le cnam
Paris

Projet 9: Statlog Heart

Prédire la présence ou l'absence de maladie du cœur

---

projet réalisé par : Bargaoui Emna

UE Rcp209
Année 2021/2022

Sommaire

Introduction : **3**

Analyse et exploration des données : **4**

Description de la base de données et outils utilisés : 4

Analyse univariée des variables: 6

Analyse multivariée/bivariée des variables : 7

Pré-traitement des données : **9**

Entraînements et optimisations de modèles prédictifs: **10**

Les modèles à base d'arbre de décision : 10

L'arbre de décision 10

Un modèle ensembliste le randomforest: 13

Un modèle ensembliste le boosting: 14

Les machines à vecteurs de supports (SVM): 15

La régression logistique : 17

Les courbes precision-recall: 18

Conclusion **20**

Annexes : **21**

Introduction :

L'essor des modèles prédictifs dans le milieu de la santé a permis une amélioration considérable du système de manière générale.
En effet on constate une progression significative des parcours de soins et de la réalisation d'études et d'évaluations , tout ceci permet une meilleure prévention de future maladie ,un diagnostic plus élaboré et bien d'autres applications. C'est le développement des technologies de l'information qui a permis d'avoir des bases de données médicales exhaustives et plus complètes ,ce qui a mené au développement de modèles plus robustes et performants .

Pour ce projet nous serons amenés à résoudre un problème de classement,prédire si un individu est sujet à une maladie du cœur selon plusieurs critères collectés par des professionnelles du milieu .
Pour cela notre travail se divisera en trois principales parties , la première consistera à une analyse de nos données , une deuxième partie plus concise sur des éventuels pré-traitements et la dernière sur l'étude détaillée de plusieurs modèles abordés au cours de cette UE .

Dans le cadre de la détection d'une maladie les indicateurs d'évaluations ou mesures de performances qui seront pertinentes sont le Recall (ne pas laisser se balader des personnes ayant des maladies du coeur) le F1 score (ne pas charger les secteurs "cardiologie" des hôpitaux avec des personnes n'étant pas atteint d'une maladie du coeur) .
L'accuracy est une métrique assez risquée car il suffirait d'être en présence de données déséquilibrées pour avoir un modèle performant ce qui pourrait ne pas être le cas .

On va se fixer comme objectif d'avoir au moins un modèle avec comme performance finale pour le Recall et le F1 score d'au minimum 80%.

I. Analyse et exploration des données :

C'est une étape importante pour la construction et l'évaluation de nos futurs modèles décisionnels .Elle nous permettra de 'nettoyer' notre dataset si nécessaire , nous donnera une vue d'ensemble sur nos variables et leurs relations ,de même de gérer les données aberrantes ou les erreurs commise lors de la saisie des données.
Pour cela nous utiliserons des librairies nous offrant des fonctionnalités d'analyses mathématiques et de visualisations comme Panda et Numpy codées en python.

1. Description de la base de données et outils utilisés :

La base de données Satlog dont on dispose provient du site UCI ML .C'est une base de données qui est similaire à la célèbre Cleveland sur les maladies du cœur conçue en 1988 et utilisée dans de nombreux articles de recherches .

La base Satlog est constituée de 270 observations ,de 13 variables explicatives (features) ainsi qu'une variable cible qui est donc notre 14ème colonne.

Types des variables explicatives (numéro des colonnes)
Réel: 1,4,5,8,10,12 Ordonnées:11, Binaire: 2,6,9 Nominale:7,3,13
Variable a prédire
Absence (1) ou présence (2) d'une maladie du coeur
No missing values.
270 observations

Tableau 1: Description fourni par UCI

Pour pouvoir tirer un maximum d'informations de notre analyse de données nous avons effectué des recherches un peu plus approfondies sur chacune des variables explicatives pour comprendre un peu plus leur rôle sur le cœur et ces maladies .
A noter que ces informations sont juste à titre informatif et ne serviront en aucun cas pour une prise de décision finale.
Dans notre cas nous en avons 13 donc cela reste possible à effectuer, mais si nous avions utilisé la base de donnée initiale du Cleveland avec 75 variables ceci aurait été contraignant.

Information des variables :

1. Age :

2. sex : (0 = female,1 = male)

3. chest pain type (4 valeures)

L'angine désigne une douleur thoracique qui apparaît généralement pendant un effort ou un stress

1: Angine typique
2: Angine atypique
3: douleur non-anginale
4:Asymptomatique

4. resting blood pressur (en mm/Hg)

La mesure de la pression artérielle au repos d'un patient

5. serum cholestoral (en mg/dl)

Un haut niveau de cholestérol peut augmenter les risques de maladie cardio-vasculaire.

6. fasting blood sugar > 120 mg/dl (1 = true; 0 = false)

Mesure du taux de sucre dans le sang à jeun **.**

1 indique que nous avons dépassé le seuil minimal toléré et donc un taux de sucre alertant

7. resting electrocardiographic results (valeures 0,1,2)

Ce test peut détecter si le cœur bas doucement normal ou vite.

0: normal
1: ST-T courbe sont anormales
2: montre une potentielle présence de "left ventricular hypertrophy" par un critère de Estes

"left ventricular hypertrophy LVH" est liée à des risques de maladie cardiaque

8. maximum heart rate achieved

En dehors de l'intervalle [60 ,100] ce n'est pas normal

9. exercise induced angina (1 = yes; 0 = no)

1:angine induite par l'effort (plainte commune chez les patients cardiaques)

10. oldpeak = ST depression induced by exercise relative to rest

Un examen sur l'activite du coeur.

St depression fait référence à une constatation sur un électrocardiogramme

11. the slope of the peak exercise ST segment

En lien avec la variable oldpeak

1: pente qui monte
2: pente plate
3: pente qui descend

12. number of major vessels (0-3) colored by flourosopy

hypothèse sur les valeures 0 à 3 indiquent le nombre de vaisseaux malades

13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

Le nom de cette variable a été abrégé donc on ne peut émettre qu'une hypothèse sur le fait qu'elle fasse peut être référence au test thallium

2. Analyse univariée des variables:

Les schémas pour cette partie sont dans “Visualisation de nos variables” du fichier EDA.ipynb

- La Variable cible

  La dernière variable du dataset Absence (1) ou présence (2) est binaire et prend soit 1 ou 2 comme valeur .Nous sommes dans un problème de classification binaire, et cette variable nous permet de constater que nous sommes en présence d’un dataset qui n’est pas très déséquilibré,c'est à dire qu'il y a une proportion d’individus avec et sans maladie cardiaque (au moins une) assez proche.
  1 ~ 56%
  2 ~ 44%

- Les variables explicatives

  Nous avons affiché la distribution de nos variables continues ,age ,resting blood pressure , maximum heart rate achieved , serum cholesterol et Oldpeak.
  Elles ont toutes une distribution normale sauf pour oldpeak qui est asymétrique sur la gauche .Avoir une distribution normale de nos données permet de faciliter les calculs mathématiques de certains algorithmes utilisés dans les modèles prédictifs ,et donc de favoriser leur performance .
  On remarque une moyenne d'âge de 55 ans ce qui semble cohérent, les personnes plus jeunes sont moins exposées aux maladie cardio-vasculaire .

