	1	0	68846	78	1

68846 rows × 15 columns

Entrée [39]: # Ne pas ré-exécuter ce code une fois la base créée - Remplissage de la table examen

```

"""
try:
    # Connexion à la base de donnée
    conn = sqlite3.connect(bdp)
    cur = conn.cursor()

    sql = "INSERT INTO examen (age, gender, height, weight, ap_hi, ap_lo, cholesterol, "+\
        "gluc, smoke, alco, active, cardio, idp, idmed, diag_conf) VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)"

    for i, row in df_examens.iterrows():
        cur.execute(sql, tuple(row))
        conn.commit()

    cur.close()
    conn.close()
    print("Table examen rempli")
except sqlite3.Error as error:
    print("Erreur lors de la connexion à SQLite", error)
"""

```

Table examen rempli

Création table diagnostique

Entrée [9]: # Ne pas ré-exécuter ce code une fois la base créée - Crédation de la table diagnostique

```

"""
try:
    conn = sqlite3.connect(bdp)
    cur = conn.cursor()

    sql = "CREATE TABLE diagnostique ( "+\
        "iddiag INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "+\
        "age INTEGER NOT NULL, "+\
        "gender INTEGER, "+\
        "height INTEGER NOT NULL, "+\
        "weight INTEGER NOT NULL, "+\
        "ap_hi INTEGER NOT NULL, "+\
        "ap_lo INTEGER NOT NULL, "+\
        "cholesterol INTEGER, "+\
        "gluc INTEGER, "+\
        "smoke INTEGER, "+\
        "alco INTEGER, "+\
        "active INTEGER, "+\
        "idp INTEGER NOT NULL, "+\
        "idmed INTEGER NOT NULL, "+\
        "cardio INTEGER , "+\
        "resultat_diag TEXT , "+\
        "FOREIGN KEY(idp) REFERENCES patient(idp),"+\
        "FOREIGN KEY (idmed) REFERENCES medecin (idmed));" #+\

```

```

        cur.execute(sql)
        conn.commit()
        cur.close()
        conn.close()
        print("Table diagnostique créée.")

    except sqlite3.Error as error:
        print("Erreur lors de la connexion à SQLite", error)
"""


```

Table diagnostique créée.

```

Entrée [ ]: # Test enregistrement donnée dans table diagnostique
"""
bdd= './datas/base_E1.db'
message_erreur=""
element= [20628, 1, 185, 90, 150, 90, 1, 1, 0, 1, 1, 1, 1, 1, {"'random-forest_rf160.h5': array([0.21875, 0.78125]), "+\
    "'gaussianNB_gs00015199.h5': array([0.12828491, 0.87171509]), "+\
    "'knn_knn35.h5': array([0.25714286, 0.74285714]), 'bernoulli_bn.h5': array([0.06879583, 0.93120417]), "+\
    "'sgd_sgd.h5': array([0.23838886, 0.76161114]), 'score': '5/5'}",1,1,1]
try:
# Connexion à la base de donnée
    conn = sqlite3.connect(bdd)
    cur = conn.cursor()

    sql = "INSERT INTO diagnostique (age, gender, height, weight, ap_hi, ap_lo, "+\
        "cholesterol, gluc, smoke, alco, active, resultat_diag, idp, idmed, cardio) "+\
        "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
    ligne= tuple(el for el in element)

    cur.execute(sql, ligne)

    #print(f"ligne: {ligne}")
    conn.commit()

    cur.close()
    conn.close()
except sqlite3.Error as error:
    message_erreur= f"Erreur lors de la connexion à SQLite: {error}"
print(message_erreur)
"""


```

Entrainement d'un modèle de machine learning avec les données de la table d'entraînement

On récupère les données

```

Entrée [10]: bdd= './datas/base_E1_train.db'
conn = sqlite3.connect(bdd)

dfbase= pd.read_sql_query("SELECT * FROM entrainement", conn)

conn.close()

# La colonne id ne servant pas, on la supprime
dfbase.drop(["id"], axis= 1, inplace= True)

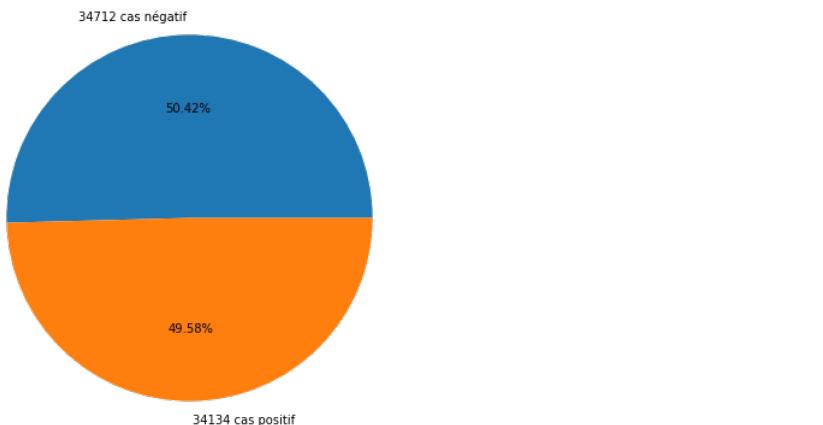
```

```

Entrée [11]: # Création des tableaux des caractéristiques et du label, et découpe en jeu
# d'entraînement et de test
X= dfbase.drop(["cardio"], axis= 1).values
Y= dfbase["cardio"]
# Création du jeux de test et de train
X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size= 0.3, random_state= 51165)

```

```
Entrée [7]: # Répartition du dataset
ze= Y[Y== 0].count() #dfbase["cardio"]== 0].count()
un= Y[Y== 1].count() #dfbase["cardio"]== 1].count()
plt.figure(figsize=(7,7))
plt.pie([ze,un], labels= [f"{{ze}} cas négatif",f"{{un}} cas positif"], autopct= '%1.2f%')
plt.show()
```



```
Entrée [37]: # Contenu des variables de classification.
```

```
for el in ("gender", "cholesterol", "gluc", "smoke", "alco", "active"):
    print(f"Valeur possible de la colonne '{el}': {np.sort(dfbase[el].unique())}")
```

```
Valeur possible de la colonne 'gender': [1 2]
Valeur possible de la colonne 'cholesterol': [1 2 3]
Valeur possible de la colonne 'gluc': [1 2 3]
Valeur possible de la colonne 'smoke': [0 1]
Valeur possible de la colonne 'alco': [0 1]
Valeur possible de la colonne 'active': [0 1]
```

```

Entrée [36]: # Répartition des différents composantes
fig, ax = plt.subplots(nrows= 3, ncols= 2, figsize= (15,8))

homme= X.gender[X.gender== 1].count()
femme= X.gender[X.gender== 2].count()
ax[0][0].pie([homme,femme], labels= [f"{homme} hommes",f"{femme} femmes"], autopct= '%1.2f%')
ax[0][0].set_title("Répartition homme femme")

fumeur= X.smoke[X.smoke== 1].count()
nonfumeur= X.smoke[X.smoke== 0].count()
ax[0][1].pie([fumeur,nonfumeur], labels= [f"{fumeur} fumeurs",f"{nonfumeur} non fumeurs"], autopct= '%1.2f%')
ax[0][1].set_title("Répartition fumeur / non fumeur")

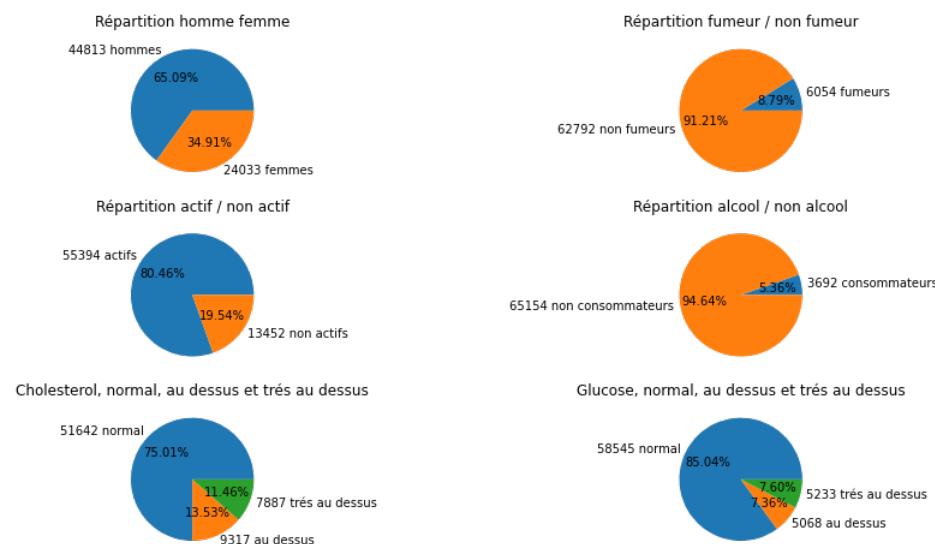
actif= X.active[X.active== 1].count()
nonactif= X.active[X.active== 0].count()
ax[1][0].pie([actif,nonactif], labels= [f"{actif} actifs",f"{nonactif} non actifs"], autopct= '%1.2f%')
ax[1][0].set_title("Répartition actif / non actif")

alcool= X.alco[X.alco== 1].count()
nonalcool= X.alco[X.alco== 0].count()
ax[1][1].pie([alcool,nonalcool], labels= [f"{alcool} consommateurs",f"{nonalcool} non consommateurs"],
    autopct= '%1.2f%')
ax[1][1].set_title("Répartition alcool / non alcool")

chol1= X.cholesterol[X.cholesterol== 1].count()
chol2= X.cholesterol[X.cholesterol== 2].count()
chol3= X.cholesterol[X.cholesterol== 3].count()
ax[2][0].pie([chol1,chol2, chol3], labels= [f"{chol1} normal",f"{chol2} au dessus",
    , f"{chol3} très au dessus"],
    autopct= '%1.2f%')
ax[2][0].set_title("Cholesterol, normal, au dessus et très au dessus")

gluc1= X.gluc[X.gluc== 1].count()
gluc2= X.gluc[X.gluc== 2].count()
gluc3= X.gluc[X.gluc== 3].count()
ax[2][1].pie([gluc1,gluc2, gluc3], labels= [f"{gluc1} normal",f"{gluc2} au dessus",
    , f"{gluc3} très au dessus"],
    autopct= '%1.2f%')
_= ax[2][1].set_title("Glucose, normal, au dessus et très au dessus")

