Machine Learning

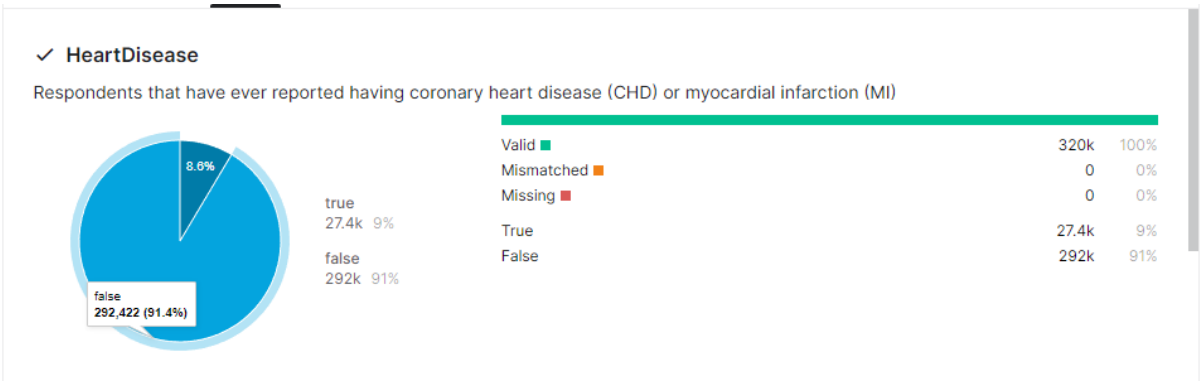
Personal Key Indicators of Heart Disease

by Adrian-Paul CARRIÈRES, Alice HIRON, Alexandre HORVILLE, Amine ADDAJOU

1. Pré analyse des données

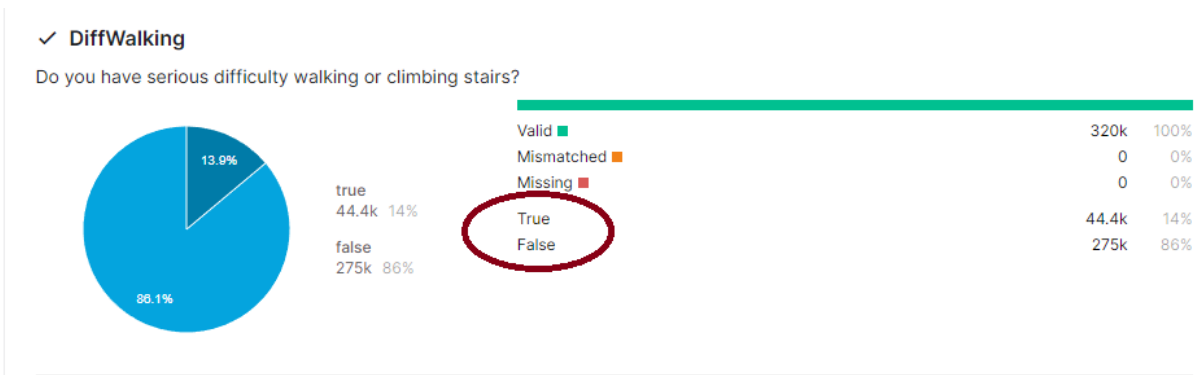
Dans le cadre de notre projet en machine learning, nous avons dans un premier temps exploré les différentes données de notre fichier CSV. En effet, avant de se lancer dans le code, il nous faut comprendre les données.

Après les différentes observations, nous avons remarqué certains points. Le premier étant que le fichier n'était pas équilibré sur Heart Disease (problème cardiaque).



En effet, nous avons 91,4% de personnes ne présentant pas de problème cardiaque. Cela peut poser un problème dans les résultats de nos différents modèles.

Dans un second temps, nous avons constaté que certaines valeurs étaient catégorisées. Par conséquent, nous devons transformer les colonnes catégorisées en valeur définie par exemple :



La colonne diff walking présente comme valeur True ou False, nous devons alors transformer ces valeurs en valeur numérique.

À partir de ces constats, nous avons décidé de procéder à un rééquilibrage et un “nettoyage” des données, ce qui nous permettra de mieux alimenter nos modèles.