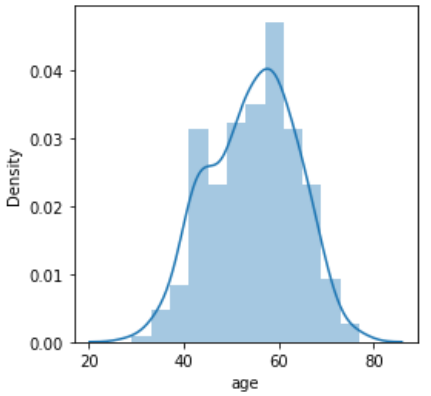


figure a : Densité de distribution de age

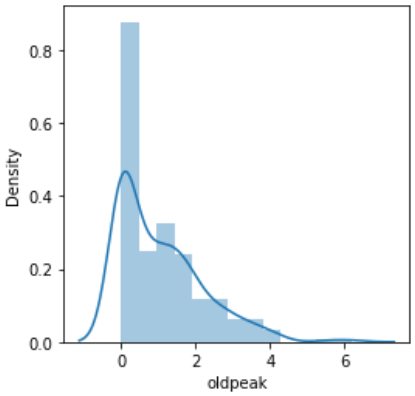


figure b : Densité de distribution de oldpeak

Avec d’autres techniques de visualisation de Pandas nous pouvons faire les constats suivants :

- On constate plus d’individus du sexe masculin que feminin

- Une grande majorité n'a pas dépassé le seuil de danger pour la variable fasting blood sugar
- rare cas ou le résultat d'une EGC nous donnes des battements de coeur anormaux (valeur 1)
- Il y'a peu de cas qui ont une douleur thoracique après un effort physique (angina induced exercice)
- De même pour la variable chest pain , il y'a une plus grande proportion pour des douleurs non liées à une angine(valeur 3 et 4),donc peu de patients ont une douleur liée à une fatigue du coeur ,mais on ne sais pas si les autres types de douleurs sont aussi liée d'une autre manière .

3. Analyse multivariée/bivariée des variables :

On veut mettre en valeur les relations dominantes entre nos différentes variables .
Nous allons d'abord analyser la relation entre la variable cible appelée aussi target et les variables explicatives dites features en utilisant des tableaux de contingences si les features ont deux ou plusieurs modalités et des histogrammes avec une ligne de distribution pour les variables continues.
Notons que l'analyse que nous effectuerons sur ces relations n'est pas robuste et qu'on pourrait être plus précis c'est à dire confirmer nos idées avec des tests statistiques(comme les tests de Students).
Les schémas pour cette partie sont dans "relation entre les variables/variable cible" du fichier EDA.ipynb.

- Target avec les variables pour les analyses sanguines:
  Les variables serum cholestoral et fasting blood sugar ne montrent aucune relation pertinente avec la target

- Target avec les variables sex et age :
  Le sex de la personne ne nous donne pas d'information particuliere car notre etude comporte plus d'homme que de femme .
  On remarque que plus on avance dans l'âge et plus on est susceptible d'avoir une maladie du coeur

- Target avec les variables pour les activités du coeur:
  (resting electrocardiographic results,oldpeak,the slope of the peak exercise ST segment,maximum heart rate achieved)
  Ils semblerait que les individus malades et non malades ont une fréquence cardiaque maximale différente,le même constat est effectué pour la variable oldpeak(figure c et d ci dessous)

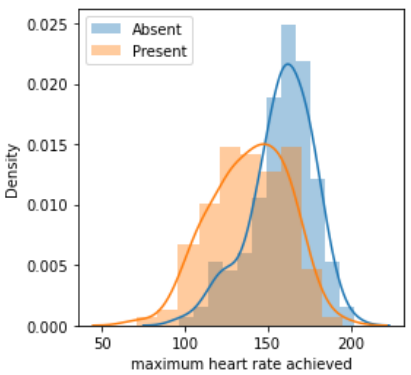


figure c : Densité de distribution

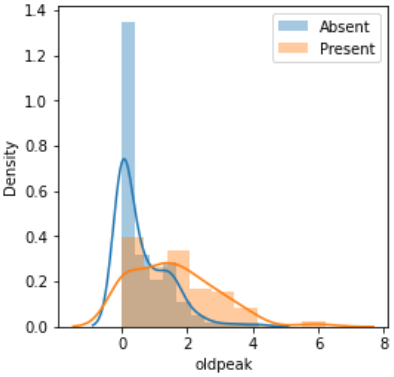


figure d : Densité de distribution

- Target avec les variables liées à la maladie angine:
  (exercise induced angina,chest pain type)
  Il semblerait que ceux qui ont une douleur asymptotique sont plus susceptibles d'avoir une maladie du coeur.De même pour ceux qui ont une valeur 1 pour la variable exercise induced angina .

- Target avec les variables d'activités sanguines:
  (resting blood pressure,number of major vessels (0-3) , thal)
  On constate qu'une grande majorité qui ont 0 vaisseau majeur n'ont pas de maladie du cœur.
  Pour la variable thal ,si elle vaut 7 une quantité non négligeable ont une maladie cardiaque et pour 3 une grand partie n'est pas malade.

Après avoir étudié la corrélation entre la target et les features nous allons faire de même mais seulement avec ces derniers.
Pour cela nous avons utilisé une matrice de corrélation (partie "Analyser les corrélations des variables explicatives" du fichier EDA.ipynb), deux variables sont corrélées dans le même sens si leur coefficient de corrélation est proche de 1 et -1 pour le sens inverse , et il sera calculé pour toutes les combinaisons de variables possibles.
On peut remarquer que les coefficients sont plutôt faibles excepté pour oldpeak et the slope of peak qui est non négligeable et vaut 63 %, généralement le seuil minimal est de 75% mais tout dépend du domaine étudié .

Enfin les variables comme serum cholestoral et resting blood pressur comportent quelques données aberrantes que l'on peut visualiser avec des boxplot , à ce stade nous n'allons pas les enlever car ils ne semblent pas être des erreurs mais plutôt des cas rares, ce qui doit être courant dans le domaine de la santé .

