# 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.

Une autre étape importante avant de placer à l’implémentation des données est la séparation entre données d’apprentissages et données de test. Cela permet de limiter l’effet de l’overffiting. On entraine les modèles sur une partie des données (train-set) et on évalue sa performance sur les données qui n’ont pas été utilisées lors de l’apprentissage (test-set). En effet, sinon il suffit que le modèle « apprenne » les données pour pouvoir les classer correctement, mais il sera incapable de classer correctement des données qu’il n’aura jamais vu.

# 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

* 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 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 |

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

F1 = 0.9908

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

Description générée automatiquement

Néanmoins ce graphique nous permet de dire qu’il y a un peu d’overfitting. La courbe bleu présente l’évolution du F1 score avec le nombre de données, sur les données d’apprentissage, la courbe orange sur les données de test.

Le modèle ne fait aucune erreur sur les données d’apprentissage (toujours à 100%), mais sur les données de test on peut voir que les performances sont plus faibles même si elles restent au-dessus de 97%. On constate également que les performances augmentent avec le nombre de données.

* Forêt aléatoire

Le RandomForest est une méthode d’ensemble qui consiste a entrainer plusieurs arbres de décisions et à voter pour la classe majoritaire.

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

Description générée automatiquement

Le graphique ci-dessus, présente l’importance accordée à chaque variable par le modèle. Parmi les 7 variables, Troponin est celle qui a le plus d’importance avec CK-MB.

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

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

Description générée automatiquement

On constate également de l’overfitting.

* Adaboost

C’est un modèle de Boosting, c’est-à-dire qu’il entraine un classifieurs sur des échantillons différents en prenant en compte et en corrigeant les erreurs à chaque fois. Ici le modèle utilisé est celui par défaut, l’arbre de décision.

Le graphique si dessous nous permet de visualiser l’importance des classifieurs faibles qui est la même à chaque fois. Les variables les plus importantes sont :

* CK-MB et Troponin

Comme pour l’arbre de décision, mais cette fois les autres variables ont plus d’impact sur les résultats.

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

Description générée automatiquement

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.
* 98% 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

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

Description générée automatiquement

De la même façon que pour les précédents modèles, il y a de l’overfitting, bien que les performances soient très bonnes.

* Machine à vecteur de support

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

Description générée automatiquementUne image contenant texte, capture d’écran, Caractère coloré, Graphique

Description générée automatiquement

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

Description générée automatiquementUne image contenant texte, capture d’écran, graphisme, Graphique

Description générée automatiquement

Ces graphiques nous permettent de visualiser la séparation des classes dans l’espace caractéristique. On peut voir qu’en fonction des variables, la séparation n’est pas nécessairement linéaire. Ce qui pourrait expliquer les faibles performances du modèle.

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 |

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

* Le modèle prédit correctement 41 cas négatifs sur 102 et 138 cas positifs sur 162.
* 98% des cas positifs sont correctement prédits
* 85% des prédictions positives sont effectivement des cas positifs
* Il y a 24 faux négatifs

F1 = 0.7645

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

Description générée automatiquement

Il n’y a pas d’overfitting ici, mais les performances sont moins bonnes. Il y a plus de faux négatifs.

* 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 |

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

* + Le modèle prédit correctement 47 cas négatifs sur 102 et 109cas positifs sur 162.
  + 68% des cas positifs sont correctement prédits
  + 67% des prédictions positives sont effectivement des cas positifs
  + Il y a 53 faux négatifs

F1 = 0.6687

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

Description générée automatiquement

Les performances de ce modèle diminuent avec le nombre de données et les performances sur le test-set sont moins bonnes que sur le train-set.

* 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 |

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

* Le modèle prédit correctement 102 cas négatifs sur 102 et 70 cas positifs sur 162.
* 100% des cas positifs sont correctement prédits
* 0.43% des prédictions positives sont effectivement des cas positifs
* Il y a 92 faux négatifs

Ce modèle prédit essentiellement les cas négatifs, puisque qu’il classe 194 observations dans cette classe et 70 dans l’autre. De ce fait, même si tous les cas négatifs sont bien identifiés, il y a beaucoup de faux négatifs. Or c’est le type d’erreur que nous cherchons à éviter.

On peut expliquer ces résultats en visualisant les probabilités à priori et à postériori.

A priori, il y a plus de cas positifs dans le jeu de données et donc plus de chances de tomber sur un individu atteint de maladies cardiaques. Mais lorsque que l’on prend en compte les variables, on voit que la probabilité de tomber sur individu sain est plus élevé.

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

Description générée automatiquement

F1 = 0.6034

Une image contenant texte, diagramme, Tracé, ligne

Description générée automatiquement

L’analyse des performances dans le train et le test set ne révèle pas d’overfitting, mais une performance qui diminue avec le nombre de données.

## 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