2. Rééquilibrage, nettoyage des données et analyse :

    a. Rééquilibrage des données :

Pour le rééquilibrage des données, nous avons utilisé “Numpy” qui est une librairie de python, qui permet de manipuler ce qu’on appelle un “dataset” qui a pour sources notre fichier CSV. Par la suite, nous avons décidé de séparer en 2 notre dataset, une partie qui contient les “Heart Disease” à “Yes” et l’autre moitié qui est à "No".

```
df_heart_disease = df[df['HeartDisease'] == "Yes"]
df_no_heart_disease = df[df['HeartDisease'] == "No"]
```

Sachant que notre déséquilibre se situe dans les “Heart Disease” à “No”, c'est-à-dire les personnes qui ne sont pas victimes de problèmes cardiaques, nous avons donc décidé de retirer aléatoirement la différence des “Heart Disease” à “No” et “Heart Disease” à “Yes”.

```
#On choisit les lignes à retirer aléatoirement
drop_indices = np.random.choice(df_no_heart_disease.index, size=len(
    df_no_heart_disease)-len(df_heart_disease), replace=False)

#On rééquilibre
df_no_heart_disease = df_no_heart_disease.drop(drop_indices)
```

On se retrouve alors avec un rééquilibrage aléatoire de notre dataset ce qui peut être intéressant pour nos modèles.

    b. Nettoyage des données et analyse :

Comme dit dans la première partie, nous avons certaines colonnes qui sont catégorisées par des valeurs définies. Nous avons donc décidé de les changer pour pouvoir alimenter correctement nos modèles.

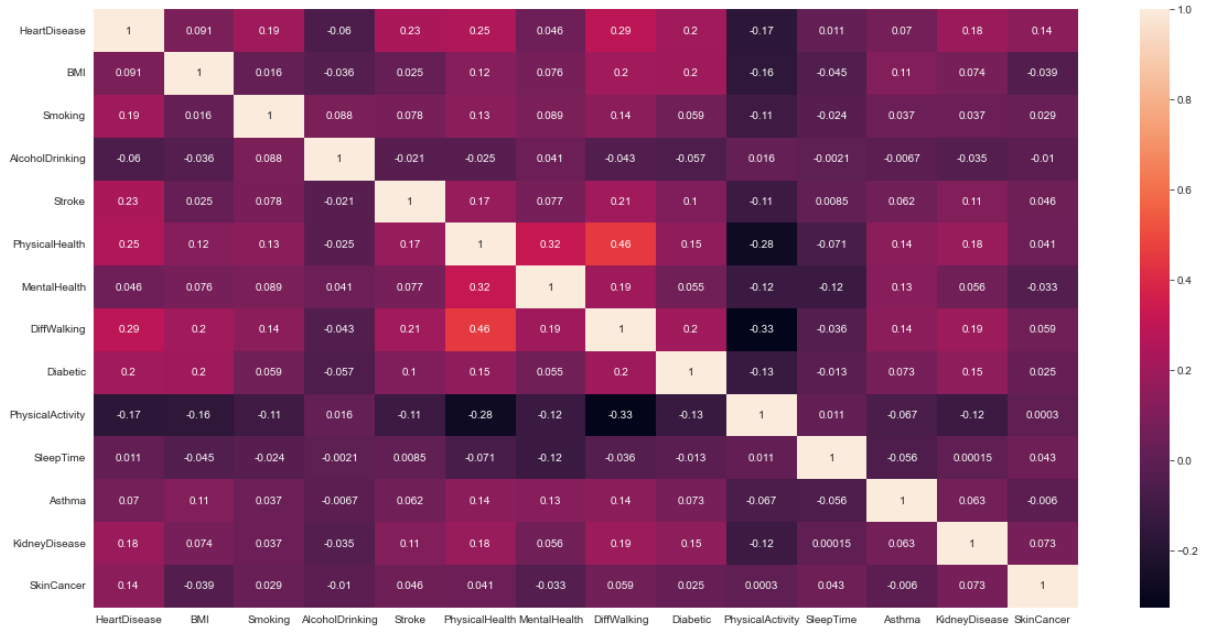
```
#Column_to_change représente les colonnes qui sont catégorisées
column_to_change = ["HeartDisease", "Smoking", "AlcoholDrinking",
"Stroke","DiffWalking", "Diabetic", "PhysicalActivity", "Asthma",
"KidneyDisease", "SkinCancer"]

#Par la suite nous remplaçons les “Yes”, “No” par 1 ou 0
```

```
#En ce qui concerne le diabète nous avons remplacé les valeurs écrites
#par des valeurs numériques

d = dict()
for c in column_to_change:
    d[c] = {"No": 0, "Yes": 1, "No, borderline diabetes": 2,
            "Yes (during pregnancy)": 3}
df = df.replace(d)
```

Une fois le nettoyage de données effectué, nous pouvons procéder à l'analyse. Pour cela, nous avons utilisé la librairie “seaborn” qui va nous permettre de visualiser la corrélation entre les différentes colonnes pour comprendre ce qui provoque les problèmes cardiaques.