II. Pré-traitement des données :

Tout d'abord nous allons nous assurer qu'il ne manque aucune donnée comme indiqué dans la description sur UCI. La fonction isna() de panda nous indique bien que pour toutes nos variables aucune donnée ne manque , donc aucune technique d'imputation ne sera nécessaire.
De même , l'étude des relations entre nos variables explicatives nous a montré que les variables oldpeak et the slop of peak ont une corrélation non négligeable , nous serions donc tenter de retirer l'une d'entre elles du dataset . Or toutes deux ont un certain impact avec la variable cible (coefficient de corrélation de 40%) donc supprimer l'une d'entre elles est risqué et on pourrait perdre des informations importantes . Cependant nous verrons plus en détail dans la dernière partie des techniques de réduction de dimension et sélection de variables pour certains modèles et leur impact sur leurs performances .

Avant toutes opérations avec nos modèles , nous avons d'abord divisé notre jeux de données en deux parties avec la méthode train_test_split de sklearn: le trainset qui est notre jeu d'entraînement et testset qui est notre jeu de donnée futurs .Nous avons choisi une proportion de 70%/30% ,car notre jeu de données ne contient pas énormément d'échantillons (270) . De même, nous avons décidé de fixer l'état aléatoire de notre méthode (train_test_split) ,c'est-à- dire que nos données seront mélangées de la même manière à chaque fois pour comparer nos résultats de manière plus cohérente.

La plupart des modèles décisionnelles nécessitent une étape de normalisation, pour éviter l'incompatibilité des unités de mesures entre nos variables , que celles situées dans un grand intervalle numérique ne dominent pas celles qui sont dans un plus petit .
Ceci améliore la performance des modèles basés sur la notion de distance en réduisant leur biais ,comme les "Support Vector Machine "qui utilisent la distance euclidienne dans leurs calculs .De même elle optimise la convergence de l'algorithme de descente de gradient en modifiant la forme de la fonction coût , c'est un algorithme d'optimisation que l'on retrouve dans les modèles linéaire ,les perceptrons etc.
Nous utiliserons la méthode Standarscaler() de Sklearn que sur certains modèles dont on justifiera la raison dans la dernière partie, nos données futures seront transformées de la même manière que celles servies à l'entraînement pour que le modèle reçoive des données cohérentes à ce qu'il a appris .

Enfin, il est nécessaire de préciser que nous pouvons avoir des fuites de données si nous appliquons les méthodes de prétraitements sur l'ensemble du dataset , et donc il est nécessaire de les effectuer après la division en trainset et testset .Pour illustrer ces propos prenons l'exemple de la standardisation , on estime la moyenne et l'écart type (nécessaire pour modifier nos valeures) en utilisant toutes les données pour ce calcul , or si nous utilisons train_test_split après ,ceux du trainset auront des informations par exemple sur la distribution de ceux du testset.

III. Entraînements et optimisations de modèles prédictifs:

Dans cette dernière partie nous nous consacrerons à l'implémentation et l'optimisation de différents modèles de classification à l'aide de méthodes étudiées au cours de l'UE RCP209 .De même , nous comparerons leur performance sur les métriques choisis (notamment f1 score) et justifierons pourquoi certains donnent de meilleure résultat que d'autres.

Nous utiliserons la bibliothèque Scikit-learn où tous nos modèles sont implémentés avec une architecture P.O.O (orientée objet), chaque modèle a sa propre classe, et on utilise la même interface pour entraîner et évaluer .
Nous avons choisi d'utiliser les pipelines ,des chaînes de transformations qui combinent des transformers (instanciations de classes associées aux prétraitements de nos données) et un estimateur (instanciation d'un modèle) sur une même ligne de code. L'avantage est qu'ils permettent de faciliter l'optimisation du modèle (GridsearchCV) choisi plus facilement car ils utilisent la validation croisée sur l'ensemble de la chaîne .Nos données seront aussi plus sécurisées , les transformations de prétraitements sur le trainset seront les mêmes que sur le testset et évitent donc toutes fuites de données .

Enfin , pour estimer les performances nous utiliserons la validation croisée qui entraînera les données sur un trainset et les validera sur un validationset "valset" pour avoir des informations plus fiables et moins biaisées sur la capacité à prédire des modèles .

A. Les modèles à base d'arbre de décision :

1. L'arbre de décision

L'arbre de décision est un modèle de base souvent utilisé pour la classification où chaque noeud correspond à une décision , un test sur une variable explicative, et dont le classement se fait par vote majoritaire.L'avantage est qu'il est simple à interpréter on l'appelle "white box" et ne nécessite pas de pré traitement spécifique .C'est un modèle dont l'algorithme d'apprentissage CART (implémenté par défaut sur Sklearn) est basé sur la notion de probabilité et non la distance ce qui justifie la non nécessité à standardiser les données.

CART a été introduit par L. Breiman en 1984 , son principe de base consiste à chercher parmi les variables explicatives dont on dispose et leur seuil pour celles qui sont continues celles qui maximisent un gain d'information et en faire un nœud de décision . Pour cela l'algorithme calcul un index d'impureté qu'il cherchera à minimiser pour chaque nœud .
Il existe différents choix pour l'index d'impureté mais Gini est le plus souvent utilisé , il quantifie la probabilité qu'une variable soit mal étiquetée selon des lois statistiques .

Nous implémentons l'estimateur DecisionTreeClassifier avec les hyper-paramètres par défauts , Gini comme index , on ne pose pas de seuil maximal pour max_depth ni de seuil

minimal pour min_samples_split . Ce sont ces hyper-paramètres qui nous intéressent pour notre étude.

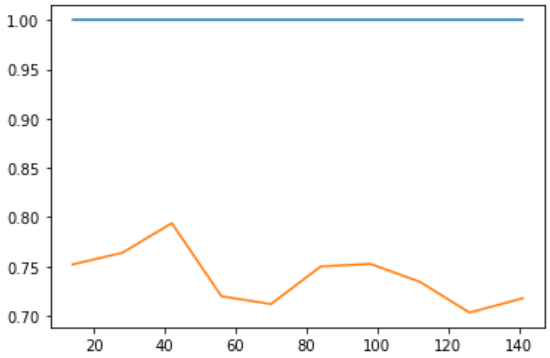
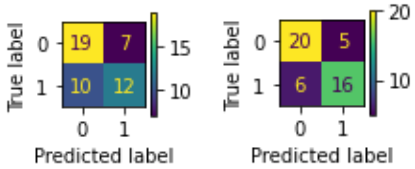


figure 1: courbe d'apprentissage-decision tree

La métrique f1 a une valeur de 1.0 sur le trainset et de 0.68 par validation croisée (figure 1 ci-dessus courbe bleu pour le trainset et orange validationset) , une performance moyenne avec un modèle qui généralise mal.
On visualise notre arbre avec la fonction plot.tree (figure générée dans le fichier projet.ipynb) , il a une profondeur de 8 et compte 21 feuilles dont un grand nombre ne contient qu'une seule observation ,elles sont trop spécifiques ,ce qui montre bien que le modèle colle beaucoup aux données d'apprentissages et est complexe d'où ce surapprentissage.

