# Description du jeu de données

(<https://plu.mx/plum/a?mendeley_data_id=wmhctcrt5v&theme=plum-bigben-theme> )

Notre jeu de données contient 1319 observations et 9 variables, dont la variables cibles.

Une image contenant texte, reçu, capture d’écran, Police

Description générée automatiquement

*Signification de chaque variable \**

Une image contenant texte, capture d’écran, Police, reçu

Description générée automatiquement

Une étude du dataset nous révèle que les individus qui le composent ont en moyenne 59 ans.

66% d’hommes et 34% de femmes

*Etc : a dvpt avec la signification de chaque variable*

Une image contenant texte, capture d’écran, Rectangle, diagramme

Description générée automatiquement

61% de cas positifs et 39% de cas négatifs

Une image contenant texte, capture d’écran, diagramme, carré

Description générée automatiquement

# Preprocessing

Le preprocessing , ou le pré-traitement des données, est une étape cruciale. L’objectif est de rendre les données utilisables pour l’apprentissage et l’implémentation des modèles.

Le prétraitement des données inclue :

* La suppression des valeurs aberrantes
* La gestion des valeurs manquantes
* La normalisation ou la standardisation des variables numériques
* L’encodage des variables qualitatives / catégorielles

Il s’agit donc de corriger les erreurs (outliers, bruits…) pour obtenir des données précises et cohérentes pour des modèles fiables.

Dans notre cas voici les traitements opérés :

* Nettoyage des valeurs aberrantes

*Thomas je te laisse expliquer ce que t’as fait et pour quoi on garde que preprocess2*

* Encodage

L’encodage consiste à convertir la façon dont l’information est présentée. En claire, c’est la façon de passer d’une variable qualitative à une variable de type numérique, tout en préservant l’information.

La seule variable dont nous avons changé l’encodage est notre variable cible « Résult ». Nous avons utilisé la « LabelEncoder » de la bibliothèque sklearn qui, suivant la documentation, est réservé à la variable cible. Il permet de transformer les catégories en valeurs binaires.

* Train-test-split

# Analyse comparative

On cherche le modèle qui obtiendra les meilleurs résultats. Etant donné que l’on veut pouvoir prédire la présence ou l’absence de maladie cardiaque, il faut définir le type d’erreur que l’on cherche à minimiser :

* Les faux positifs : prédire la présence de maladie cardiaque chez une personne à tort.
* Les faux négatifs : prédire l’absence de maladie cardiaque chez une personnes à tort.

Dans notre cas l’erreur la plus grave serait de prédire l’absence de maladie cardiaque à tort, donc on va chercher à minimiser le nombre de faux négatifs.

Ainsi pour évaluer nos modèles nous utiliserons le f1-score, une métrique qui fait la moyenne entre les valeurs du recall et de la précision.

## Apprentissage-supervisé

Les modèles d’apprentissage supervisé se base sur des données étiquetées

* Arbre de décision

*Cette classification repose sur des règles déterminées par une suite de tests sur la valeur des attributs qui permettent de créer des sous-ensembles de données. L’objectif est de déterminer les tests, les règles qui permettent à l’issue de tous les tests d’obtenir des sous-ensembles de données appartenant à la même classe (ou presque).*

*L’implémentation du modèle nous permet d’obtenir l’arbre ci-dessous :*

Une image contenant diagramme, ligne, conception

Description générée automatiquement

Figure 1: Arbre de décision

Nous pouvons constater que l’ensemble des variables sont utilisées dans l’arbre.

Une image contenant texte, Tracé, ligne, nombre

Description générée automatiquement

Une image contenant texte, diagramme, ligne, capture d’écran

Description générée automatiquement

La matrice de confusion et le rapport de classification nous apprennent que :

* + Le modèle prédit correctement 100 cas négatifs sur 102 et 161 cas positifs sur 162.
  + 99% des cas positifs sont correctement prédits
  + 99% des prédictions positives sont effectivement des cas positifs
  + Il n’y a qu’un seul faux négatif

Une image contenant capture d’écran, Caractère coloré, diagramme, texte

Description générée automatiquement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.99 | 0.98 | 0.99 | 102 |
| 1 | 0.99 | 0.99 | 0.99 | 162 |

F1 = 0.9908

* Forêt aléatoire

Une image contenant texte, ligne, Tracé, diagramme

Description générée automatiquementUne image contenant capture d’écran, ligne, diagramme, texte

Description générée automatiquement

Une image contenant capture d’écran, Caractère coloré, carré, Rectangle

Description générée automatiquement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.98 | 0.98 | 0.98 | 102 |
| 1 | 0.99 | 0.99 | 0.99 | 162 |

La matrice de confusion et le rapport de classification nous apprennent que :

* + Le modèle prédit correctement 100 cas négatifs sur 102 et 160 cas positifs sur 162.
  + 99% des cas positifs sont correctement prédits
  + 99% des prédictions positives sont effectivement des cas positifs
  + Il n’y a que 2 faux négatif

F1 = 0.9877

* Adaboost

Une image contenant capture d’écran, Caractère coloré, texte, diagramme

Description générée automatiquement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.99 | 0.97 | 0.98 | 102 |
| 1 | 0.98 | 0.99 | 0.99 | 162 |

La matrice de confusion et le rapport de classification nous apprennent que :

* + Le modèle prédit correctement 99 cas négatifs sur 102 et 161 cas positifs sur 162.
  + 99% des cas positifs sont correctement prédits
  + 99% des prédictions positives sont effectivement des cas positifs
  + Il n’y a qu’un seul faux négatif

F1 = 0.9877

* Machine à vecteur de support

Une image contenant texte, capture d’écran, jaune, diagramme

Description générée automatiquement

Une image contenant capture d’écran, texte, diagramme, Rectangle

Description générée automatiquement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.639 | 0.40 | 0.49 | 102 |
| 1 | 0.98 | 0.85 | 0.76 | 162 |

F1 = 0.7645

* K-nearest neighbors (KNN)

Une image contenant capture d’écran, texte, Caractère coloré, diagramme

Description générée automatiquement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.47 | 0.46 | 0.47 | 102 |
| 1 | 0.68 | 0.67 | 0.67 | 162 |

F1 = 0.6687

* Naïve Bayes

Une image contenant capture d’écran, texte, Caractère coloré, diagramme

Description générée automatiquement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1 | Support |
| 0 | 0.53 | 1 | 0.69 | 102 |
| 1 | 1 | 0.43 | 0.60 | 162 |

F1 = 0.6034

Une image contenant Rectangle, Tracé, ligne, diagramme

Description générée automatiquement

## Apprentissage non-supervisé

* K-Means Clusturing
* Clusturing Hierarchique Agglomératif
* Clusturing DBSCAN
* Clusturing HDBSCAN

# Sélection du modèle

## Apprentissage supervisé

## Une image contenant texte, diagramme, ligne, capture d’écran Description générée automatiquement

De manière générale ce les modèles à base d’arbre et AdaBoost qui obtiennent les meilleures performances. Leurs accuracy sont tous égaux (raison de plus pour ne pas l’utiliser comme métrique). Néanmoins, un modèle se détache des autres, l’Arbre de Décision avec un f1 de 99.08%.

Afin de confirmer ce choix et d’améliorer si possibles les performances de notre classifieur nous avons décidé d’ajuster les hyperparamètres, de faire un apprentissage par validation croisée et de faire du sur-échantillonnage, puisque notre jeu de donnée est déséquilibré. Même si le déséquilibre concerne la classe positive.

## Apprentissage non-supervisé