STT3795
Fondements théoriques en science des données

Si Da Li (20086868)
Ronnie Liu (20154429)
Van Nam Vu (20170148)

Remis à
Guy Wolf

25 avril 2022

1 Introduction

Les maladies cardiovasculaires sont reliées avec le coeur ainsi que les vaisseaux sanguins. Cet ensemble de maladies est composé de plusieurs terminologies différentes dépendamment de l'organe alimenté par les vaisseaux sanguins. Par exemple, quand on se référe aux vaisseaux sanguins qui alimentent le cerveau, il s'agit de maladies cérébro-vasculaires. Cependant, lorsque ces derniers alimentent le muscle cardiaque, il s'agit de cardiopathies coronariennes.

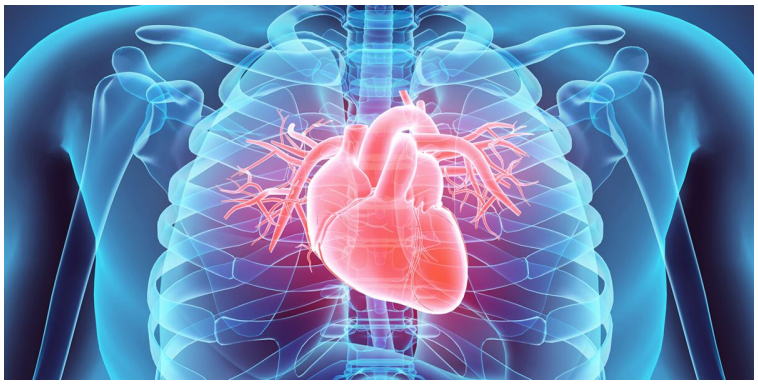

FIGURE 1 – Maladies cardio-vasculaires

Les maladies cardiovasculaires sont "la première cause de mortalité dans le monde [...], soit 31% de la mortalité mondiale totale"[1]. Il est donc essentiel de savoir comment prévenir ce type de maladie chez les patients afin de diminuer le taux de mortalité. Il existe déjà des méthodes diagnostiques qui tiennent en compte de facteurs comportementaux dans le but d'évaluer les maladies cardiovasculaires. En effet, l'influence du tabagisme, de la forte consommation d'alcool, de l'obésité, ainsi que de la malnutrition sont seulement quelques facteurs qui peuvent nous donner des indications sur la présence ou non de ces maladies. Cependant, ces méthodes ne sont pas suffisantes étant donné que la liste des facteurs comportementaux est longue et que le temps est limité quant aux observations des patients.

2 Objectifs

Pour ces raisons, au lieu de se fier uniquement aux facteurs comportementaux, nous avons décidé de créer un modèle optimal permettant de prédire la présence des maladies cardiovasculaires chez différents individus à l'aide de l'apprentissage automatique.

On essaie de combiner plusieurs méthodes vues au cours STT3795 afin de déterminer laquelle sera plus optimale avec notre jeu de données.

En d'autres mots, si on compare la qualité des deux types de classifieurs (forêt aléatoire et SVM) selon différents ensembles de données (données standardisées, traitées sous PCA, traitées sous Isomap), on aura six modèles différents à comparer.

3 Description des données analysées

3.1 Description générale du jeu de données

Source de données[2] : `https://www.kaggle.com/yasserh/heart-disease-dataset`

Le jeu de données contient 13 attributs sur 303 patients qui sont les informations et les indices des patients. La dernière colonne est constituée des valeurs 0 (non-présence de la maladie cardio-vasculaire) ou 1 (présence de la maladie).

Jeu de données original[3] : `https://archive.ics.uci.edu/ml/datasets/heart+disease`

Le jeu de données utilisé dans notre expérience est une partie de l'ensemble de données sur la détection des maladies cardiovasculaires chez les patients. Originalement, il contient 75 attributs et le jeu de données vient de quatre régions différentes : Cleverland, Hongrie, Suisse ainsi que VA Long Beach. Afin de simplifier le prétraitement, nous avons pris le jeu de données de Kaggle qui contient 13 attributs au lieu de 75. En outre, toutes les informations obtenues des 303 patients viennent de Cleveland.

3.2 Informations par rapport aux attributs

Les attributs sont de types qualitatifs ou quantitatifs. Il faut les traiter de différentes façons en fonction de leurs types.