On remarque que le nœud racine (numéro 12 =variable thal) sur lequel se fait le découpage le plus important est la variable qui avait la corrélation la plus importante avec notre variable cible . De même pour ses deux nœuds enfants l'algorithme CART choisit oldpeak et chest pain type qui eux aussi ont une relation non négligeable avec target d'après notre analyse de données de la partie 2.

On peut utiliser aussi d'autres métriques de classifications comme la matrice de confusion qui visuellement nous donne un meilleure aperçu de la performance .On génère plusieurs matrices en fonction du nombre de "fold" dans la validation croisée . Prenons en deux avec des résultats distincts dont on expliquera la raison par la suite .



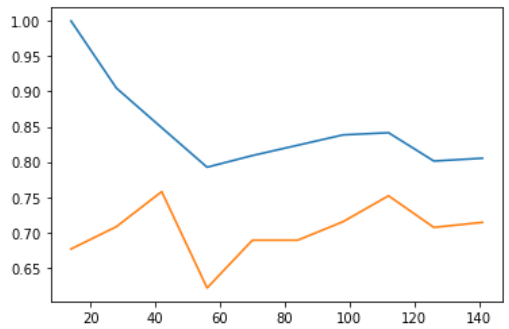
Ce qui nous intéresse le plus ce sont les cas malades , donc on va s'intéresser à l'erreur de type 2 (false-negative) calculée dans la deuxième ligne et qui nous indique que le modèle prédit incorrectement 10 individus non malades sur un total de 22 pour l'une et 6 pour l'autre ,un résultat moyen que l'on tentera d'améliorer par la suite .

De même , il est important d'ajouter que les arbres sont des estimateurs de variance élevée. On calcule pour cela l'écart type moyen sur un validationset (fonction variance du fichier projetfinal.ipynb) cela nous retourne une valeure de 3% non négligeable dans le domaine de la santé et qui montre bien une instabilité de ce model .En effet si on teste notre performance sur plusieurs découpages de notre jeu de donnée cela produit des arbres

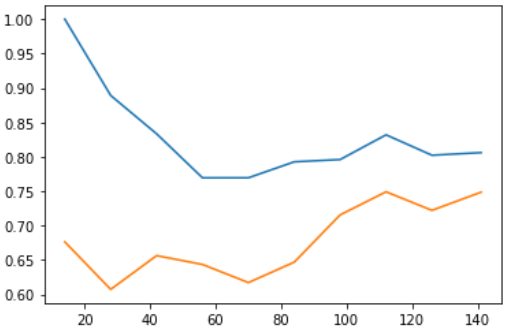
différents à chaque fois et donc une performance qui n'est pas uniforme sur l'ensemble et qui explique aussi les différents résultats pour les matrices de confusions vues.

L'étape d'optimisation d'un modèle nous permettra d'améliorer ses performances en essayant plusieurs combinaisons et valeurs possibles pour les hyper-paramètres que l'on jugera importants, et éventuellement à lutter contre le surapprentissage.
Pour tous les modèles nous utiliserons le même procédé avec la méthode RandomizedSearchCV de Sklearn qui est plus rapide et efficace qu'une simple GridSearchCv lorsqu'on a plusieurs paramètres à optimiser.

L'arbre de décision que nous avons obtenu est profond et complexe d'où le sur-apprentissage .Nous devons donc l'élaguer ,diminuer sa taille et pour cela nous optimisons les hyper-paramètres qui permettent de mettre des contraintes sur sa construction, maxdepth et min_sample_split.



f1 mean/f1 std : 0.70
recall mean/recall std : 0.67
figure 2: sans SelectKbest



f1 mean/f1 std : 0.77
recall mean/recall std : 0.77
figure 3: avec selectKbest

Nous allons utiliser la fonction Evaluate du fichier projetfinal.ipynb qui nous génère des courbes d'apprentissages et les scores moyens sur le trainset et validationset issus de la validation croisée de différentes métriques de classifications dont celles qui nous intéressent.
Ce nouveau modèle est un arbre de taille maximale 2 ,les courbes d'apprentissage de la figure 2 ci-dessus nous indiquent que le modèle généralise mieux,le score sur le trainset diminu au fur et à mesure que le nombre de données augmente . Les métriques recall et f1 valent respectivement 70% et 67% , on a une légère augmentation par rapport aux scores de base .Notons que RandomizedSearchCV ne nous donnera pas une solution optimale globale ,c'est pour cela que nous avons fixé son paramètre nombre d'itération à 50 pour affiner le plus possible nos recherches.

L'arbre de décision possède une sélection de variable intégrée grâce à l'algorithme CART mais nous pouvons quand même utiliser une méthode de sélection externe indépendante au modèle comme SelectKbest de sklearn pour voir si cela peut améliorer nos performances.
On intègre selecKbest dans notre pipeline du modèle ce qui nous permettra aussi d'optimiser sur ses hyper-paramètres en même temps que ceux de l'arbre. On remarque que travailler avec seulement 4 variables explicatives ,engendre une hausse pour f1 et recall qui sont maintenant respectivement égale à 74% et 76% (figure 3 ci -dessus). Les

courbes semblent continuer à monter avec l'ajout de données donc avoir un dataset plus grand aiderait sûrement à améliorer les performances.Ajoutons que CART est un algorithme glouton et ne nous retourne pas forcément une performance optimal globale.
Ceci nous laissera nous diriger vers des modèles plus sophistiqués et qui peut être nous permettront de combler les lacunes de ce modèle notamment le problème de la variance.