```



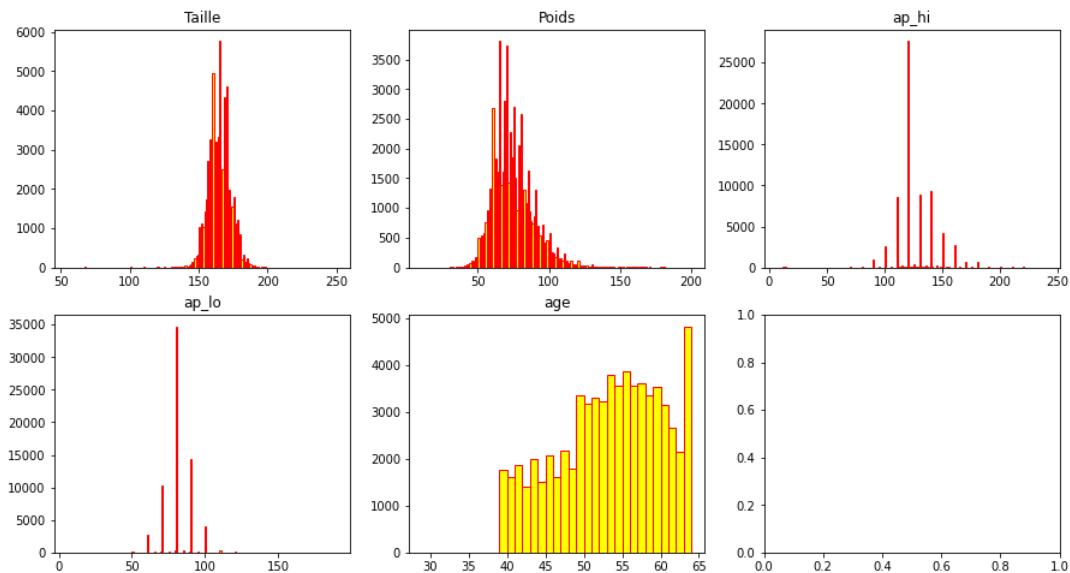
```
Entrée [8]: # Affichage des variables quantitatives
fig, ax = plt.subplots(nrows= 2, ncols= 3, figsize= (15,8))

titre= [["Taille", "Poids","ap_hi"], ["ap_lo", "age","", ] ]
xt = [ [np.array(X.height), np.array(X.weight), np.array(X.ap_hi)], [np.array(X.ap_lo),
(np.array(X.age)/365).astype(int), None] ]

for i in range(0,2):
    for j in range(0,3):
        t= titre[i][j]
        if t == "": break

        x= xt[i][j]
        mini= int(x.min())
        maxi= int(x.max())

        ax[i][j].hist(x, range = (mini, maxi), bins = maxi-mini, color = 'yellow',
            edgecolor = 'red')
        ax[i][j].set_title(t)
```



```
Entrée [10]: # Contenu des variables discrètes
for col in dfbase.select_dtypes("int"):
    print(f"{col}: {np.sort(dfbase[col].unique())}")

age      : [10798 10859 10878 ... 23692 23701 23713]
gender   : [1 2]
height   : [ 55  57  64  65  66  67  68  70  71  72  74  75  76  80  81  91  96  97
  98  99 100 104 105 108 109 110 111 112 117 119 120 122 125 128 130 131
132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149
150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167
168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185
186 187 188 189 190 191 192 193 194 195 196 197 198 207 250]
ap_hi   : [ 7 10 11 12 13 14 15 16 17 20 24 60 70 80 85 90 93 95
  96 97 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114
115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132
133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168
169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 187
190 191 192 193 194 195 196 197 200 202 210 215 220 230 240]
ap_lo   : [ 6  7  8  9 10 15 20 30 40 45 49 50 52 53 54 55 56 57
  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75
  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93
  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111
112 113 114 115 116 117 118 119 120 121 122 125 126 130 135 140 150 160 170 180
182 190]
cholesterol: [1 2 3]
gluc     : [1 2 3]
smoke    : [0 1]
alco     : [0 1]
active   : [0 1]
cardio   : [0 1]
```

```
Entrée [11]: #corrMatrix =
plt.figure(figsize=(15,10))
sn.heatmap(dfbase.corr(), annot=True)
plt.show()
```



Analyse en composantes principales

- Sans normalisation
- Normalisation minmax
- Normalisation Standard
- Normalisation Robuste

```

Entrée [48]: model= PCA(n_components= 2)
X_train_red2= model.fit_transform(X_train)

normalisation= RobustScaler()
X_train_normalise_Robust= normalisation.fit_transform(X_train)
X_train_normalise_Robust_red2= model.fit_transform(X_train_normalise_Robust)

normalisation= MinMaxScaler()
X_train_normalise_MinMax= normalisation.fit_transform(X_train)
X_train_normalise_MinMax_red2= model.fit_transform(X_train_normalise_MinMax)

normalisation= StandardScaler()
X_train_normalise_Standard= normalisation.fit_transform(X_train)
X_train_normalise_Standard_red2= model.fit_transform(X_train_normalise_Standard)

fig, ax = plt.subplots(nrows= 2, ncols= 2, figsize= (12,12))

ax[0][0].scatter(X_train_red2[:,0],X_train_red2[:,1], c=Y_train, s= 3)
ax[0][0].set_title("Pas de normalisation")

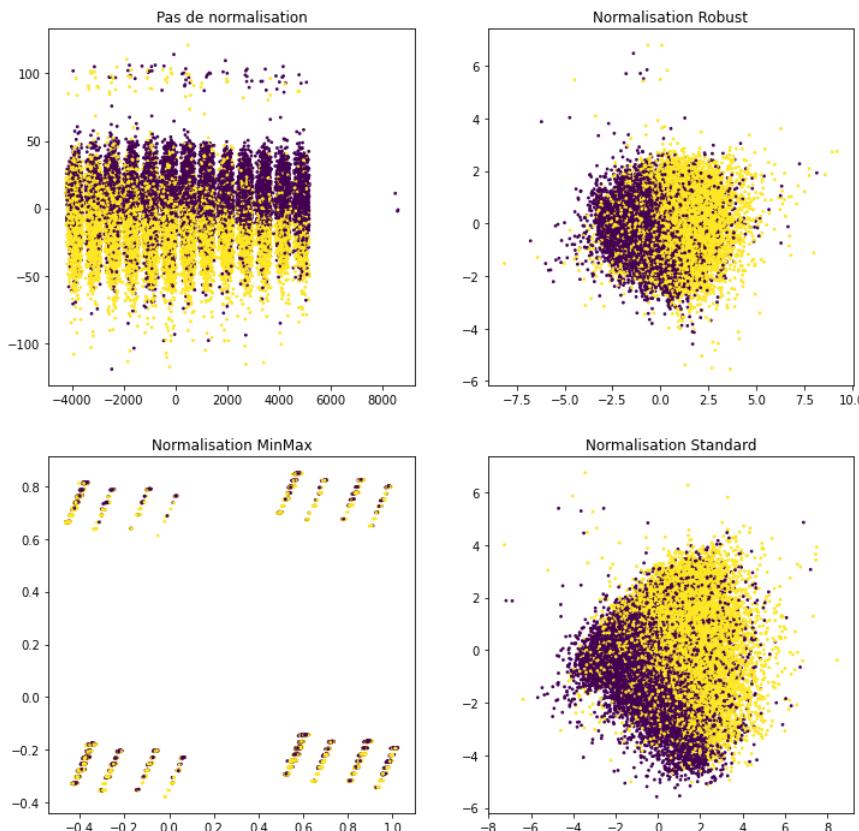
ax[0][1].scatter(X_train_normalise_Robust_red2[:,0],X_train_normalise_Robust_red2[:,1], c=Y_train, s= 3)
ax[0][1].set_title("Normalisation Robust")

ax[1][0].scatter(X_train_normalise_MinMax_red2[:,0],X_train_normalise_MinMax_red2[:,1], c=Y_train, s= 3)
ax[1][0].set_title("Normalisation MinMax")

ax[1][1].scatter(X_train_normalise_Standard_red2[:,0],X_train_normalise_Standard_red2[:,1], c=Y_train, s= 3)
ax[1][1].set_title("Normalisation Standard")

_= plt.show()

```



Test de différents modèles

```

Entrée [49]: for mod in [KNeighborsClassifier(), AdaBoostClassifier(), BaggingClassifier(),
    RandomForestClassifier(), SGDClassifier(), GaussianNB(), BernoulliNB(), SVC()]:
    print(f"Pour le modèle {mod}:")
    for std in [None, StandardScaler(), RobustScaler(), MinMaxScaler()]:
        deb= time.time()
        pre= test_algo(modele= mod, standardisation= std, Xtrain= X_train, Ytrain= Y_train, Xtest= X_test,
                        Ytest= Y_test)
        fin= time.time()
        if std== None: écrit= "None"
        else: écrit= str(std)
        print(f"Standardisation: {écrit}<15>\tprécision: {100*pre:.2f} %\ttemps d'exécution: {fin-deb:.2f} s")