- Variables qualitatives : sex, cp, fbs, restecg, exang, slope, ca, thal
- Variables quantitatives : age, trestbps, chol, thalach, oldpeak

Voici les informations par rapport à chaque attribut utilisé dans notre expérience :

- age : âge en années
- sex : sexe du patient
  - 0 : femme
  - 1 : homme
- cp : type de douleur à la poitrine :
  - 0 : angine typique (ressource du sang au coeur diminuée)
  - 1 : angine atypique (non reliée avec le coeur)
  - 2 : douleur non angineuse (spasmes à l'oesophage)
  - 3 : asymptomatique ('silent angina' : aucun symptôme de maladie)
- trestbps : pression artérielle au repos en mm Hg
- chol : niveau de cholestérol en mg/dL
- fbs : glycémie à jeun supérieur à 120 mg/dL :
  - 0 : non
  - 1 : oui
- restecg : résultats électrocardiographiques au repos
  - 0 : normal
  - 1 : abnormalité dans les ondes ST-T
  - 2 : hypertrophie probable à la ventricule gauche
- thalach : fréquence cardiaque maximale

— exang : angine causée par l'exercice :
  — 0 : non
  — 1 : oui
— oldpeak : dépression ST induite par l'exercice par rapport au repos
— slope : pente du segment ST d'effort maximal
  — 1 : pente positive (meilleure fréquence cardiaque à l'aide des exercices)
  — 2 : pente nulle (coeur en santé)
  — 3 : pente négative (coeur n'étant pas en santé)
— ca : nombre de vaisseaux sanguins coloriés par fluoroscopie (entre 0 et 4)
— thal : Un trouble sanguin appelé thalassémie
  — 0 : circulation sanguine normale 0
  — 1 : circulation sanguine normale 1
  — 2 : "fixed defect"
  — 3 : "reversable defect"

NOTE : Nous ne sommes pas certains de la raison pour laquelle il y a deux valeurs différentes (0 et 1) pour représenter la circulation sanguine normale quant à l'attribut *thal*. Nous avons décidé de ne pas modifier cela étant donné qu'il se peut qu'il y a quelques nuances entre les valeurs 0 et 1 qui ne sont pas mentionnées dans la documentation de ce jeu de données.

3.3 Analyse générale du jeu de données

Nous avons 13 attributs pour 303 patients avec 5 attributs numériques et 8 attributs catégoriques. En séparant les données en fonction de leurs classes (présence ou non d'une maladie cardiovasculaire), nous avons 138 patients qui n'ont pas de maladies cardiaques, tandis que 165 en ont. On peut voir que les données sont assez balancées. En outre, il ne contient aucune valeur manquante, donc l'étape d'imputation n'est pas nécessaire.

Statistiques sommaires des attributs

	age	trestbps	chol	thalach	oldpeak
count	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	131.623762	246.264026	149.646865	1.039604
std	9.082101	17.538143	51.830751	22.905161	1.161075
min	29.000000	94.000000	126.000000	71.000000	0.000000
25%	47.500000	120.000000	211.000000	133.500000	0.000000
50%	55.000000	130.000000	240.000000	153.000000	0.800000
75%	61.000000	140.000000	274.500000	166.000000	1.600000
max	77.000000	200.000000	564.000000	202.000000	6.200000

	sex	cp	fbs	restecg	exang	slope	ca	thal
count	303	303	303	303	303	303	303	303
unique	2	4	2	3	2	3	5	4
top	1	0	0	1	0	2	0	2
freq	207	143	258	152	204	142	175	166