2. Un modèle ensembliste le randomforest:

On va maintenant s'intéresser aux forêts aléatoires créé en 1995 et officiellement proposées par L.Breiman et A.Culter en 2001.Ils dérivent de la méthode du bagging qui consiste à agréger plusieurs estimateurs qui sont de bon apprenants "strong learners".Le fait d'associer un grand nombre de modèle va augmenter la performance et réduire la variance finale , du à la loie des grands nombres .
Le bagging consiste à entraîner chaque estimateur sur une partie aléatoire du dataset , en utilisant une technique d'échantillonnage le "bootstrapping" .L'algorithme va replacer après chaque tirage au sort les données qui ont été sélectionnées , on aura donc une foule d'estimateur diversifiée mais qui partage certaine connaisance en commun .Le random forest vient améliorer le bagging en y ajoutant une étape aléatoire supplémentaire au niveau des variables explicatives .Elles aussi seront tirées aléatoirement pour diminuer la corrélation entre les estimateurs et donc avoir une foule encore plus nuancée.

Comme leur nom l'indique, les forêts aléatoires utilisent des arbres de décisions CART qui sont entraînées en parallèle et n'ont besoin d'aucun pré-traitement spécifique.
Avant de commencer l'analyse , nous précisons que l'on utilise pas l'erreur out-of-bag pour estimer , mais nous gardons notre procédure de validation croisée pour plus de cohérence dans la comparaison des modèles .Cependant ces méthodes sont similaires malgré que out-of-bag peut être plus efficace pour appréhender le sur-apprentissage mais dans notre cas il peut surestimer les performance car le dataset est petit .
Nous initialisons donc notre modèle avec les hyper-paramètres par défauts , le nombre d'estimateurs est fixé à 100, un "max--depth " sans seuil maximal pour la taille des arbres et un nombre de variables explicatives pour prendre les décisions "max-feature" à tirer aléatoirement fixé à √nombre-de-variables-explicatives. Comme le modèle précédent, ce sont ces hyper-paramètres sur lesquels nous allons nous concentrer pour la partie optimisation.

Le score F1 sur le trainset et validationset vaut respectivement 100% et 77%(figure 4 ci dessous),on a un amélioration par apport aux arbres de décisions .Le recall quand à lui est estimé à 75% , on a pu à ce stade de notre analyse prédire correctement 75% de personnes malades.Agréger 100 arbres décisionnelles différents a permis de combler les erreurs respectives de chacun d'entre eux durant le vote final .
Cependant le modèle est lui aussi en sur-apprentissage, cela est peut être dû à cette initialisation pour contrôler la construction des arbres de même que leur nombre .
Si nous avons des arbres très profonds avec des nœuds feuilles trop spécifiques qui comportent peu d'observations il faut alors augmenter le nombre d'estimateurs pour l'aider à moins coller aux données et donc diminuer la variance même si cela risque de ralentir l'algorithme.

L'étape finale d'optimisation nous permettra de trouver la meilleure combinaison pour des hper-paramètres , mais observons d'abord si sa variance a bien diminué par rapport au modèle précédent . En effet l'écart type indique une dispersion de 2% pour F1 contre 3.3% pour l'arbre décisionnelle .

Nous allons donc essayer divers combinaisons pour "max-feature" , "n-estimators" et "max-depth",avec 405 estimateurs de taille maximale 2 et à peu près 4 variables explicatives on obtient un score moyen pour F1 de 82% et 78% pour recall(figure 5 ci dessous).

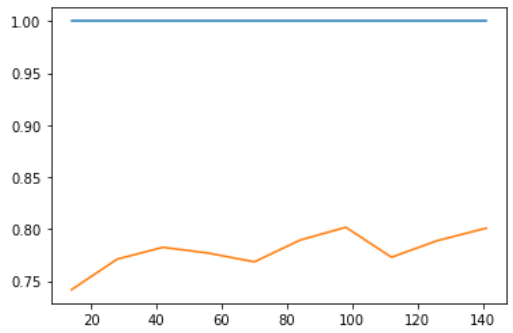


figure 4 :courbe d'apprentissage-avant optimisation

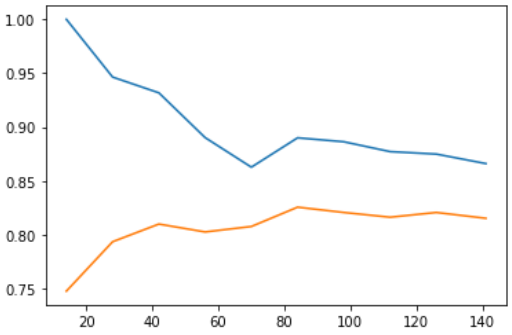


figure 5 :courbe d'apprentissage-après optimisation

Contrairement à un bagging classique le Random forest se limite à des arbres peu profond pour avoir de meilleurs résultats ce qui est notre cas ici , car une plus petite profondeur permet d'avoir moins de corrélation.Les courbes d'apprentissage indiquent que plus on ajoute de données mieux il généralise , donc si nous ajoutons plus d'observations on pourrait diminuer l'écart entre les performances du trainset et validationset. On peut ici aussi essayer un SelectKbest en amont du modèle ,mais il nous donne des résultats similaires donc on préféra garder toutes nos variables initiales .

La méthode feature-random de Scikit-learn peut nous montrer qu'elles variables ont eu un poid important dans la décision en calculant l'index de Gini(cf figure 6 annexe).On voit que les variables explicatives qui ont le plus servie pour les décisions sont ceux qui ont un pourcentage de corrélation (positif ou négatif) avec la target plus important que le reste .

Donc les forêts aléatoires sont plus robustes qu'un arbre de décision , qui avec le bon paramétrage peut donner de bons résultats , un jeu de données plus grand aurait sûrement contribué à une meilleure généralisation et de meilleure performance.

3. Un modèle ensembliste le boosting:

Une autre manière d'obtenir des ensembles de modèles diversifiés est d'utiliser une technique appelée boosting . Le principe est d'entraîner l'un après l'autre plusieurs modèles relativement faibles qu'on appelle "weak learners " en demandant à chacun d'entre eux de corriger les erreurs effectuées par son prédécesseur, les faiblesses des uns sont compensées par les faiblesses des autres . On choisira d'adopter l'algorithme Adaboost

développé par Freund et Schapire à la fin des années 90 , c'est un des algorithmes les plus populaires du boosting.
On entraine une première fois le modèle initialiser avec les hyper-paramètres par défauts sur les données d'apprentissage , nous avons donc 50 classifieurs qui sont des arbres de décisions de profondeur maximale 1 et entraînés les uns à la suite des autres . Ce type d'arbre est communément appelé "stumps" et est souvent utilisé comme classifieur faible pour le boosting.