```

```

Pour le modèle KNeighborsClassifier():
Standardisation: None           précision: 68.25 %      temps d'exécution: 4.17 s
Standardisation: StandardScaler()  précision: 69.47 %      temps d'exécution: 15.81 s
Standardisation: RobustScaler()    précision: 69.81 %      temps d'exécution: 12.25 s
Standardisation: MinMaxScaler()    précision: 68.88 %      temps d'exécution: 18.23 s
Pour le modèle AdaBoostClassifier():
Standardisation: None           précision: 73.11 %      temps d'exécution: 4.85 s
Standardisation: StandardScaler()  précision: 73.11 %      temps d'exécution: 4.88 s
Standardisation: RobustScaler()    précision: 73.11 %      temps d'exécution: 4.77 s
Standardisation: MinMaxScaler()    précision: 73.11 %      temps d'exécution: 4.69 s
Pour le modèle BaggingClassifier():
Standardisation: None           précision: 69.03 %      temps d'exécution: 5.85 s
Standardisation: StandardScaler()  précision: 69.42 %      temps d'exécution: 5.96 s
Standardisation: RobustScaler()    précision: 69.34 %      temps d'exécution: 6.21 s
Standardisation: MinMaxScaler()    précision: 68.77 %      temps d'exécution: 6.06 s
Pour le modèle RandomForestClassifier():
Standardisation: None           précision: 71.14 %      temps d'exécution: 24.23 s
Standardisation: StandardScaler()  précision: 71.21 %      temps d'exécution: 23.51 s
Standardisation: RobustScaler()    précision: 71.38 %      temps d'exécution: 23.09 s
Standardisation: MinMaxScaler()    précision: 71.06 %      temps d'exécution: 23.23 s
Pour le modèle SGDClassifier():
Standardisation: None           précision: 53.14 %      temps d'exécution: 7.85 s
Standardisation: StandardScaler()  précision: 72.19 %      temps d'exécution: 0.55 s
Standardisation: RobustScaler()    précision: 72.18 %      temps d'exécution: 0.57 s
Standardisation: MinMaxScaler()    précision: 72.70 %      temps d'exécution: 0.46 s
Pour le modèle GaussianNB():
Standardisation: None           précision: 70.70 %      temps d'exécution: 0.11 s
Standardisation: StandardScaler()  précision: 70.62 %      temps d'exécution: 0.10 s
Standardisation: RobustScaler()    précision: 70.62 %      temps d'exécution: 0.14 s
Standardisation: MinMaxScaler()    précision: 70.62 %      temps d'exécution: 0.09 s

```

Optimisation des différents algorithmes

On utilisera le StandardScaler pour la normalisation des données.

```
Entrée [50]: normalisation= StandardScaler()
X_train_normalise= normalisation.fit_transform(X_train)
X_test_normalise= normalisation.transform(X_test)
```

KNeighborsClassifier

```
Entrée [51]: param_grid = { "kneighborsclassifier_n_neighbors" : [25, 30, 40],
                           "kneighborsclassifier_algorithm" : ["ball_tree"],
                           "kneighborsclassifier_weights" : ['uniform']
                         }

model= make_pipeline(StandardScaler(),KNeighborsClassifier())

gridkn = GridSearchCV(model, param_grid, cv=5, n_jobs= -1)

gridkn
```

```
Out[51]: GridSearchCV(cv=5,
                      estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
                                                ('kneighborsclassifier',
                                                 KNeighborsClassifier())]),
                      n_jobs=-1,
                      param_grid={'kneighborsclassifier_algorithm': ['ball_tree'],
                                  'kneighborsclassifier_n_neighbors': [25, 30, 40],
                                  'kneighborsclassifier_weights': ['uniform']})
```

```
Quelques résultats:
5714 erreurs sur 20654 images soit une précision de 72.33 %
{'kneighborsclassifier_algorithm': 'ball_tree',
 'kneighborsclassifier_n_neighbors': 32,
 'kneighborsclassifier_weights': 'uniform'}

5675 erreurs sur 20654 images soit une précision de 72.52 %
{'kneighborsclassifier_algorithm': 'ball_tree',
 'kneighborsclassifier_n_neighbors': 35,
 'kneighborsclassifier_weights': 'uniform'}

5689 erreurs sur 20654 images soit une précision de 72.46 %
{'kneighborsclassifier_algorithm': 'ball_tree',
 'kneighborsclassifier_n_neighbors': 40,
 'kneighborsclassifier_weights': 'uniform'}

5654 erreurs sur 20654 images soit une précision de 72.63 %
{'kneighborsclassifier_algorithm': 'ball_tree',
 'kneighborsclassifier_n_neighbors': 70,
 'kneighborsclassifier_weights': 'uniform'}

5656 erreurs sur 20654 images soit une précision de 72.62 %
{'kneighborsclassifier_algorithm': 'ball_tree',
 'kneighborsclassifier_n_neighbors': 71,
 'kneighborsclassifier_weights': 'uniform'}
```

```
Entrée [56]: model_knn= make_pipeline(StandardScaler(),
                                         KNeighborsClassifier(n_neighbors= 35, algorithm= "ball_tree", weights= "uniform"))

model_knn.fit(X_train, Y_train)
prediction = model_knn.predict(X_test)

err, pre, d= erreur_prediction(valeurs_predites= prediction, valeurs_reelle= np.array(Y_test))
print(f'{err} erreurs sur {len(X_test)} données soit une précision de {pre:.2f} %')
```

```
#dump(model_knn, './modeles/knn_knn35.h5')
5675 erreurs sur 20654 données soit une précision de 72.52 %
```

```
Out[56]: ['./modeles/knn_knn35.h5']
```

```
Entrée [57]: confusion_matrix(Y_test, model_knn.predict(X_test))
```

```
Out[57]: array([[8062, 2321],
 [3354, 6917]])
```

RandomForest

```
Entrée [58]: param_grid = { "randomforestclassifier__n_estimators" : [200,500,1000],
 "randomforestclassifier__criterion" : ["gini"],
 "randomforestclassifier__max_features": ['auto'], # auto= sqrt
 }
```

```
model= make_pipeline(StandardScaler(),RandomForestClassifier())
```

```
gridrf = GridSearchCV(model, param_grid, cv=5, n_jobs= -1)
```

```
gridrf
```

```
Out[58]: GridSearchCV(cv=5,
 estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
 ('randomforestclassifier',
 RandomForestClassifier())]),
 n_jobs=-1,
 param_grid={'randomforestclassifier__criterion': ['gini'],
 'randomforestclassifier__max_features': ['auto'],
 'randomforestclassifier__n_estimators': [200, 500,
 1000]})
```

```
GridSearchCV(cv=5,
 estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
 ('randomforestclassifier',
 RandomForestClassifier())]),
 n_jobs=-1,
 param_grid={'randomforestclassifier__criterion': ['gini',
 'entropy'],
 'randomforestclassifier__max_features': ['sqrt',
 'log2'],
 'randomforestclassifier__n_estimators': [100, 140, 145,
 150, 155, 160,
 180]})
```

```
Meilleurs paramètres:
```

```
{'randomforestclassifier__criterion': 'gini',
'randomforestclassifier__max_features': 'sqrt',
'randomforestclassifier__n_estimators': 160}
```

```
5954 erreurs sur 20654 images soit une précision de 71.17 %
```

```
GridSearchCV(cv=5,
 estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
 ('randomforestclassifier',
 RandomForestClassifier())]),
 n_jobs=-1,
 param_grid={'randomforestclassifier__criterion': ['gini'],
 'randomforestclassifier__max_features': ['auto'],
 'randomforestclassifier__n_estimators': [200, 500,
 1000]})
```

```
{'randomforestclassifier__criterion': 'gini',
'randomforestclassifier__max_features': 'auto',
'randomforestclassifier__n_estimators': 500}
```

```
5924 erreurs sur 20654 images soit une précision de 71.32 %
```

```
Entrée [59]: model_rf= make_pipeline(StandardScaler(),RandomForestClassifier(n_estimators= 160, criterion= "gini"))
```

```
model_rf.fit(X_train, Y_train)
```

```
prediction = model_rf.predict(X_test)
```

```
err, pre, d= erreur_prediction(valeurs_predites= prediction, valeurs_reelle= np.array(Y_test))
print(f'{err} erreurs sur {len(X_test)} données soit une précision de {pre:.2f} %")
```

```
dump(model_rf, './modeles/random-forest_rf160.h5')
```

```
5910 erreurs sur 20654 données soit une précision de 71.39 %
```

```
Out[59]: ['./modeles/random-forest_rf160.h5']
```

```
Entrée [60]: confusion_matrix(Y_test, model_rf.predict(X_test))
```

```
Out[60]: array([[7613, 2770],
 [3140, 7131]])
```