FIGURE 2 – Statistiques sommaires des attributs numériques et catégoriques

Matrice de corrélation des attributs

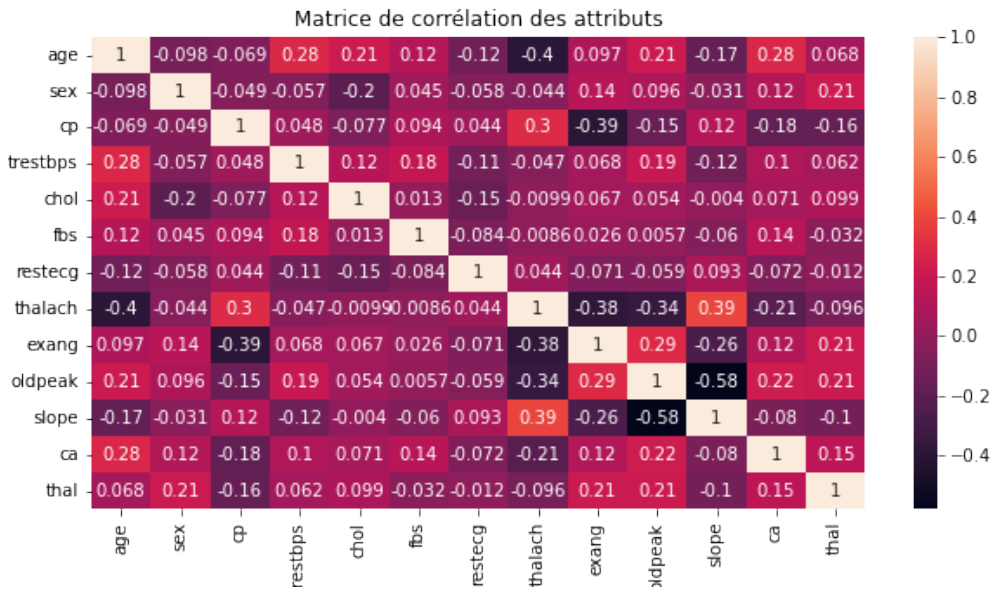

FIGURE 3 – Matrice de corrélation des attributs

Les valeurs de corrélation minimale et maximale sont respectivement -0.58 (entre *slope* et *oldpeak*) et 0.39 (entre *slope* et *thalach*). Étant donné que la valeur absolue de ces deux valeurs ne sont pas tout à fait proches de 1, nous avons décidé de garder tous les attributs avant de passer à l'application des méthodes supervisées et non supervisées. Si on remarque que les résultats ne semblent pas être satisfaits, une des tentatives serait donc d'enlever soit l'attribut *slope* soit *oldpeak* puisque les deux attributs donnent une force de corrélation de 0.58.

4 Méthodologie

4.1 Librairies utilisées en Python :

pandas, numpy, sklearn, graphviz, matplotlib.pyplot

4.2 One hot encoding pour prétraitement

One Hot Encoding est une méthode qui permet de transformer les valeurs des attributs catégoriques en valeurs binaires. Cette méthode permet de traiter les attributs comme les entiers, mais un des inconvénients c'est qu'il va créer beaucoup d'attributs supplémentaires. On a utilisé OneHotEncoding dans la librairie Sklearn. On va appliquer l'encodage *One-Hot* sur toutes les variables catégoriques.

thal	thal0	thal1	thal2	thal3
1	0.0	1.0	0.0	0.0
2	0.0	0.0	1.0	0.0
2	0.0	0.0	1.0	0.0
2	0.0	0.0	1.0	0.0
2	0.0	0.0	1.0	0.0

FIGURE 4 – One Hot Coding pour attribut 'thal'

4.3 Standardisation des données

On se sert de la standardisation *Min-Max* après avoir fait l'encodage One-Hot sur les attributs catégoriques. L'objectif de cette étape est de centrer les valeurs entre 0 et 1 afin d'avoir de meilleures performances dans l'apprentissage supervisé. En outre, puisque les attributs catégoriques sont binaires, la méthode *Min-Max* ne modifie pas les valeurs binaires de ces derniers. Autrement dit, elle va uniquement modifier les variables numériques. On applique cette méthode de standardisation à l'aide de la fonction *MinMaxScaler* dans la librarie *sklearn* qui transforme les données de la façon suivante[4] :

$$X_{standardized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

4.4 Forêt aléatoire

Normalement, l'utilisation d'un seul arbre de décision ne garantit pas d'avoir un résultat optimal. Donc, on veut appliquer l'algorithme de forêt aléatoire qui contient plusieurs arbres de décision qui nous permettent d'avoir une meilleure prédiction. Dans la discussion, on va uniquement comparer la performance entre la forêt aléatoire et SVM.

Les **hyperparamètres** qu'on va varier sont le nombre d'arbres (10, 30, 50, 100, 200) et la profondeur de chaque arbre (11, 15, 20, 30, 50).

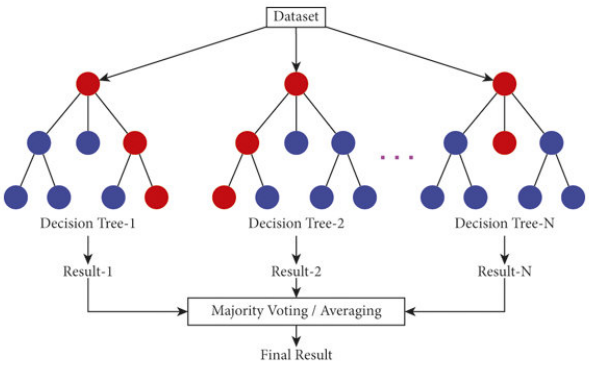

FIGURE 5 – Schéma de forêt aléatoire[5]