Les 50 stumps vont essayer de corriger les observations mals classifiées par leur prédécesseurs ,celles ci sont pondérées avec un poid plus important que les autres qui sera mis à jour pour chaque classifieur .Le résultat final est décidé par un vote majoritaire pondéré par la "confiance" qui est calculée pendant le Adaboost et qui est accordée aux arbres.
Ici encore le modèle a du mal à généraliser, on a un F1 de 70% sur le validationset (figure 7 ci-dessous), on remarque qu' avec un nombre important de données la courbe sur le trainset diminue et sur le validationset augmente ,donc ajouter des observation peut être bénéfique .
De même, nous avons un faible nombre de classifieurs avec un learning rate de 1 sûrement pas adapté au nombre d'arbres, choisir une valeure plus faible permettrait de diminuer le sur-apprentissage.
L'étape d'optimisation se déroule de la même manière que pour les précédents modèles . Ici les hyper-paramètres sont le nombre d'estimateurs "n-estimators" et learning rate qui contrôle la vitesse du changement de poids dans la fonction coût de Adaboost.
Avec 307 stumps et une faible valeur pour la vitesse d'apprentissage on atteint respectivement 81% et 78% pour F1 et recall(figure 8 ci-dessous) , nous avons des résultats similaires qu'avec une forêt aléatoire.

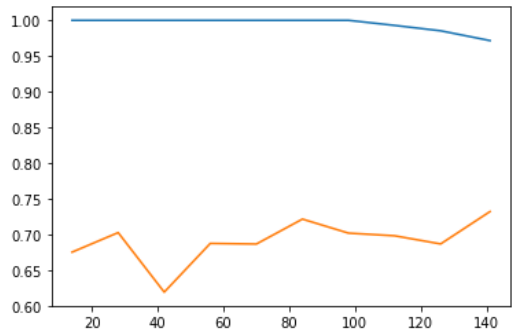


figure 7 : courbe d'apprentissage-avant optimisation

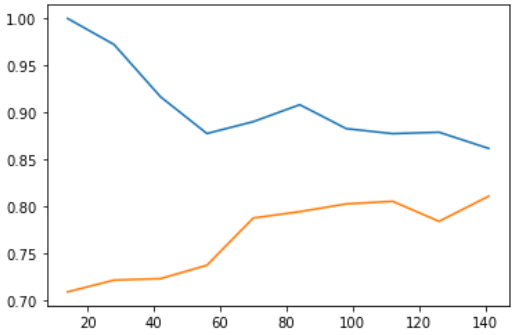


figure 8 : courbe d'apprentissage-après optimisation

Effectuer une sélection de variable avec selectKbest donne de moins bon résultats donc on garde le modele actuel.De meme, comme pour la forêt aléatoire , ce sont à peu près les mêmes variables qui ont contribué aux décisions tels que "chest-type-pain" et "thal" (cf 9 figure annexe).

B. Les machines à vecteurs de supports (SVM):

Les machines à vecteurs de supports ont été développées dans les années 90 par Vladimir Vapnik .Ils utilisent des algorithmes efficaces aussi bien en classification que en régression.L'objectif avec les SVM est de déterminer une frontière pour un problème de classification afin de séparer le jeu de données en deux groupes distincts pour notre cas tout en maximisant une marge . Ce modèle n'utilise pas toutes les observations pour construire sa frontière de décision mais seulement quelques unes bien précises les "vecteurs de supports" , ce qui le rendra plus rapide et performant.
Dans notre cas nous ne savons pas si le problème est linéairement séparable , il est donc préférable de directement utiliser les SVM avec l'astuce à noyaux , consistant à projeter les données dans un espace vectoriel de plus grande dimension où il est probable qu'il existe des séparations linéaires.
De même l'avantage du "kernel trick" est qu'il généralise aussi l'approche linéaire avec le noyau Linear Kernel.
Donc nous entraînons notre modèle sur le trainset qui sera précédé d'une étape de normalisation, avec ses hyper-paramètres de base . Le régularisateur C fixé à 1 , gamma un paramètre d'échelle qui est utilisé dans certains noyaux à 1/13 , et enfin le noyau sera gaussien et projettera les observations dans un espace de dimension infini.

On est en sur-apprentissage ici aussi même si les performances sur le validationset se sont améliorées (figure 10 ci-dessous).On pourrait d'abord penser que la cause est le paramètre C , choisir une valeure plus petite pourrait certe augmenter le taux d'erreur et donc le biais mais rendrait l'algorithme plus robuste et collerait moins aux bruits .C'est le principe de la marge souple que nous utilisons aussi dans les espaces à grandes dimension .De même , on pourrait tester d'autres types de noyaux plus simple car le gaussien sur des petits dataset peut mener à un sur-apprentissage.
Nous allons tester plusieurs combinaisons aléatoires pour les hyper-paramètres et pour le choix du noyau on se limitera au linéaire et gaussien.Nous avons un resultat interessant car c'est le noyau linéaire qui donne de meilleur performance avec C tres petit ,donc on a une marge plus grande.On se ramène à un classifieur linéaire sans changement d'espace et cela indique que notre problème se rapproche plus d'un cas linéairement séparable .De même nos courbes(figure 11 ci-dessous) indiquent qu'au début avec peu de données le modèle était en sous-apprentissage et qu'à partir d'un certains nombre de d'observations les courbes se stabilisent avec un écart qui se rétrécit et montre qu'on généralise mieux avec un score de 83% et 82% pour F1 et recall.

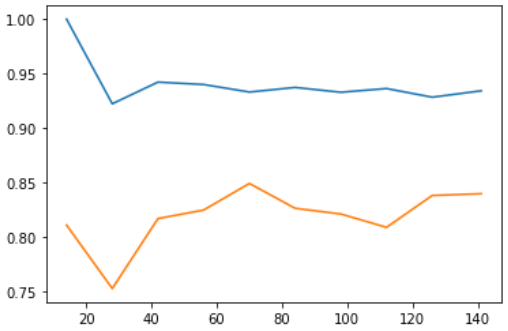


figure 10 :courbe d'apprentissage-avant optimisation

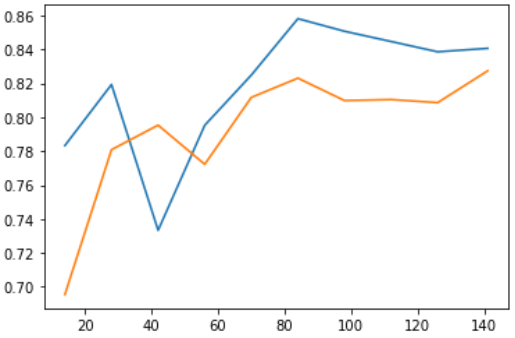


figure 11:courbe d'apprentissage-après optimisation

On peut pour les SVM essayer en amont une technique de réduction de dimension la pca qui réduit la complexité superflu des données et accélère l'apprentissage .Les résultats ne changent pas ,donc on garde le modèle sans pca , ceci peut s'expliquer par le faite que nous n'avons pas vraiment une forte colinéarité entre les variables explicatives initiales comme nous l'avons vu dans la partie 2 et ceci rend pca inefficace.
Donc le modèle SVM généralise mieux que les modèles ensemblistes et l'arbre de décision avec de meilleures performances sur F1 et recall.