La matrice de corrélation indique les valeurs de corrélation, qui mesurent le degré de relation linéaire entre chaque paire de variables. Les valeurs de corrélation peuvent être comprises entre -1 et +1. Si les deux variables ont tendance à augmenter et à diminuer en même temps, la valeur de corrélation est positive. Lorsqu'une variable augmente alors que l'autre diminue, la valeur de corrélation est négative.

3. Les modèles

a) XGBoost Classifier with Bagging and Boosting

Le résultat obtenu du modèle de XGBoost, résultat d’un ensemble de modèles plus simple et plus faible afin de fournir une meilleure prédiction, est le suivant :

```
#Bagging :
Train score: 0.694013882365221
Test score: 0.6932537749634681
Accuracy : 0.7570628348757915
```

```
# Adaboost :
Train score: 0.49950420124210637
Test score: 0.5011568436434486
Accuracy: 0.7570628348757915
```

b) Decision Tree Classifier with Bagging and Boosting

Le modèle arbre décisionnel, basé sur un schéma ayant la forme d'un arbre, sous forme de Data possibles d'une série de choix interconnectés, renvoie les prédictions suivantes :

```
Train score: 0.6736078492771777
Test score: 0.6759620068192889
Accuracy: 0.6918533852898198
```

c) Naive Bayes Classifier

Le résultat de la classification naïve bayésienne, modèle de probabilités, renvoie le résultat suivant :

```
Accuracy: 0.742510959571359
```

d) Logistic Regression

Ce modèle, basé sur la technique de modélisation linéaire, renvoie une accuracy suivante :

```
Accuracy:  0.7631514856307842
```

e) Random Forest Classifier

Ce modèle machine learning permet de choisir au hasard des caractéristiques et fait des observations pour construire une forêt d'arbres de décision et de calculer la moyenne des résultats.

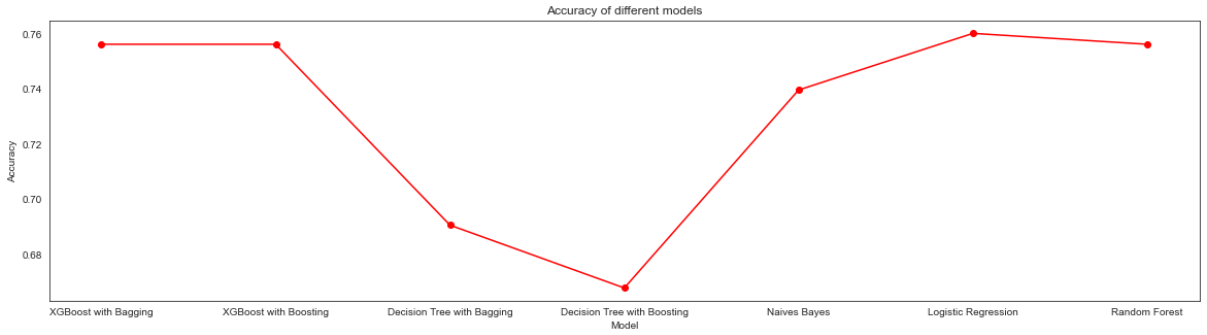
Le concept veut qu'un grand nombre d'arbres non corrélés crée des prédictions plus précises qu'un arbre de décision individuel. Cela s'explique par le fait que le volume d'arbres contribue à protéger des erreurs individuelles.

Les chiffres sont :

```
Random Forest Train accuracy: 0.673 Test accuracy 0.677
```

4. Conclusion et critique du modèle :

Nous avons constaté, après l'utilisation des différents modèles, que les prédictions étaient plus ou moins basses.



Ces résultats peuvent venir par nos méthodes de nettoyage. Par exemple, rééquilibrer de manière aléatoire le dataset peut provoquer un manque de données pour nos modèles de machine learning.

Nous n'avons pas essayé tous les modèles : le KNN est difficilement utilisable avec autant de points et le Naive Bayes Classifier est utilisé avec des données textuelles.

Nous avons aussi essayé d'utiliser le RFE pour pouvoir sélectionner les features les plus intéressantes. Malheureusement, nous n'avons pas vraiment compris ce que le RFE choisissait comme catégories (certaines features plus corrélées à la maladie cardiaque étaient retirées tandis que d'autres non). De plus, en relançant nos modèles avec le RFE, nous n'avons pas observé d'amélioration sur nos modèles.

Autre remarque, le dataset n'était peut-être pas assez volumineux pour pouvoir faire du machine learning dessus.