4.5 SVM

Dans ce rapport, nous allons utiliser le SVM linéaire à marge souple afin qu'on puisse contrôler la valeur de l'hyperparamètre C. On trouve les paramètres optimisés à l'aide de cette condition :

$$arg\,min_{w,b,\xi_1,...,\xi_N} \|w\| + C \sum_{i=1}^{N} \xi_i$$
$$\text{tels que } \forall i \in \{1, ..., N\}, c_i(\langle w, x_i \rangle - b) \geq 1 - \xi_i \text{ et } \xi_i \geq 0$$

Lorsque la valeur de C est petite, on pénalise moins les erreurs, donc les variables d'écart seront plus grandes. Cependant, lorsque C augmente, on pénalise plus les erreurs, donc les variables d'écart vont être plus petites.
En d'autres mots, c'est la borne supérieure de toutes les variables slack. Une valeur plus grande nouse donnerait plus de liberté dans la séparation.
Nous allons tester différentes valeurs pour **l'hyperparamètre** : $C = \{0.1, 10, 1000, 1\ 000\ 000\}$.

4.6 PCA - Principal component analysis

En raison de l'encodage One-Hot sur les attributs catégoriques, nous avons un total de 30 attributs. Les méthodes non-supervisées comme PCA et Isomap permettent de réduire la dimension de l'ensemble de données. Quant à PCA (*Principal Components Analysis*), cette méthode projette les données sur un sous-espace linéaire à une dimension plus faible (nouvelle système d'axes : les composantes principales). Elle est réalisée à l'aide de la décomposition SVD de la matrice X, notre jeu de données.

On choisit de plonger les données en **2 dimensions**.

4.7 ISOMAP

Isomap est une autre méthode non supervisée permettant de réduire la dimension du jeu de données. Ici, on se sert de la matrice des distances géodésiques à l'aide de l'algorithme du plus court chemin (Dijkstra)[6]. Ensuite, on applique l'algorithme MDS avec la matrice des distances géodésiques comme la valeur d'entrée de ce dernier.

On choisit de plonger les données en **2 dimensions avec 5 voisins les plus proches**.

4.8 Entraînement des données

En commençant par les données brutes, on emploie l'encodage One-Hot pour les attributs catégoriques et on standardise les données à l'aide de la méthode *MinMaxScaler* de *sklearn*. Ce sont donc nos données traitées (standardisées). Par la suite, on compare les performances en fonction de différentes méthodes supervisées (Forêt aléatoire et SVM Linéaire à marge souple) et de différentes méthodes non supervisées (PCA et Isomap).

On sépare les données en trois parties : Entraînement, Validation et Test (40%-30%-30%). Pour les méthodes supervisées, on choisit les valeurs d'hyperparamètres qui donnent le meilleur résultat pour les données de validation (en terme d'exactitude) tout en évitant le sur-apprentissage. Par la suite, on emploie le modèle dans les données de test avec les hyperparamètres trouvées.

5 Résultats

5.1 Forêt aléatoire

5.1.1 Données traitées

```
[88] #arbre decision
     decision_tree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=10)
     decision_tree = decision_tree.fit(X_train, y_train)
     print('Accuracy = ',decision_tree.score(X_test, y_test))

     Accuracy =  0.7582417582417582
```