C. La régression logistique :

Le dernier modèle que nous avons choisi est la régression logistique introduite pour la premiere fois en 1944 par J.Berkson et est couramment utilisée dans le domaine de l'intelligence artificielle .C'est un modèle simple et facile à interpréter , il va prédire pour chaque observation la probabilité qu'elle appartienne à la classe présence ou absence de maladie .Pour cela il utilise la fonction sigmoid qui va renvoyer une valeur entre 0 et 1 pour chaque donnée.L'algorithme d'apprentissage qu'il implémente est la descente de gradient qui a pour but de minimiser la fonction coût et nécessite qu'on normalise les données .
Plusieurs variantes de cet algorithme peuvent être utilisés suivant le jeu de donnée et cela impactera la vitesse d'apprentissage et la capacité à appréhender des grands et des petits dataset.Nous allons l'entraîner sur ses hyper-paramètres initialisés par défauts , le régularisateur C fixé à 1 ,et le solver "lbgs" qui est un algorithme des méthodes quasi-newton .Celui ci est souvent choisi pour des petits datasets ,d'autres algorithmes seront un peu plus détaillé dans la partie optimisation .
On remarque que l'estimateur généralise très mal au début (figure 12 ci-dessous) sur une petite quantité de données , puis le score sur le trainset diminue mais se rapproche de celui sur le validationset . Le modèle commence donc a mieu généraliser mais l'écart reste assez important ici aussi sûrement liée à la taille du dataset.De même comme pour les SVM d'autres valeurs pour C doivent être envisagées pour avoir un modèle plus robuste.
On va essayer différentes valeurs de paramètres pour voir si nous pouvons améliorer nos scores sur F1 et recall et diminuer un peu plus l'écart entre le trainset et validationset.
Les solvers ici seront limités à "lgbfs" et "liblinear" , ce dernier utilise la méthode de la descente de gradient par batch.

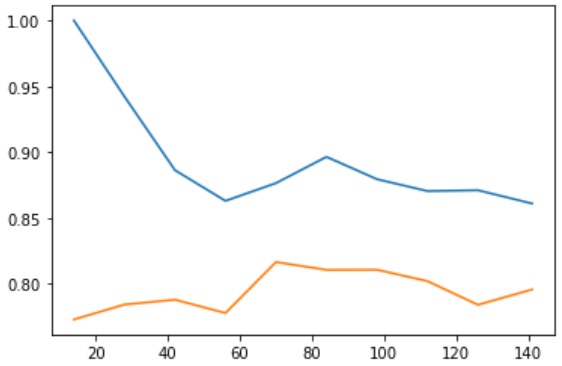


figure 12 :courbe d'apprentissage-avant optimisation

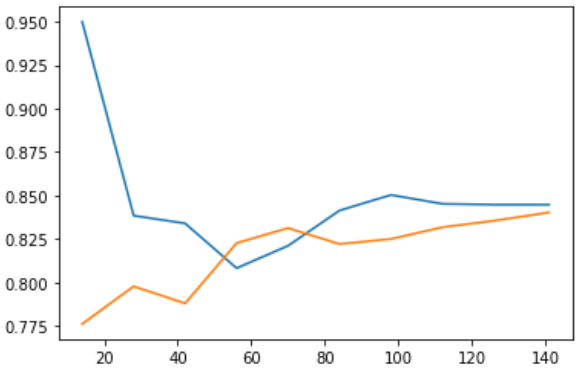


figure 13:courbe d'apprentissage-après optimisation

La régression logistique avec "liblinear" et une forte régularisation , C valant 0.006 est le modèle retourné qui donne les meilleures performances avec 84% pour F1 et recall(figure 13 ci-dessus) . Normalement "lgbfs" est plus performant sur de petit dataset mais ici ce n'est pas le cas , de même les courbes d'apprentissages semblent monter vers la fin indiquant ici aussi qu'une augmentation de données augmenterait les performances.Enfin l'écart entre les deux courbes est très petit par apport aux autres modèles , et effectuer une Pca n'améliore pas le score de F1 et recall et donc mieux vaut travailler sur nos variables initiales qui contribuent à une meilleure classification .

La capacité de ce modèle à proposer une interprétation des coefficients est précieuse et comme pour les modèles ensemblistes on peut visualiser les variables qui ont le plus contribué à la prédiction (cf figure 14 en annexes) .Notons d'abord que nous disposons d'un petit dataset donc la valeur des coefficients n'est pas assez précise, et c'est la variable "âge " qui a le plus contribué .

On peut conclure que parmis nos modèles, la régression logistique est le SVM sont ceux pour qui avec un bon paramétrage nous donnent les meilleures résultats autour de 80% sur des données qu'ils n'ont pas appris (la validation croisée) .Ils tendent à mieux généraliser et donc à avoir une plus faible variance que les autres.

D. Les courbes precision-recall:

On va donc sélectionner la régression logistique comme modèle pour notre problème de classification binaire , notre dernière étape est de finaliser sa création en observant les courbes precision-recall et en définissant notre propre seuil de décision.
On utilise la méthode precision-recall-curve de sklearn sur les données futurs du testset pour visualiser la futur précision et sensibilité en fonction du seuil qu'on choisit .En effet ce modèle fait ses prédictions en fonction du threshold qui est fixe par défaut et l'impact de celui ci peut améliorer nos performances selon l'objectif du cahier des charges .Pour Le diagnostic de maladie cardio-vasculaire , on préfère avoir une sensibilité plus importante que la précision si possible .

D'après les courbes (figure 15 ci-dessous) pour un seuil de -0.75 on a un recall de 1 qui veut dire qu'on aura su identifier toutes les personnes malades ,mais cela n'est pas pertinent il suffit d'imposer que tout le monde soit malade , de même la précision a une valeure très basse car même les personnes n'ayant rien seront diagnostiquées malades.

Donc nous devons trouver un bon compromis recall-precision et c'est le score F1 ou se croisent les courbes pour un seuil de -0.24.Avec ceci on obtient un score final sur nos données testset de 81% pour F1 et recall , nous avons donc atteint notre objectif fixe au début de ce projet avec des performances autour des 80% .