SGDClassifier():

```
Standardisation: StandardScaler() précision: 72.14 %
```

```
Entrée [62]: param_grid = { "sgdclassifier_loss" : ['hinge'],
                           "sgdclassifier_penalty" : ['elasticnet'],
                           "sgdclassifier_learning_rate" : ['adaptive'],
                           "sgdclassifier_eta0": [1,5,10,15]
                         }
```

```
model= make_pipeline(StandardScaler(),SGDClassifier())
gridsgd = GridSearchCV(model, param_grid, cv=5, n_jobs= -1)
gridsgd
```

```
Out[62]: GridSearchCV(cv=5,
                      estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
                                                ('sgdclassifier', SGDClassifier())]),
                      n_jobs=-1,
                      param_grid={'sgdclassifier_eta0': [1, 5, 10, 15],
                                  'sgdclassifier_learning_rate': ['adaptive'],
                                  'sgdclassifier_loss': ['hinge'],
                                  'sgdclassifier_penalty' : ['elasticnet']})
```

```
Entrée [63]: gridsgd.fit(X_train, Y_train)
```

```
Out[63]: GridSearchCV(cv=5,
                      estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
                                                ('sgdclassifier', SGDClassifier())]),
                      n_jobs=-1,
                      param_grid={'sgdclassifier_eta0': [1, 5, 10, 15],
                                  'sgdclassifier_learning_rate': ['adaptive'],
                                  'sgdclassifier_loss': ['hinge'],
                                  'sgdclassifier_penalty' : ['elasticnet']})
```

```
Entrée [64]: gridsgd.best_estimator_
```

```
Out[64]: Pipeline(steps=[('standardscaler', StandardScaler()),
                          ('sgdclassifier',
                           SGDClassifier(eta0=5, learning_rate='adaptive',
                                         penalty='elasticnet'))])
```

```
Entrée [65]: gridsgd.best_params_
```

```
Out[65]: {'sgdclassifier_eta0': 5,
           'sgdclassifier_learning_rate': 'adaptive',
           'sgdclassifier_loss': 'hinge',
           'sgdclassifier_penalty': 'elasticnet'}
```

```
Entrée [66]: gridsgd.best_score_
```

```
Out[66]: 0.7254523793485176

GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
                                       ('sgdclassifier', SGDClassifier())]),
             n_jobs=-1,
             param_grid={'sgdclassifier_loss': ['hinge', 'log',
                                                 'modified_huber',
                                                 'squared_hinge',
                                                 'perceptron'],
                         'sgdclassifier_penalty': ['l2', 'l1', 'elasticnet']})
{'sgdclassifier_loss': 'hinge', 'sgdclassifier_penalty': 'elasticnet'}
0.7231077399410265

GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
                                       ('sgdclassifier', SGDClassifier())]),
             n_jobs=-1,
             param_grid={'sgdclassifier_eta0': [1],
                         'sgdclassifier_learning_rate': ['optimal', 'constant',
                                                        'invscaling',
                                                        'adaptive'],
                         'sgdclassifier_loss': ['hinge'],
                         'sgdclassifier_penalty': ['elasticnet']})
{'sgdclassifier_eta0': 1,
 'sgdclassifier_learning_rate': 'adaptive',
 'sgdclassifier_loss': 'hinge',
 'sgdclassifier_penalty': 'elasticnet'}
0.7253486169240394
```

```
Entrée [17]: model_sgd= make_pipeline(StandardScaler(),
                                         SGDClassifier(loss= 'hinge', penalty= 'elasticnet', learning_rate= 'adaptive', eta0= 5))
model_sgd.fit(X_train, Y_train)
prediction = model_sgd.predict(X_test)

err, pre, d= erreur_prediction(valeurs_predites= prediction, valeurs_reelle= np.array(Y_test))
print(f'{err} erreurs sur {len(X_test)} données soit une précision de {pre:.2f} %')

#dump(model_sgd, './modeles/sgd_sgd.h5')
```

5675 erreurs sur 20654 données soit une précision de 72.52 %

```
Entrée [18]: confusion_matrix(Y_test, model_sgd.predict(X_test))
```

```
Out[18]: array([[8493, 1890],  
                 [3785, 6486]])
```

```
Entrée [22]: # Le paramètre "hinge" ne permet l'utilisation de la méthode predict_proba qui peut être utilisé dans voting
```

```
model_sgd= make_pipeline(StandardScaler(),  
                         SGDClassifier(loss= 'log', penalty= 'elasticnet', learning_rate= 'adaptive', eta0= 5))  
  
model_sgd.fit(X_train, Y_train)  
  
prediction = model_sgd.predict(X_test)  
  
err, pre, d= erreur_prediction(valeurs_predites= prediction, valeurs_reelle= np.array(Y_test))  
print(f" {err} erreurs sur {len(X_test)} données soit une précision de {pre:.2f} %")  
  
dump(model_sgd, './modeles/sgd_sgd.h5')
```

```
5631 erreurs sur 20654 données soit une précision de 72.74 %
```

```
Out[22]: ['./modeles/sgd_sgd.h5']
```

GaussianNB

```
GaussianNB(): Standardisation: None précision: 70.70 % temps d'exécution: 0.04 s Standardisation: StandardScaler() précision: 70.62 % temps d'exécution: 0.05 s  
Standardisation: RobustScaler() précision: 70.62 % temps d'exécution: 0.05 s Standardisation: MinMaxScaler() précision: 70.62 % temps d'exécution: 0.03 s
```

```
Entrée [25]: param_grid = { "gaussiannb__var_smoothing" : np.logspace(0,-9, num=100),}  
  
model= make_pipeline(StandardScaler(),GaussianNB())  
  
gridgs = GridSearchCV(model, param_grid, cv=5) #, n_jobs= -1  
  
gridgs
```

```
Out[25]: GridSearchCV(cv=5,  
                      estimator=Pipeline(steps=[('standardscaler', StandardScaler()),  
                                              ('gaussiannb', GaussianNB())]),  
                      param_grid={'gaussiannb__var_smoothing': array([1.00000000e+00, 8.11130831e-01, 6.57933225e-01, 5.33669923e-01,  
1,  
4.32876128e-01, 3.51119173e-01, 2.84803587e-01, 2.31012970e-01,  
1.87381742e-01, 1.51991108e-01, 1.23284674e-01, 1.00000000e-01,  
8.11130831e-0...  
1.23284674e-07, 1.00000000e-07, 8.11130831e-08, 6.57933225e-08,  
5.33669923e-08, 4.32876128e-08, 3.51119173e-08, 2.84803587e-08,  
2.31012970e-08, 1.87381742e-08, 1.51991108e-08, 1.23284674e-08,  
1.00000000e-08, 8.11130831e-09, 6.57933225e-09, 5.33669923e-09,  
4.32876128e-09, 3.51119173e-09, 2.84803587e-09, 2.31012970e-09,  
1.87381742e-09, 1.51991108e-09, 1.23284674e-09, 1.00000000e-09]))}
```

```
Entrée [26]: gridgs.fit(X_train, Y_train)
```

```
Out[26]: GridSearchCV(cv=5,  
                      estimator=Pipeline(steps=[('standardscaler', StandardScaler()),  
                                              ('gaussiannb', GaussianNB())]),  
                      param_grid={'gaussiannb__var_smoothing': array([1.00000000e+00, 8.11130831e-01, 6.57933225e-01, 5.33669923e-01,  
1,  
4.32876128e-01, 3.51119173e-01, 2.84803587e-01, 2.31012970e-01,  
1.87381742e-01, 1.51991108e-01, 1.23284674e-01, 1.00000000e-01,  
8.11130831e-0...  
1.23284674e-07, 1.00000000e-07, 8.11130831e-08, 6.57933225e-08,  
5.33669923e-08, 4.32876128e-08, 3.51119173e-08, 2.84803587e-08,  
2.31012970e-08, 1.87381742e-08, 1.51991108e-08, 1.23284674e-08,  
1.00000000e-08, 8.11130831e-09, 6.57933225e-09, 5.33669923e-09,  
4.32876128e-09, 3.51119173e-09, 2.84803587e-09, 2.31012970e-09,  
1.87381742e-09, 1.51991108e-09, 1.23284674e-09, 1.00000000e-09]))}
```

```
gridgs.best_params_  
{'gaussiannb__var_smoothing': 0.0015199110829529332}
```

```
Entrée [31]: model_gs= make_pipeline(StandardScaler(), GaussianNB(var_smoothing= 0.0015199110829529332) )
```

```
model_gs.fit(X_train, Y_train)  
  
prediction = model_gs.predict(X_test)  
  
err, pre, d= erreur_prediction(valeurs_predites= prediction, valeurs_reelle= np.array(Y_test))  
print(f" {err} erreurs sur {len(X_test)} données soit une précision de {pre:.2f} %")  
  
dump(model_gs, './modeles/gaussianNB_gs00015199.h5')
```

```
6069 erreurs sur 20654 données soit une précision de 70.62 %
```

```
Out[31]: ['./modeles/gaussianNB_gs00015199.h5']
```

```
Entrée [42]: confusion_matrix(Y_test, model_gs.predict(X_test))
```

```
Out[42]: array([[8469, 1914],  
                 [4155, 6116]])
```

Bernoulli

```
Pour le modèle BernoulliNB(): Standardisation: None précision: 51.89 % temps d'exécution: 0.03 s Standardisation: StandardScaler() précision: 71.24 % temps
```

```
d'exécution: 0.07 s Standardisation: RobustScaler()

Entrée [8]: model_bn= make_pipeline(StandardScaler(), BernoulliNB())

model_bn.fit(X_train, Y_train)

prediction = model_bn.predict(X_test)

err, pre, d= erreur_prediction(valeurs_predites= prediction, valeurs_reelle= np.array(Y_test))
print(f'{err} erreurs sur {len(X_test)} données soit une précision de {pre:.2f} %')

dump(model_bn, './modeles/bernoulli_bn.h5')

5940 erreurs sur 20654 données soit une précision de 71.24 %

Out[8]: ['./modeles/bernoulli_bn.h5']

Entrée [10]: confusion_matrix(Y_test, model_bn.predict(X_test))

Out[10]: array([[7987, 2396],
       [3544, 6727]])
```

Test voting

```
Entrée [24]: rf= load("modeles/random-forest_rf160.h5")
knn= load("modeles/knn_knn35.h5")
sgd= load("modeles/sgd_sgd.h5")
gs= load("modeles/gaussianNB_gs00015199.h5")
bn= load("./modeles/bernoulli_bn.h5")
```

```
Entrée [25]: eclf1 = VotingClassifier(estimators=[('rf', rf), ('knn', knn), ('sgd', sgd), ('gs', gs), ('bn', bn)], voting='hard')
eclf1 = eclf1.fit(X_train, Y_train)
```

```
Entrée [26]: print(f"100 * eclf1.score(X_test, Y_test):.2f")
```