FIGURE 6 – Niveau accuracy de Arbre de décision

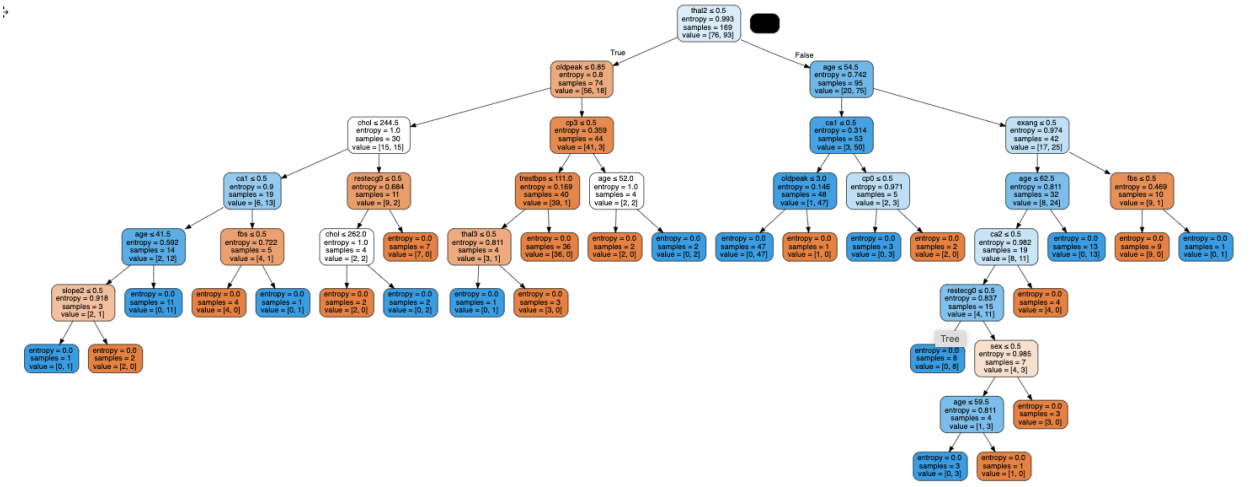

FIGURE 7 – Arbre de décision

```
[251] #foret aleatoire
      from sklearn.ensemble import RandomForestClassifier

      foret_aleatoire = RandomForestClassifier(n_estimators=50, max_depth=11)
      foret_aleatoire.fit(X_train, y_train)
      print('Accuracy val = ',foret_aleatoire.score(X_val, y_val))
      print('Accuracy test = ',foret_aleatoire.score(X_test, y_test))

      Accuracy val =  0.813953488372093
      Accuracy test =  0.7912087912087912
```

FIGURE 8 – Niveau accuracy de Forêt aléatoire

5.1.2 Forêt aléatoire avec PCA

```
#arbre decision avec PCA
decision_tree_pca = tree.DecisionTreeClassifier(criterion='entropy', max_depth=10)
decision_tree_pca = decision_tree_pca.fit(X_train, y_train)
print('Accuracy = ',decision_tree_pca.score(X_test, y_test))
```

```
Accuracy =  0.7802197802197802
```

FIGURE 9 – Niveau accuracy de Arbre de décision avec PCA

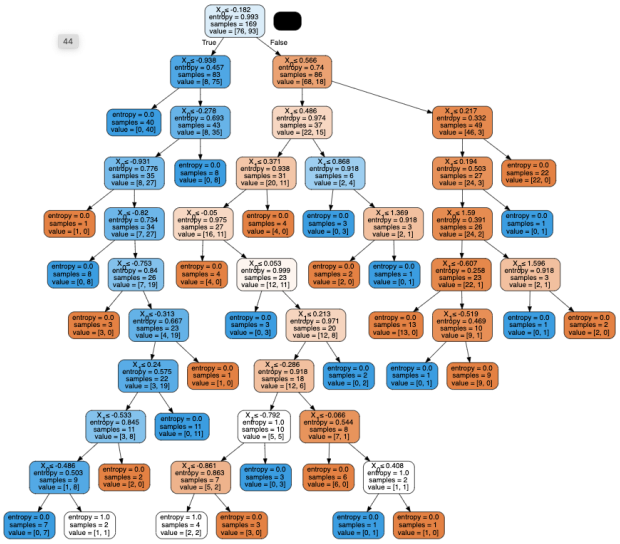

FIGURE 10 – Arbre de décision avec PCA

```
#foret aleatoire avec PCA
from sklearn.ensemble import RandomForestClassifier

foret_aleatoire_pca = RandomForestClassifier(n_estimators=50, max_depth=11)
foret_aleatoire_pca.fit(X_train, y_train)
print('Accuracy val = ',foret_aleatoire_pca.score(X_val, y_val))
print('Accuracy test = ',foret_aleatoire_pca.score(X_test, y_test))
```

```
Accuracy val =  0.7674418604651163
Accuracy test =  0.8241758241758241
```

FIGURE 11 – Niveau accuracy de Forêt aléatoire avec PCA

5.1.3 Forêt aléatoire avec ISOMAP

```
#arbre decision pour isomap
decision_tree_isomap = tree.DecisionTreeClassifier(criterion='entropy', max_depth=10)
decision_tree_isomap = decision_tree_isomap.fit(X_train, y_train)
print('Accuracy = ',decision_tree_isomap.score(X_test, y_test))
```

```
Accuracy =  0.7802197802197802
```