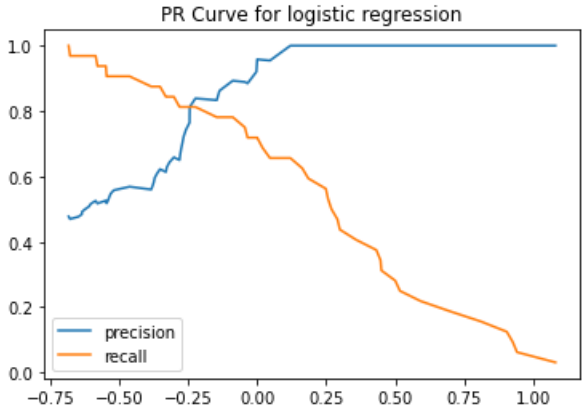


figure 15 : courbe precision-recall

Conclusion

Durant ce projet nous avons pu mettre à profit nos connaissances acquises durant l'UE Rcp209 .Nous avons vu comment des algorithmes simples et complexes de prédictions apprennent et prédisent sur de petites base de données et nous montre que des modèles comme la régression logistique peuvent donner d'aussi bon résultat voir mieux que des modèles sophistiqués commes les machines à vecteurs de supports .

Donc malgré les connaissances théoriques que nous avons sur les modèles de prédictions ,l'influence de la nature et la quantité des données peuvent donner des résultats différents de ce qu'on imaginait.

Ce projet m'aura aussi permis de mieux maîtriser les librairies d'analyse comme Panda et Seaborn mais aussi Scikit learn .Les compétences acquises seront un atout pour des futurs projets académiques ou professionnelles .

Annexes :

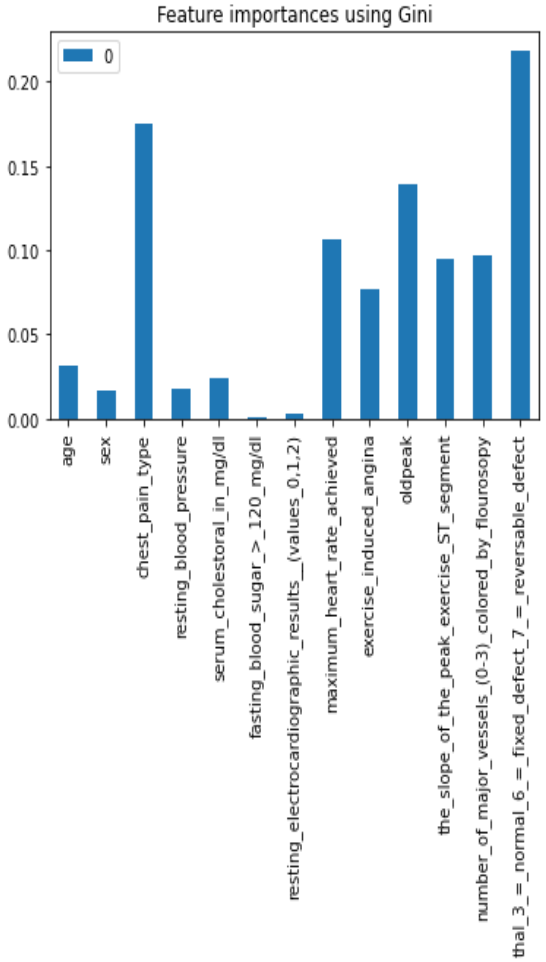


figure 6 : feature importance -random forest

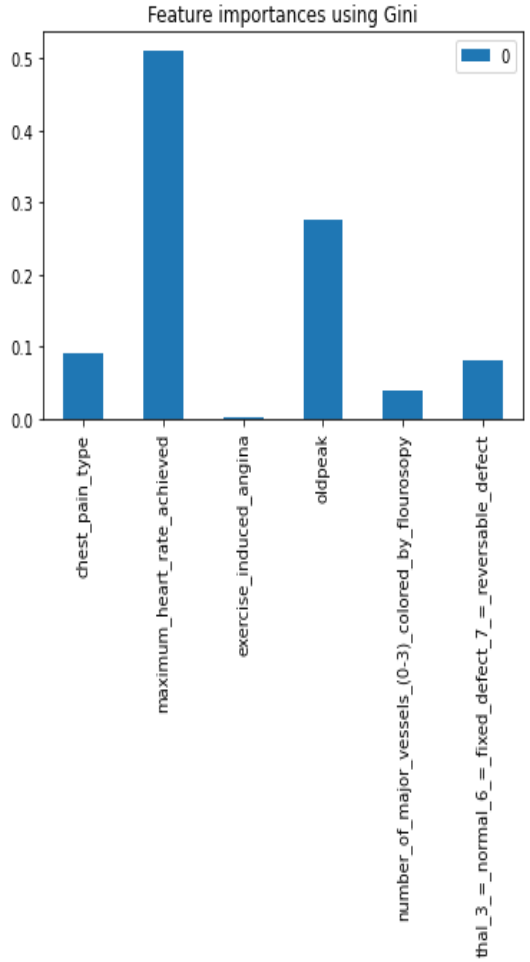


figure 9 : feature importance -adaboost

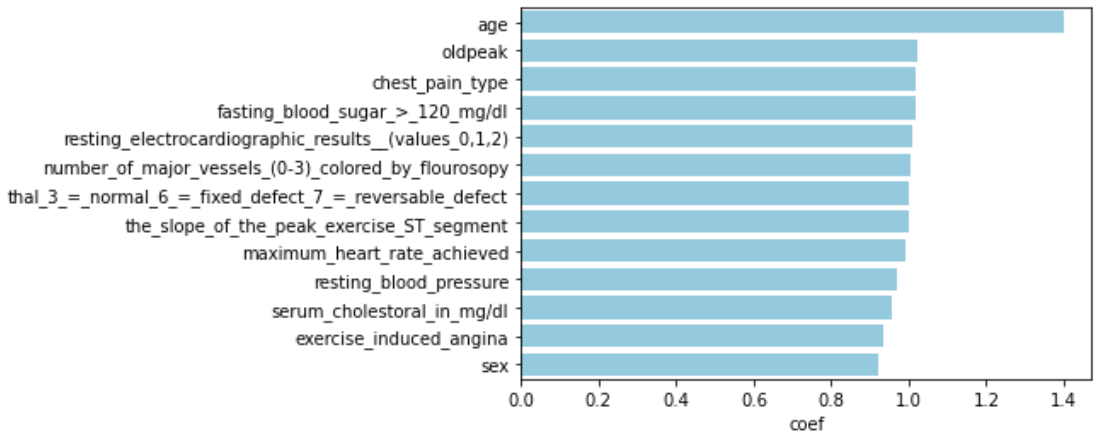


figure 14 : coefficient des features-régression logistique