72.78 %

```
Entrée [27]: eclf2 = VotingClassifier(estimators=[('rf', rf), ('knn', knn), ('sgd', sgd), ('gs', gs), ('bn', bn)], voting='soft')
eclf2 = eclf2.fit(X_train, Y_train)
```

```
Entrée [29]: print(f"100*eclf2.score(X test, Y test):.{2f} %")
```

72 78 %

Test stacking

```
Entrée [30]: eclf3 = StackingClassifier(estimators=[('rf', rf), ('knn', knn), ('sgd', sgd), ('gs', gs), ('bn', bn)],  
                                         final_estimator= make_pipeline(StandardScaler(), RandomForestClassifier()))  
eclf3 = eclf3.fit(X_train, Y_train)  
print(f"\{100*eclf3.score(X_test, Y_test):.2f} %")
```

71.79 %

Test boosting avec adaboost

```
Entrée [37]: clfab = make_pipeline(StandardScaler(), AdaBoostClassifier(n_estimators=100, random_state=0))

        clfab.fit(X_train, Y_train)
        print(f"{100*clfab.score(X_test, Y_test):.2f} %")

73.01 %
```

Optimisation AdaBoost

Certains algorithmes ne sont pas compatible avec Adaboost, comme le knn. D'autres doivent être adapté comme le sgd en mettant le paramètre loss à la valeur log.

```

Entrée [36]: gridab.fit(X_train, Y_train)
/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:
2 fits failed out of a total of 80.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
-----
2 fits failed with the following error:
Traceback (most recent call last):
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/pipeline.py", line 394, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/ensemble/_weight_boosting.py", line 486, in fit
    return super().fit(X, y, sample_weight)
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/ensemble/_weight_boosting.py", line 145, in fit
    sample_weight, estimator_weight, estimator_error = self._boost(
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/ensemble/_weight_boosting.py", line 548, in _boost
    return self._boost_real(iboost, X, y, sample_weight, random_state)
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/ensemble/_weight_boosting.py", line 557, in _boost_real
    estimator.fit(X, y, sample_weight=sample_weight)
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/naive_bayes.py", line 245, in fit
    return self._partial_fit()
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/naive_bayes.py", line 468, in _partial_fit
    new_theta, new_sigma = self._update_mean_variance(
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/naive_bayes.py", line 299, in _update_mean_variance
    new_mu = np.average(X, axis=0, weights=sample_weight)
  File "<__array_function__ internals>", line 180, in average
  File "/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/numpy/lib/function_base.py", line 524, in average
    raise ZeroDivisionError()
ZeroDivisionError: Weights sum to zero, can't be normalized

    warnings.warn(some_fits_failed_message, FitFailedWarning)
/home/jpphi/anaconda3/envs/prjfe/lib/python3.8/site-packages/sklearn/model_selection/_search.py:969: UserWarning: One or more of the test scores are non-finite: [0.71814832 0.71814832 0.71814832 0.71814832 0.50298812 0.50176413
      nan      nan 0.72343957 0.72308681 0.72300379 0.72279628
      0.67419892 0.67780965 0.67778876 0.67689668]
    warnings.warn()

Out[36]: GridSearchCV(cv=5,
estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
                           ('adaboostclassifier',
                            AdaBoostClassifier())]),
param_grid={'adaboostclassifier__base_estimator': [BernoulliNB(),
                                                    GaussianNB(),
                                                    SGDClassifier(loss='log'),
                                                    DecisionTreeClassifier()],
            'adaboostclassifier__n_estimators': [50, 100, 150,
                                                200]})

gridab.best_estimator_
Pipeline(steps=[('standardscaler', StandardScaler()),
               ('adaboostclassifier',
                AdaBoostClassifier(base_estimator=SGDClassifier(loss='log')))])

gridab.best_params_
{'adaboostclassifier__base_estimator': SGDClassifier(loss='log'),
 'adaboostclassifier__n_estimators': 50}

gridab.best_score_
0.7232528282192805

Entrée [13]: model_ab= make_pipeline(StandardScaler(), AdaBoostClassifier(base_estimator= SGDClassifier(loss= 'log'),
n_estimators= 50))

model_ab.fit(X_train, Y_train)

prediction = model_ab.predict(X_test)

err, pre, d= erreur_prediction(valeurs_predites= prediction, valeurs_reelle= np.array(Y_test))
print(f'{err} erreurs sur {len(X_test)} données soit une précision de {pre:.2f} %')

dump(model_ab, './modeles/adaboost_absgd.h5')

5708 erreurs sur 20654 données soit une précision de 72.36 %

Out[13]: ['./modeles/adaboost_absgd.h5']

Entrée [14]: confusion_matrix(Y_test, model_ab.predict(X_test))

Out[14]: array([[8051, 2332],
               [3376, 6895]])

```

Accueil

Prévention des maladies cardio-vasculaires

Ce programme permet :

- Utiliser des algorithmes pré-entraînés chargé d'évaluer le risque pour un patient d'avoir un problème cardio-vasculaire.
- Enregistrer les nouvelles données entrées qui pourront ainsi enrichir la base de données.
- De consulter les éléments enregistrés dans la base de données.
- D'accéder aux différentes statistiques réalisées à partir des données enregistrées.

Utilisateur : Nom Mot de passe : Connexion

Erreur de connexion

Reconnexion ?

L'utilisateur n'existe pas, ou le mot de passe n'est pas correct..

Même joueur, joue encore !

Login :

Utilisateur : Nom Mot de passe Connexion

Accès autorisé

Bienvenue dans le programme d'aide au dépistage de maladies cardio-vasculaires.

Souhaitez-vous travailler sur les données ou effectuer une prédiction ?

[Effectuer une prédiction sur les données](#).

Ouverture des tables examen, médecin et patient depuis /home/jpbh/Documents/brief/ProjetFinDEtude/E1/datas/base_E1.db

Déconnexion

Vous êtes déconnecté

Utilisateur : Nom Mot de passe Connexion

Prédictions

Prédiction du risque cardio-vasculaire.

Fiche patient.

Médecin prescripteur: Dr-faus kather • Homme • Activité physique
Glucose: 1 • Cholesterol: 1 • Taille (en cm): 185 • Poids (en kg): 90 • PA systolique: 150 • PA diastolique: 90
Femme • Sans activité physique • Fumeur • Non fumeur • Consommation d'alcool • Pas de consommation d'alcool

Nom: gudule Prénom: gérard Date de naissance: 05-11-1965 Date de la consultation: 03-05-2022 Lancer l'analyse

Veuillez remplir tous les champs du formulaire avant de lancer une analyse.

Données

Base de donnée - données brutes

Cette base comporte plusieurs tables.

- La table examen: Comporte l'ensemble des données cliniques ainsi que l'identifiant du patient ainsi que l'identifiant du médecin.
- La table médecin: Le nom des médecins prescripteurs.
- La table patient: Le nom des patients ayant participé à l'étude.
- La table diagnostic: Les différents diagnostics réalisés

Examens Médecins Patients Diagnostics

Index	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	ldp	idmed	cardio	diag_conf
1	18393	2	168	62	110	80	1	1	0	0	1	1	187	0	1
2	20228	1	156	85	140	90	3	1	0	0	1	2	63	1	1
3	18857	1	165	64	130	70	3	1	0	0	0	3	46	1	1
4	17623	2	169	82	150	100	1	1	0	0	1	4	96	1	1
5	17474	1	156	56	100	60	1	1	0	0	0	5	94	0	1
6	21934	1	151	67	120	80	2	2	0	0	0	6	127	0	1
7	22113	1	157	93	130	80	3	1	0	0	1	7	19	0	1
8	22584	2	178	95	130	90	3	3	0	0	1	8	36	1	1

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
Application de création de base de donnée avec table utilisateur et
table type d'utilisateur.

Fichier lançant l'application. Doit se situer à la racine du site.

Ressources:
    https://ichi.pro/fr/comment-configurer-l-authentification-utilisateur-pour-les-
applications-
dash-a-l-aide-de-python-et-flask-118945949211473
    https://dash.plotly.com/
    https://dash-bootstrap-components.opensource.faculty.ai/docs/components/
"""

@auteur: jpphi

"""

#from turtle import width
from dash import dcc, html

import dash
from dash.dependencies import Input, Output, State

#from sqlalchemy import Table, create_engine, Table
#from sqlalchemy import *
#from flask_sqlalchemy import SQLAlchemy
#import sqlalchemy

from sqlalchemy.sql import select
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash #, check_password_hash
import sqlite3
import warnings
import os
from flask_login import login_user, logout_user, current_user, LoginManager, UserMixin
import configparser

from apps import home, createdb, createuser

#warnings.filterwarnings("ignore")

db = SQLAlchemy()
#config = configparser.ConfigParser()

data_base=""

app = dash.Dash(__name__)
server = app.server
app.config.suppress_callback_exceptions = True

app.layout = html.Div([
    dcc.Location(id='url', refresh= True),
    html.Div(id='page-content')
])

@app.callback(
    Output('page-content', 'children'),
    [Input('url', 'pathname')])
def display_page(pathname):
    if pathname == "/createdb":
        return createdb.layout
    elif pathname == '/createuser':
        return createuser.layout
    else:

```

```

    return home.layout

@app.callback( [Output('container-button-basic', "children")],
    [Input('submit-val', 'n_clicks')],
    [State('dbname', 'value'), State('creatorname', 'value'), State('password', 'value'),
State('email', 'value')])
def create_database(n_clicks, dbase, creatorname, password, email):
    global data_base

    if dbase is not None and creatorname is not None and password is not None and email
is not None:

        password_encrypted = generate_password_hash(password, method= 'sha256')

        if os.path.isfile(dbase)== False:
            conn = sqlite3.connect(dbase)
            conn.execute("CREATE TABLE type_utilisateur ("+\\
                "id_type_utilisateur INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ,"+\\
                "type TEXT NOT NULL )")

            conn.execute("CREATE TABLE utilisateur ("+\\
                "id_utilisateur INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ,"+\\
                "pseudo TEXT NOT NULL , " + \
                "mdp TEXT NOT NULL,"+\\
                "email TEXT NOT NULL,"+\\
                "id_fk_type_utilisateur INTEGER NO NULL,"+\\
                "FOREIGN KEY(id_fk_type_utilisateur) REFERENCES
type_utilisateur(id_type_utilisateur) )")

            conn.execute("INSERT INTO type_utilisateur (type) "+\
                "VALUES ('Propriétaire'), ('Administrateur'), ('Utilisateur'),\n('Visiteur')")

            conn.execute(f"INSERT INTO utilisateur (pseudo, mdp, id_fk_type_utilisateur,\nemail) "+\
                f"VALUES ('{creatorname}', '{password_encrypted}', '{1}', '{email}')")

            # Mise à jour de la base de données
            conn.commit()

            conn.close()
            data_base= dbase

        return [html.Div([html.H2(f"Création de la base {dbase} réussi !
{creatorname} en est le propriétaire "+\
            "et a les privilèges d'administrateur de cette base."),\n            dcc.Link("Connexion à la base, et création d'utilisateurs.",\n            href="/createuser")])
    else:
        return [html.Div([html.H2(f"La base {dbase} existe déjà !"),\n            dcc.Link("Connexion à la base, et création d'utilisateurs.",\n            href="/createuser")])
    else:
        return [html.Div([html.H2("Pour créer la base de données vous devez renseigner\ntout les champs."),\n            dcc.Link("Cliquez ici pour revenir à la page d'accueil", href="/-\nhome")])]

@app.callback( [Output('container-button-basic3', "children")],
    [Input('submit-val', 'n_clicks')],
    [State('username', 'value'), State('password', 'value'), State('email', 'value'),
State('type_user', 'value')])
def insert_users(n_clicks, ps, pwd, em, typus):
    global data_base

    #data_base= "test"

    if ps is not None and pwd is not None and em is not None:

```

```

if data_base!="":
    try:
        # Connection à la base
        conn= sqlite3.connect(data_base)
        cur= conn.cursor()

        # récupération de la liste des pseudo depuis la base de données
        cur.execute("SELECT pseudo FROM utilisateur")
        liste_pseudos_bdd = cur.fetchall()
        liste_pseudos= [el[0] for el in liste_pseudos_bdd]

        # garantir l'unicité du pseudo
        if ps not in liste_pseudos:
            print(liste_pseudos)
            password_encrypted = generate_password_hash(pwd, method='sha256')

            cur.execute(f"INSERT INTO utilisateur (pseudo, mdp,
id_fk_type_utilisateur, email) "+\
                       f"VALUES ('{ps}', '{password_encrypted}', '{typus}', '{em}')")

            # Mise à jour de la base de données
            conn.commit()

            conn.close()

            return [html.Div([html.H2(f"Utilisateur {ps} créé."),
dcc.Link("Cliquez ici pour revenir à la page d'accueil", href='/-
home')])]

        else: # Le pseudo existe déjà
            conn.close()
            return [html.Div([html.H2(f"L'utilisateur {ps} existe déjà !"),
dcc.Link("Cliquez ici pour revenir à la page d'accueil", href='/-
home')])]

    except:
        return [html.Div([html.H2(f"Erreur d'ouverture de la base {data_base}."),
dcc.Link("Cliquez ici pour revenir à la page d'accueil", href='/-
home')])]

        #conn.close()
    else:
        return [html.Div([html.H2(f"Aucune base n'est ouverte."),
dcc.Link("Cliquez ici pour créer une base.", href="/createdb")])]

else:
    return [html.Div([html.H2(f"Tout les champs doivent être rempli."),
dcc.Link("Cliquez ici pour revenir à la page d'accueil", href="/home")])]

if __name__ == '__main__':
    app.run_server(debug=True, port=8080)
    #serve(app, host="0.0.0.0", port=8080)

```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
    Application de création de base de donnée avec table utilisateur et
    table type d'utilisateur.

Fichier de la page home. Doit se situer dans le sous répertoire apps.

Ressources:
    https://ichi.pro/fr/comment-configurer-l-authentification-utilisateur-pour-les-
applications-
dash-a-l-aide-de-python-et-flask-118945949211473
    https://dash.plotly.com/
    https://dash-bootstrap-components.opensource.faculty.ai/docs/components/

@auteur: jpphi
"""

from dash import html
import dash_bootstrap_components as dbc

layout = html.Div([
    dbc.Container(children= [
        html.H1(children= ["Bienvenue dans l'utilitaire de création de base de données"], className="text-center"),

        html.P(children= ["Cette application permet de créer la base de données en local.  
Celle-ci "+\
            "pourra être exportée par la suite sur un serveur distant. Outre la  
création de la base de données "+\
            "cet utilitaire crée les tables et un super utilisateur permettant  
d'administrer la base. "+\
            "Enfin, une page permet de créer d'autres utilisateurs avec différents  
privileges."], className="mb-4"),

        dbc.Button("bdd", href="/createdb"),
        html.Br(),
        dbc.Button("Création d'utilisateurs", href="/-createuser"),

    ], className="mb-5")
])
```

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
Application de création de base de donnée avec table utilisateur et
table type d'utilisateur.

Fichier de création de la base. Doit se situer dans le sous répertoire apps.

Ressources:
    https://ichi.pro/fr/comment-configurer-l-authentification-utilisateur-pour-les-
applications-
dash-a-l-aide-de-python-et-flask-118945949211473
    https://dash.plotly.com/
    https://dash-bootstrap-components.opensource.faculty.ai/docs/components/
"""

@auteur: jpphi

from dash import dcc, html

import dash_bootstrap_components as dbc

# -----
# - Layout -
# -----
# - Layout - 

layout = html.Div(children= [
    html.Div(children= [
        html.H1("Création de la base de donnée, des tables et du login du créateur de la
base."),
        dcc.Location(id='create_db', refresh=True),
        dcc.Input(id= "dbname",
            type= "text",
            placeholder= "Nom de la base de données",
            maxLength= 30),
        dcc.Input(id= "creatorname",
            type= "text",
            placeholder= "Pseudo du créateur de la base",
            maxLength= 30),
        dcc.Input(id= "password",
            type= "password",
            placeholder= "mot de passe",
            maxLength= 50),
        dcc.Input(id= "email",
            type= "text",
            placeholder= "e-mail",
            maxLength= 50),
        html.Button('Crée !', id='submit-val', n_clicks=0),
        html.Div(id='container-button-basic')
    ])
])

"""

layout = dbc.Container([
    html.Div(id='container-button-basic2', children= [
        html.H1("Création de la base de donnée, des tables et du login du créateur de la
base."),
        dcc.Location(id='create_db', refresh=True),
        dcc.Input(id= "dbname",
            type= "text",
            placeholder= "Nom de la base de données",
            maxLength= 30),
        dcc.Input(id= "creatorname",
            type= "text",
            placeholder= "Pseudo du créateur de la base",
            maxLength= 30)
])
]
)

```

```
maxLength= 30),
dcc.Input(id= "password",
type= "password",
placeholder= "mot de passe",
maxLength= 50),
dcc.Input(id= "email",
type= "text",
placeholder= "e-mail",
maxLength= 50),
html.Button('Crée !', id='submit-val', n_clicks=0),
html.Div(id='container-button-basic', children= [html.P(children="Ici et
maintenant")])
])
])
"""

```

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Créé en mars 2022

Projet de fin d'étude Simplon
Application de création de base de donnée avec table utilisateur et
table type d'utilisateur.

Fichier de création d'utilisateur. Doit se situer dans le sous répertoire apps.

Ressources:
    https://ichi.pro/fr/comment-configurer-l-authentification-utilisateur-pour-les-
applications-
dash-a-l-aide-de-python-et-flask-118945949211473
    https://dash.plotly.com/
    https://dash-bootstrap-components.opensource.faculty.ai/docs/components/
"""

@auteur: jpphi
"""

from dash import dcc, html
import dash_bootstrap_components as dbc

# L'utilisateur "Propriétaire" est créé au moment de la création de la table. Il n'y a
# qu'un seul propriétaire, il ne peut donc pas être créé ici !
tab= ["Administrateur", "Utilisateur", "Visiteur"]

#-----
# - Layout -

layout = html.Div(children= [
    html.Div(children= [ html.H1("La base est créée ? Ajout d'utilisateur."),
        dcc.Location(id='create_user', refresh= True),
        dbc.Container([
            dcc.Input(id="username", type="text", placeholder="user name", maxLength =15,
                style= {'display': 'inline-block'}),
            dcc.Input(id="password", type="password", placeholder="password",
                style= {'display': 'inline-block'}),
            dcc.Input(id="email", type="email", placeholder="email", maxLength = 50,
                style= {'display': 'inline-block'}),
            dcc.Dropdown(id='type_user', options=[ {"label": tab[i], "value": i+2} for i
in range(len(tab)) ],
                value= 2, style= {'display': 'block'}),
        ]),
        html.Br(),
        html.Button('Create User', id='submit-val', n_clicks=0),
        html.Div(id='container-button-basic3', style= {'display': 'block'})
    ])
])

```

Chat-bot JARVIS

Application des méthodes Agiles

Présentation

- Jean-Pierre Maffre
- En formation Data IA à l'école Microsoft by Simplon
- Recherche d'alternance à compter de mai 2021

Compétences

- Gestion de base de données SQL – NoSQL
- Programmation divers langages (Python, C, PHP, ...)
- Connaissance en algorithmes de machine learning

Organisation du projet 1/2

Agence Stark

Développement et S. M.



Anthony Bonfils



Constant Junior Amos



Jean-Pierre Maffre



Ludo Randon



Myriam Bouhlel



Olivier Prince

Équipe Clients

Clients et P. O. à "mi-temps"



Stéphanie PUTTAGIO



Adrien DULAC



Anne-Laure MEALIER

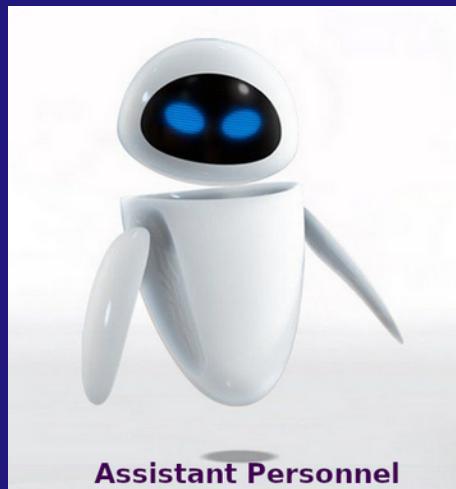
Les équipes

Organisation du projet 2/2

Le projet

Besoin client

Créer un chat-bot fonctionnel sur Discord utilisable par la communauté.



Gestion du projet

Respect de la méthode agile
Scrum (rôles, rituels, outils).

Livrable

Codes livrés sur Github. Product Backlog à jour.

Évolution

Gestion multi-langue. Doter le bot de personnalité. Réponse à des questions complexes.

La demande client – Product backlog

Extrait PBI

- Projet organisé sur 2 semaines
- 1 sprint / semaine
- Sprint 1 composé de 3 users stories
- Sprint 2 composé de 3 users stories livrable + 3 autres « bonus »

Thèmes	US	Spr.	En tant que	Je veux / peux	Afin de	Commentaires
Livrables	US1	1	Développeur	Extraire des données et alimenter une BdD avec un schéma cohérent	Rendre accessible les données pour l'alimentation d'un bot et y faire de futur requêtes	Permettre de retrouver rapidement des réponses à des questions à partir d'une recherche textuelle tout en contrôlant les limites physique de la machine. Exemple de résultats attendus : Création d'un base de donnée "stackexchange" avec deux collections : une collection questions contenant toutes les questions ainsi qu'une liste d'ID unique des réponses associées plus les meta données de la question (tags et thème)une collection answers contenant toutes les réponses ainsi qu'un ID unique de la question plus les meta données (nombre de votes etc)
Livrables	US2	1	Utilisateur	Interagir avec un BOT sur DISCORD	Avoir une discussion basique, d'en comprendre le fonctionnement, ou de lui demander de l'aide via une commande spéciale.	Exemple de résultats attendus : > user : Salut > bot : Salut user > user : Au revoir > bot : Au revoir user > user : /help > bot : Je suis capable de dire : * bonjour * au revoir * répondre à vos questions * afficher de l'aide avec la commande /help
Livrables	US3	1	Utilisateur	Poser des questions précises au BOT	d'avoir des réponses pertinentes	Exemple de résultats attendus : user > How to append a value in a numpy array ? bot > append() creates a new array which can be the old array with the appended element. I think it's more normal to use the proper method for adding an element : a = référence : https://stackoverflow.com/a/732977/4223749
Livrables	US4	2	Utilisateur	poser des questions plus poussées au bot	Enrichir la qualité des réponses obtenues à l'aide d'une conversation Exemple de résultats attendus : permettant de préciser le contexte.	Exemple de résultats attendus : permettant de préciser le contexte. you specify it to be in the dataframne. user> How to append a value in a python list bot> do you talk abour earthscience ? user> no bot> do you talk abour programmation ? user> yes ! bot> You can use df.loc[i], where the row with index i will be what référence: https://stackoverflow.com/a/24888331/4223749
Livrables	US5	2	Client	Avoir des feedbacks de ma communauté	imaginer des axes d'amélioration du produit	
BONUS	US6	2	Client	Augmenter la pertinence de mon bot et sa personnalité	instaurer une vraie proximité de avec sa communauté Discord	
BONUS	US7	2	Apprenant/développeur	réaliser des tests unitaires	valider mon code avant de le déployer	Exemple de résultats attendus >test.py 9 test passed, 1 test failed
BONUS	US8	2	Client	discuter en Français avec le bot	toucher une communauté d'utilisateurs encore plus grande	

Planification - Scrum poker

Le principe :

Après avoir "découpé" les users stories (fonctionnalités) en tâches, nous estimons la difficulté (ou le temps de développement) de chaque tâche.

La somme donnera la difficulté ou le temps de travail sur le sprint qui va commencer.

The screenshot shows a digital estimation tool for Scrum Poker. At the top, there are six boxes representing story points: 5, 8, 13, 20, 40, and 100, each with a 'Submit' button below it. Below this is a 'Results' section with a 'CLEAR ROOM' button, a 'DELETE ESTIMATES' button, and a 'HIDE' button. A table lists the names of team members and their assigned story points:

Name	Story Points
● Anthony BONFILS	40
● JP	100
● buu	40
● Constant junior Amos	20
● Olivier	40
● Myriam	
● Myriam Bouhlel	40

État d'avancement du projet – Trello (Kanban)

Exemple Trello

Les tâches nécessaires pour réaliser les fonctionnalités sont regroupées ici.

4 états possibles :

- À faire
- En cours
- En test
- Réalisé

The image shows a Trello board titled "StarkChatBot". The board is organized into four columns: "To Do", "In Progress", "Test", and "Done".

- To Do:** Contains a button "+ Add a card".
- In Progress:** Contains one card titled "Bonus : French dataset".
- Test:** Contains one card titled "US3 Fonctionnalité TextSearch".
- Done:** Contains three cards:
 - "Création, optimisation modèle (optimisation hyperparamètres,...)" (due Feb 16 - Feb 16)
 - "Chatter bot : Intégration"
 - "Nettoyage BDD"

A large, abstract, metallic sculpture of a dog's head is visible in the background of the board.

La petite réunion du matin – Daily stand up

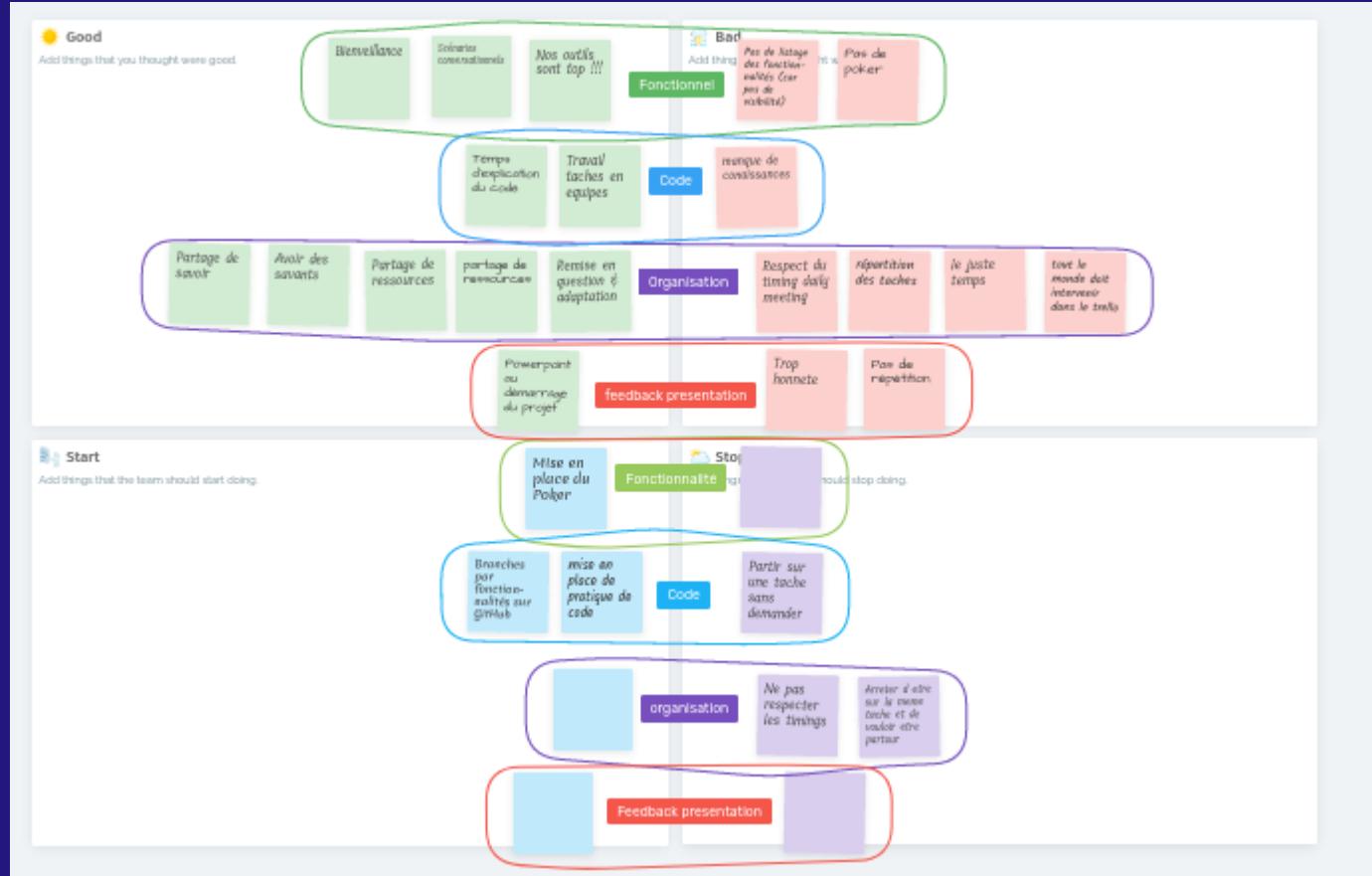
Réunion de début de journée en temps limité (3/4 minutes max par personnes)

- Chacun exprime ce qu'il a fait
- Ce qu'il va faire pour la journée
- Ses points de blocage
- Son état d'esprit

SPRINT 2						
15/02/2021	Anthony 😊	Constant	Jean-Pierre 😊	Ludia 😊	Myriam (SM)	Olivia
Diane						
Yé-Dié						
Warning						
Questions to PD						
16/02/2021	Anthony 😊	Constant 😊	Jean-Pierre (SM) 😊	Ludia 😊	Product Backlog Sprint	Product Backlog Sprint
Diane	Product Backlog Sprint	Product Backlog Sprint	Product Backlog Sprint	Product Backlog Sprint	Product Backlog Sprint	Product Backlog Sprint
Yé-Dié	réunion optimisation module fonctionnalité hyperparamètres... Chatter bot: Vieille + Code	réunion optimisation module fonctionnalité hyperparamètres... Chatter bot: Vieille + Code	réunion optimisation module fonctionnalité hyperparamètres... Chatter bot: Vieille + Code	réunion optimisation module fonctionnalité hyperparamètres... Chatter bot: Vieille + Code	réunion optimisation module fonctionnalité hyperparamètres... Chatter bot: Vieille + Code	réunion optimisation module fonctionnalité hyperparamètres... Chatter bot: Vieille + Code
Warning		Problème d'installation (dépendances)	Temps d'exécution très long			
Questions to PD						
17/02/2021	Anthony 🚗	Constant (SM)	Jean-Pierre 😊	Ludia 😊	Myriam 😊	Olivia 😊
Diane	Database + Synchron + trainable modèle	Chatterbot: vieille + Code	Chatterbot	Prise en feedback	Prise en feedback	Gestion de données DB + fonctions
Yé-Dié	designer le cockpit + vieille	Chatterbot + connexion	Chatterbot: pas possible les fonction	Connexion le feedback sur DB et charge git + repartir ultérieur dans charge git + repartir ultérieur dans fonction DB + vieille	feedback: NIK + Vieille (je n'arrive pas à faire fonctionner) + vieille	designer fonction de gestion de données
Warning		problème d'installation	interface le chatterbot			
Questions to PD						
18/02/2021	Anthony 😊	Constant 😊	Jean-Pierre 😊	Ludia (SM) 🏴	Myriam 😊	Olivia 😊
Diane	MAP model detection topic de la Create DB	Chatter bot fais intégration	Chatter bot fais intégration	Prise en feedback rapport US	Prise en feedback rapport US	Gérer les données
Yé-Dié	Explorateur model formulaire model	Intégration chatterbot	Élever la BD à niveau	Prise FEEDBACK rapport US	Prise FEEDBACK rapport US	Nettoyage interface DB
Warning						
Questions to PD						

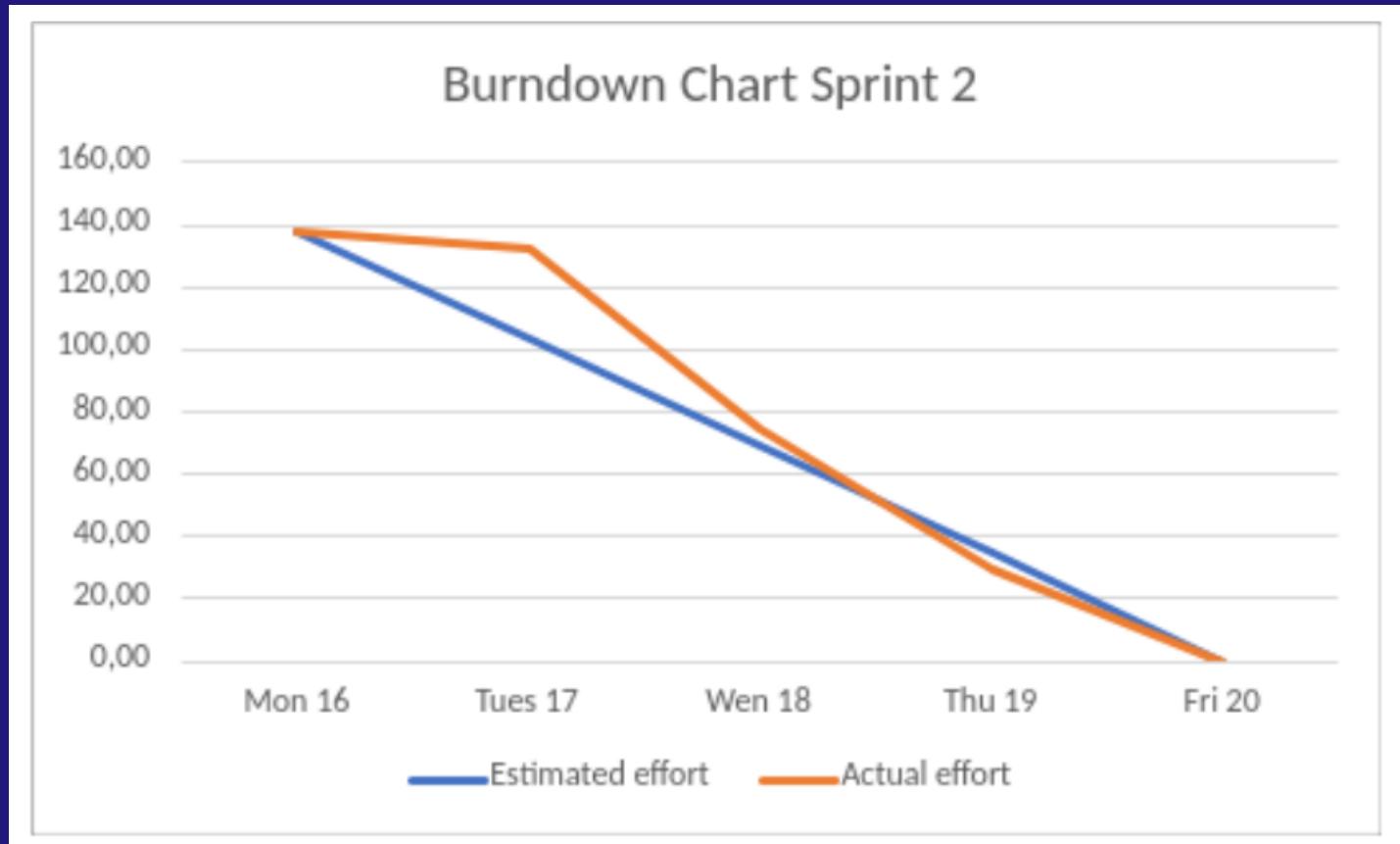
Bilan de fin de sprint - Rétrospective

[https://metroretro.io/
board/
LBTNUUFTO7GO#](https://metroretro.io/board/LBTNUUFTO7GO#)



Burn down chart

Reprend sous forme graphique les données du scrum pocker et les compare à l'avancement réel du projet.

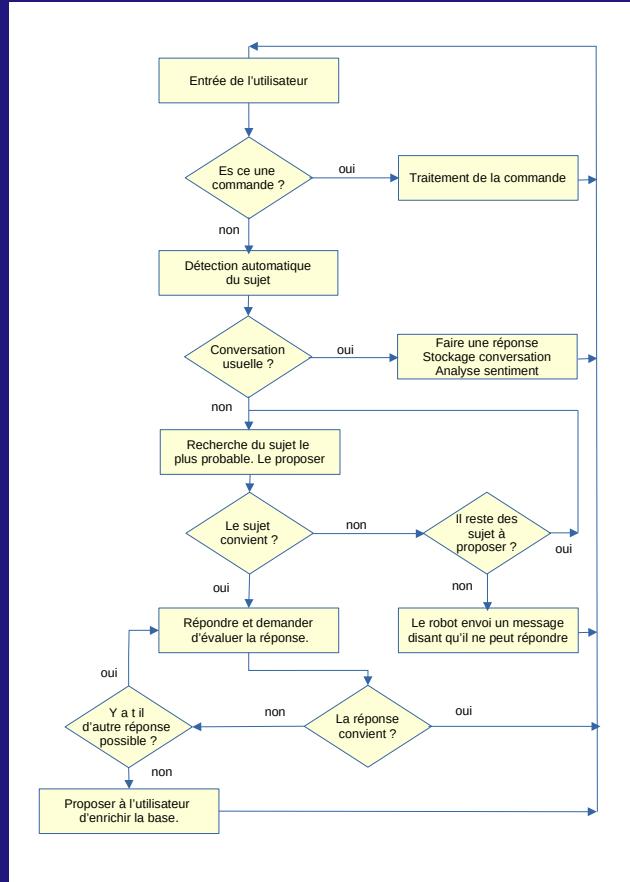


Présentation du sprint au client - Review

Schéma fonctionnel

Entrée utilisateur

- Si c'est une commande, l'exécuter
- 1^{er} niveau de conversation
- Niveau de conversation plus complexe
- Analyse du sentiment de l'utilisateur
- Retour utilisateur sur la pertinence de la réponse proposée par Jarvis.



Présentation du sprint au client - Review



J.A.R.V.I.S. BOT Hier à 13:04

Hey Human !

I would be pleased to help you on any topic in astronomy,
earthscience, electronic, engineering and Space fields !

Please feel free to address me your concern any time.
For more information you may just type : \help

Actually, my masters are working on an amazing BOT Challenge brief !
If you are curious, you may download the brief here :

https://cdn.discordapp.com/attachments/783660084395769887/808333411948429322/Brief-IA-Methodes-Agiles_-_Sprint_1.pdf

Have a nice day !

Démo