FIGURE 12 – Niveau accuracy de Arbre de décision avec ISOMAP

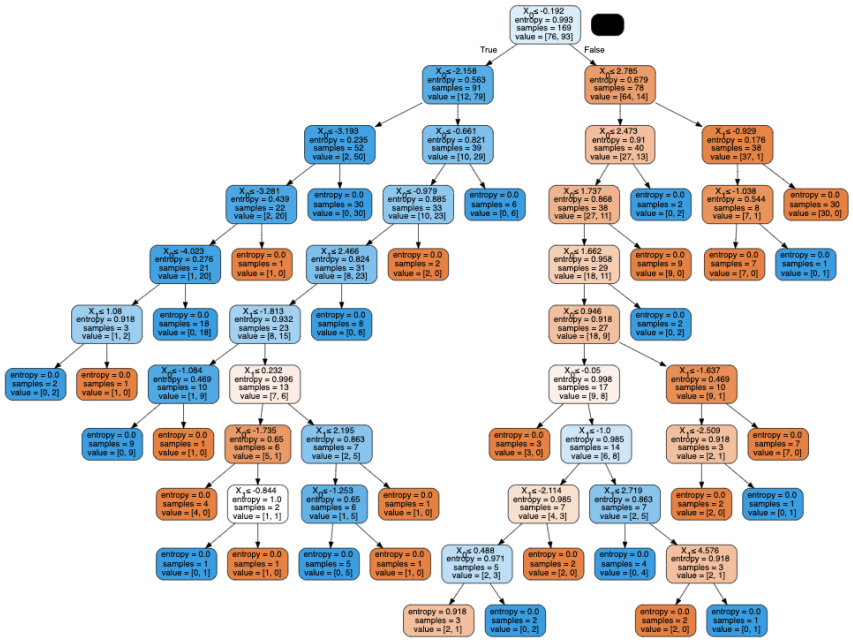

FIGURE 13 – Arbre de décision avec ISOMAP

```
#foret aleatoire pour isomap
from sklearn.ensemble import RandomForestClassifier

foret_aleatoire_isomap = RandomForestClassifier(n_estimators=50, max_depth=11)
foret_aleatoire_isomap.fit(X_train, y_train)
print('Accuracy val = ',foret_aleatoire_isomap.score(X_val, y_val))
print('Accuracy test = ',foret_aleatoire_isomap.score(X_test, y_test))
```

```
Accuracy val =  0.813953488372093
Accuracy test =  0.8351648351648352
```

FIGURE 14 – Niveau accuracy de Forêt aléatoire avec ISOMAP

5.2 SVM

5.2.1 Données traitées

Avec 30% des données disponibles utilisées pour les essais, nous sommes possibles d'obtenir une précision maximale de 84.62% avec les données traitées avec $C = 10$ pour SVM linéaire.

5.2.2 PCA

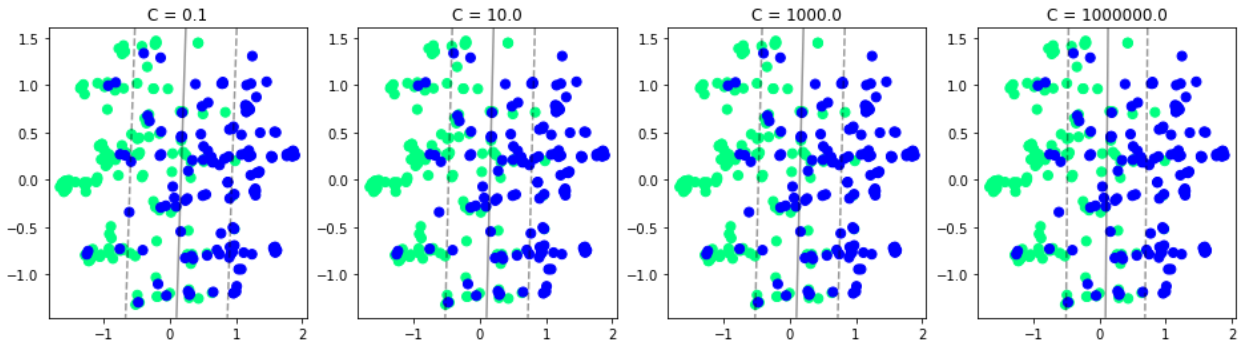

FIGURE 15 – SVM linéaire avec données traitées sous PCA.

5.2.3 ISOMAP

L'algorithme ISOMAP a été appliqué sur les données traitées avec 2 composantes et 5 voisins les plus proches.

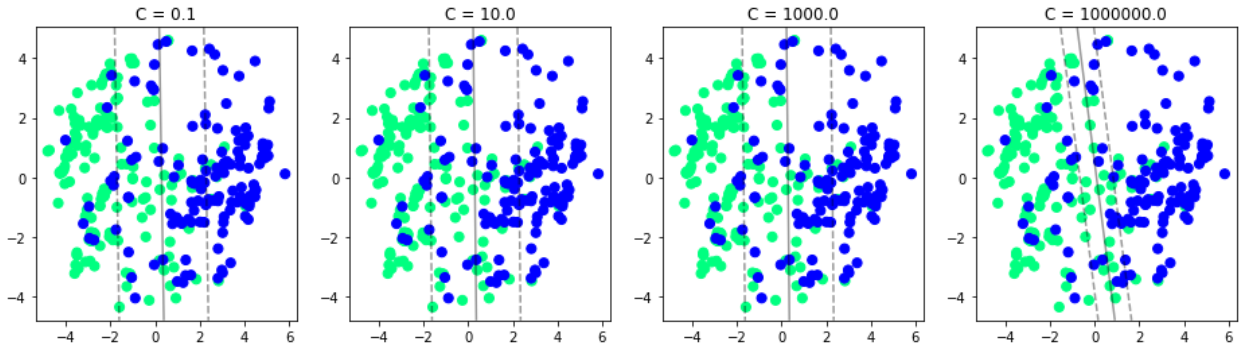

FIGURE 16 – SVM linéaire avec données traitées sous Isomap.

NOTE : les points turquoises représentent les patients ayant une maladie cardiovasculaire, tandis que ceux qui sont bleus n'ont pas de maladies cardiaques.

5.3 Résumé

	Forêt aléatoire	SVM linéaire
Données traitées	79.12%	84.62%
PCA	82.418%	80.22%
Isomap	83.516%	82.42%

TABLE 1 – Résumé de l'exactitude des modèles sur les données de test

Hyperparamètres trouvées en fonction de l'exactitude des données de validation :

- Forêt aléatoire : 50 arbres et profondeur de 11
- SVM Linéaire à marge souple : C = 10 pour données traitées et Isomap, et C = 0.1 pour PCA

6 Discussion

Impression générale

Tout d'abord, à l'aide de la table 1, on peut voir que la meilleure méthode est SVM Linéaire à marge souple avec les données traitées (exactitude de 84.62%). En deuxième place, il s'agit de la forêt aléatoire avec Isomap (exactitude de 83.516%). On s'attendait à avoir une meilleure performance avec la forêt aléatoire étant donné que cette dernière est une méthode qui est non linéaire, tandis que SVM sous-entend que les données sont linéairement séparables.

Forêt aléatoire

Pour la forêt aléatoire, étant donné qu'on génère plusieurs arbres (50 arbres) pour un seul modèle, cela devient une méthode de "bagging", permettant de créer un modèle plus robuste. Nous avons constaté cela étant donné que l'idée initiale était de comparer les performances entre les arbres de décision ainsi que SVM Linéaire. Par exemple, pour les données traitées, nous avons obtenu une exactitude de 75.824% pour l'arbre de décision, tandis que la forêt aléatoire nous donne 79.12%. De plus, PCA et Isomap nous ont donné exactement la même performance pour l'arbre de décision qui est 78.021%, mais on a 82.418% (PCA) et 83.516% (Isomap) pour la forêt aléatoire. Bref, pour la forêt aléatoire, on constate que l'utilisation de distances géodésiques nous mène à de meilleurs résultats dès qu'on plonge les données en deux dimensions.

SVM Linéaire à marge souple

Cependant, pour SVM Linéaire à marge souple, sans aucune réduction de dimensionnalité, nous avons déjà une meilleure performance (84.62%) que la forêt aléatoire avec Isomap. En outre, nous avons l'impression que réduire la dimension des données nous donne des résultats moins bons (80.22% et 82.42%) que celui avec les données traitées. Une explication derrière cela est peut-être la valeur de la dimension réduite. Il se peut que l'exactitude de PCA et d'Isomap sera plus élevé dès qu'on prend une dimension supérieure à 2 afin de maximiser encore plus la variance des données.

PCA et Isomap : influence sur la distribution des données

En ce qui concerne des méthodes non supervisées, lorsqu'on observe les graphiques des figures 15 et 16, on peut voir la distribution des données après plongement en deux dimensions. Pour PCA, on voit que plusieurs points turquoises et bleus sont jumelés dans la même région (au milieu des graphiques). Même principe pour Isomap, mais la séparation entre les points bleus et turquoises semblent être un peu plus clair.

Pourquoi pas SVM Non-Linéaire à marge souple ?

	Temps d'exécution (en s)	Précision
Données traitées + SVM linéaire	236.62362504005432	84.62%
Données traitées + SVM non-linéaire	0.006001949310302734	76.92%
PCA + SVM linéaire	15.52930760383606	80.22%
PCA + SVM non-linéaire	8.497514486312866	76.92%
ISOMAP + SVM linéaire	116.88613080978394	79.12%
ISOMAP + SVM non-linéaire	4.166353464126587	78.02%

TABLE 2 – Comparaison du temps d'exécution de SVM avec la précision, $C = 10^6$

Nous nous sommes demandés si remplacer SVM linéaire par SVM non-linéaire (astuce de noyau - RBF) donnera de meilleurs résultats. En terme de précision, à l'aide de la table 2, SVM linéaire est toujours plus performant peu importe le prétraitement utilisé (standardisation, PCA et Isomap). D'après les figures 15 et 16, nous somme capables de déterminer que les caractéristiques sont linéairement séparables sur deux dimensions mais la présence de bruit rend ce travail difficile à automatiser. Une chose est certaine, c'est que les méthodes linéaires semblent nous donner des résultats au coût de beaucoup plus de temps de calcul.

Par contre, en prétraitant les données avec PCA ou Isomap, nous sommes capables de réduire le temps de calcul de beaucoup sans perdre beaucoup de précision. Par exemple, un test avec les données non-réduites prend 236 secondes. Ce temps de calcul est réduite à 116 secondes pour Isomap et à 15 secondes pour PCA. À noter que ce temps de calcul est négligeable pour les petites bases de données et c'est pour cela que nous avons utilisé $C = 10^6$ pour noter le temps de calcul.

L'algorithme SVM avec l'astuce du noyeau est dans toutes les scénarios becaucoup plus rapide que la méthode linéaire. C'est à cause du fait que la marge est dynamique, ce qui baisse la précision mais crée aussi un risque d'*artifacting* qui n'est pas toujours visible à l'oeil nu pour les résultats de grande dimension.

Pistes d'amélioration

Quelques autres possibilités qu'on pourrait explorer, c'est d'appliquer la méthode de cartes de diffusion étant donné que cette dernière est une autre méthode de réduction de dimensionnalité, mais de façon probabiliste. Dans nos essais, nous avons remarqué que les variables slack posaient un grand problème au temps d'exécution, surtout pour les bases de données plus grandes. Il serait pertinent de créer un algorithme moins glouton pour déterminer une marge de liberté moins précise, mais assez bonne pour éviter à utiliser une valeur de C trop grande. Finalement, puisque SVM Linéaire semble être le plus précis, on pourrait comparer cela avec d'autres modèles linéaires comme la régression logistique.

7 Conclusion

Somme toute, afin de prédire la présence ou non de la maladie cardiovasculaire chez les patients, nous conseillons d'utiliser SVM Linéaire à marge souple avec C = 10 pour avoir le meilleur résultat. Cependant, on peut tout de même avoir de bon résultats lorsqu'on applique la méthode de la forêt aléatoire avec les données traitées sous Isomap.

8 Contributions des membres de l'équipe

8.1 Si Da Li

1. Recherche du jeu de données pour le projet
2. Implémentation PCA et SVM Linéaire et Non-Linéaire
3. Rédaction du rapport

8.2 Ronnie Liu

1. Recherche du jeu de données pour le projet
2. Prétraitement des données et Isomap
3. Rédaction du rapport

8.3 Van Nam Vu

1. Recherche du jeu de données pour le projet
2. Arbre décision et forêt aléatoire
3. Rédaction du rapport

Références

[1] Maladies cardiovasculaires. (2017). WHO. https://www.who.int/fr/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)#:~:text=Les%20maladies%20cardiovasculaires%20constituent%20un,sanguins%20qui%20alimentent%20le%20cerveau

[2] Yasser H. M. (s.d.). *Heart Disease Dataset*. Kaggle. https://www.kaggle.com/datasets/yasserh/heart-disease-dataset

[3] Heart Disease Data Set. (s.d.). UCI. https://archive.ics.uci.edu/ml/datasets/heart+disease

[4] sklearn.preprocessing.MinMaxScaler. (s.d.). Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[5] Khan Yaseen, M. et al. (2021). *Illustration of Random Forest Trees* [image]. Research Gate. https://www.researchgate.net/figure/Illustration-of-random-forest-trees_fig4_354354484

[6] Dijkstra's algorithm, https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm