REPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie
****
UNIVERSITE DE YAOUNDE I
****
ECOLE NATIONALE SUPERIEUR POLYTECHNIQUE DE YAOUNDE
****
DEPARTEMENT DU GENIE INFORMATIQUE

REPUBLIC OF CAMEROON
Peace-Work-Fatherland
****
UNIVERSITY OF YAOUNDE I
****
NATIONAL ADVANCED SCHOOL OF ENGINEERING OF YAOUNDE
****
DEPARTEMENT OF COMPUTER SCIENCE ENGINEERING

INTELLIGENCE ARTIFICIELLE ET APPLICATIONS NUMERIQUES

Rapport de Projet : Détection des maladies cardiaques

Etudiant
NGUAZONG AUREL 20P001

Sous la supervision : M. Bitha

Année académique :
2023/2024

Table des matières

INTRODUCTION

Intitulé du projet :

Les maladies cardiaques, également appelées maladies cardiovasculaires, sont un terme général utilisé pour désigner les maladies et affections affectant le cœur et le système circulatoire. C'est une cause majeure de handicap partout dans le monde. Le cœur étant l'un des organes les plus vitaux du corps, ses maladies affectent également d'autres organes et certaines parties du corps. Il existe plusieurs types et formes de maladies cardiaques. Les plus courants provoquent un rétrécissement ou un blocage des artères coronaires, un dysfonctionnement des valvules cardiaques, une hypertrophie de la taille du cœur et plusieurs autres conduisant à une insuffisance cardiaque et à une crise cardiaque.

Faits marquants selon l'OMS (Organisation Mondiale de la Santé) :
Les maladies cardiovasculaires (MCV) sont la principale cause de décès dans le monde.
On estime que 17,9 millions de personnes sont mortes de maladies cardiovasculaires en 2019, ce qui représente 32 % de tous les décès dans le monde. Parmi ces décès, 85 % étaient dus à une crise cardiaque ou à un accident vasculaire cérébral.
Plus des trois quarts des décès dus à des maladies cardiovasculaires surviennent dans des pays à revenu faible ou intermédiaire.
Sur les 17 millions de décès prématurés (moins de 70 ans) dus à des maladies non transmissibles en 2019, 38 % étaient dus à des maladies cardiovasculaires.
La plupart des maladies cardiovasculaires peuvent être évitées en s'attaquant aux facteurs de risque comportementaux tels que le tabagisme, une mauvaise alimentation et l'obésité, l'inactivité physique et la consommation nocive d'alcool.
Il est important de détecter les maladies cardiovasculaires le plus tôt possible afin que la prise en charge avec des conseils et des médicaments puisse commencer.
Le jeu de données Analyse_cardiaque comporte en son sein des données reparties dans les colonnes suivantes :

- Âge "age" : L'âge de l'individu.
- Sexe "sex" : Le sexe de l'individu.
- Type de douleur thoracique "cp" : Le type de douleur thoracique ressentie par l'individu.
- Pression artérielle au repos "trtbps" : tension artérielle au repos de l'individu (en mm Hg).
- Cholestérol "chol" : Les niveaux de cholestérol de l'individu (en mg/dL).
- Taux de sucre dans le sang à jeun "fbs" : Le taux de sucre dans le sang à jeun de l'individu (> 120 mg/dL est considéré comme élevé).
- Résultats électrocardiographiques au repos (ECG) "restecg" : Résultats de l'électrocardiogramme au repos.
- Fréquence cardiaque maximale atteinte "thalachh" : la fréquence cardiaque maximale atteinte par l'individu pendant l'exercice.
- Angine induite par l'exercice "exng" : Si la personne a souffert d'angine de poitrine pendant l'exercice.
- Dépression ST induite par l'exercice "oldpeak" : Dépression ST induite par l'exercice par rapport au repos.
- Pente du segment ST d'exercice de pointe "slp" : La pente du segment ST d'exercice de pointe.
- Nombre de vaisseaux sanguins majeurs colorés par fluoroscopie "saa" : Le nombre de vaisseaux sanguins majeurs colorés par fluoroscopie.

- Thalassémie "thall" : Un trouble sanguin ; différents types de thalassémie peuvent être représentés dans l’ensemble de données.
- Cible "Output" : Si l'individu présente le risque d'une crise cardiaque (généralement représenté sous forme binaire : 0 pour aucune crise cardiaque, 1 pour une crise cardiaque).

Vous êtes appelez à créer un modèle capable de prédire des risques d'accident cardiaque chez un patient en utilisant des méthodes d'apprentissage automatique.

I. Méthodologie de conception

1. Importation et exploitation de la dataset

La dataset importée comporte 303 lignes et 14 colonnes de base et sont diversifiées selon les types de données car toutes les colonnes n'ont pas le même type de données.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trtbps    303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  output    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Après avoir vérifié les données dupliquées et après les avoir supprimées, l'on obtient une dataset de 302 lignes et 14 colonnes.

```
#Nombre de duplication dans la dataset

data.duplicated().sum()
```

```
1
```

```
#Suppression des lignes dupliquées

data.drop_duplicates(keep='first', inplace=True)
data.duplicated().sum()
```

```
0
```

Par la suite, l'on inspecte la dataset pour vérifier les valeurs différentes de chaque colonne.

```
pd.set_option('display.max_rows', None)
data.nunique()
```

```
age          41
sex           2
cp            4
trtbps       49
chol        152
fbs           2
restecg       3
thalachh     91
exng          2
oldpeak      40
slp           3
caa           5
thall         4
output        2
dtype: int64
```

L'on constate dans ce cas que chaque colonne a sont intervalle de valeurs différentes assez disproportionnées avec l'âge qui a 41 valeurs différentes ou le taux de cholestérol qui a 152 valeurs différentes tandis que le sexe a 2 valeurs différentes. Cela conduit au tableau de description :

```
data.describe()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543	2.314
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748	0.613
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000	2.000
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152.500000	0.000000	0.800000	1.000000	0.000000	2.000
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000

N'ayant pas de minimum, maximum ou de moyenne équivalent, il va falloir réguler pour pouvoir entraîner le/les modèle(s).

2. Visualisation des données

Pour commencer l'on sépare la dataset en 2 :

- Les variables numériques : celles dont leurs valeurs sont une représentation d'une mesure numérique.

```
# descriptive statistics of the continuous variables
numeric_var = ['age', 'restecg', 'chol', 'thalachh', 'oldpeak']
```

- Les variables catégoriques : celles qui représentes une catégorie labélisée par une valeur numérique.

```
cat_var = ['sex', 'fbs', 'exng', 'output', 'cp', 'restecg', 'slp', 'caa', 'thall']
```

Par la suite l'on visualise le tout pour compter les valeurs de chaque variable.

Afin d'avoir la corrélation entre les variables et leur impact entre elles, l'on doit concevoir la matrice de corrélation.

```
correlation = data.corr()
```

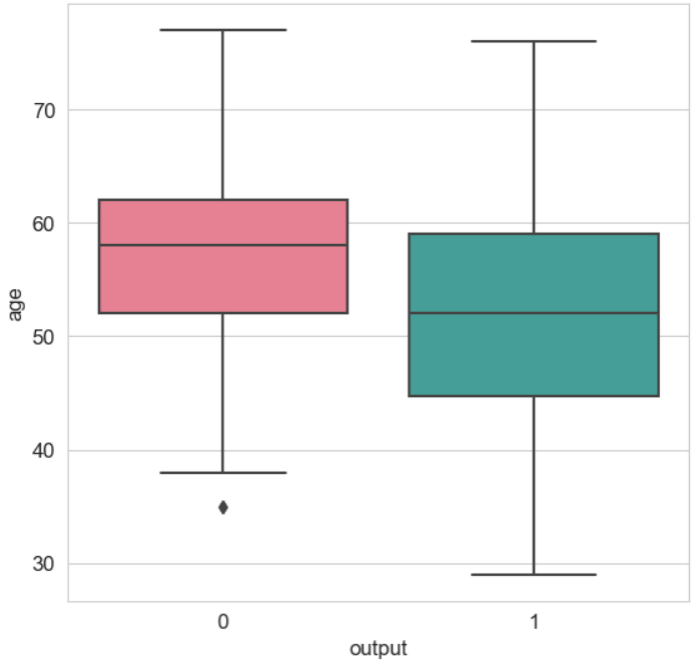
```
correlation
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
age	1.000000	-0.094962	-0.063107	0.283121	0.207216	0.119492	-0.111590	-0.395235	0.093216	0.206040	-0.164124	0.302261	0.065317	-0.221476
sex	-0.094962	1.000000	-0.051740	-0.057647	-0.195571	0.046022	-0.060351	-0.046439	0.143460	0.098322	-0.032990	0.113060	0.211452	-0.283609
cp	-0.063107	-0.051740	1.000000	0.046486	-0.072682	0.096018	0.041561	0.293367	-0.392937	-0.146692	0.116854	-0.195356	-0.160370	0.432080
trtbps	0.283121	-0.057647	0.046486	1.000000	0.125256	0.178125	-0.115367	-0.048023	0.068526	0.194600	-0.122873	0.099248	0.062870	-0.146269
chol	0.207216	-0.195571	-0.072682	0.125256	1.000000	0.011428	-0.147602	-0.005308	0.064099	0.050086	0.000417	0.086878	0.096810	-0.081437
fbs	0.119492	0.046022	0.096018	0.178125	0.011428	1.000000	-0.083081	-0.007169	0.024729	0.004514	-0.058654	0.144935	-0.032752	-0.026826
restecg	-0.111590	-0.060351	0.041561	-0.115367	-0.147602	-0.083081	1.000000	0.041210	-0.068807	-0.056251	0.090402	-0.083112	-0.010473	0.134874
thalachh	-0.395235	-0.046439	0.293367	-0.048023	-0.005308	-0.007169	0.041210	1.000000	-0.377411	-0.342201	0.384754	-0.228311	-0.094910	0.419955
exng	0.093216	0.143460	-0.392937	0.068526	0.064099	0.024729	-0.068807	-0.377411	1.000000	0.286766	-0.256106	0.125377	0.205826	-0.435601
oldpeak	0.206040	0.098322	-0.146692	0.194600	0.050086	0.004514	-0.056251	-0.342201	0.286766	1.000000	-0.576314	0.236560	0.209090	-0.429146
slp	-0.164124	-0.032990	0.116854	-0.122873	0.000417	-0.058654	0.090402	0.384754	-0.256106	-0.576314	1.000000	-0.092236	-0.103314	0.343940
caa	0.302261	0.113060	-0.195356	0.099248	0.086878	0.144935	-0.083112	-0.228311	0.125377	0.236560	-0.092236	1.000000	0.160085	-0.408992
thall	0.065317	0.211452	-0.160370	0.062870	0.096810	-0.032752	-0.010473	-0.094910	0.205826	0.209090	-0.103314	0.160085	1.000000	-0.343101
output	-0.221476	-0.283609	0.432080	-0.146269	-0.081437	-0.026826	0.134874	0.419955	-0.435601	-0.429146	0.343940	-0.408992	-0.343101	1.000000

Avec bien entendu son graphe de corrélation.

L'on remarque que les variables qui ont un fort impact sur la sortie sont (cp, thalachh, slp). Ce seront les variables à considérer en priorité.

Concernant la visualisation des catégories des variables entre elle en analyse bivariée, l'on peut prendre pour exemple la relation entre l'âge et la sortie :



Cela s'interprète par le fait que ceux qui auront tendance à être malade sont plus dans la tranche des 45 à 59 ans et ceux non malades entre environ 52 à 62 ans. Ça n'a pas d'impact significatif dans l'analyse, c'est pour cette raison que sont taux d'impact dans la matrice de confusion plus haut était de -22%.

3. Feature Processing

Cette partie est essentiellement conçu pour normaliser les données dans un même intervalle pour homogénéiser la dataset afin de ne pas avoir d'erreur dans l'entraînement.

```
# dispose of outlier (non-delete method)
# there's some outliers (illegal value) occurred in ca (ca = 4) and thal (thal = 0)
# df['ca'] == 4 -> 3; df['thal'] == 0 -> 1
data['caa'] = data['caa'].replace(4,3)
data['thall'] = data['thall'].replace(0, 1)
# due to the excessive categorical imbalance in 'restecg' category,
# I decide to merge 1 and 2 after checking the interpretation of the resting electr
# df['restecg'] = 2 -> 1
data['restecg'] = data['restecg'].replace(2,1)
# then 'restecg' would be a binary variable
```

```
# change categorical vars into objects
# numeric: 'age', 'trestbps', 'chol', 'thalach', 'oldpeak'
# binary: 'sex', 'fbs', 'exang', 'target', 'restecg' -> ordinal encoder
# multi-catagorical: 'cp', 'slope', 'ca', 'thal' -> one hot encoding
cat_var = ['sex', 'fbs', 'exng', 'output', 'cp', 'restecg', 'slp', 'caa', 'thall']
for var in cat_var:
    data[var] = data[var].astype('object')
data.info()
```

```
# for binary variables, ordinary encoder is enough
from sklearn.preprocessing import OrdinalEncoder
bin_var = ['sex', 'fbs', 'exng', 'output', 'restecg']
enc_oe = OrdinalEncoder()
for bins in bin_var:
    enc_oe.fit(data[[bins]])
    data[[bins]] = enc_oe.transform(data[[bins]])

data.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1.0	3	145	233	1.0	0.0	150	0.0	2.3	0	0	1	1.0
1	37	1.0	2	130	250	0.0	1.0	187	0.0	3.5	0	0	2	1.0
2	41	0.0	1	130	204	0.0	0.0	172	0.0	1.4	2	0	2	1.0
3	56	1.0	1	120	236	0.0	1.0	178	0.0	0.8	2	0	2	1.0
4	57	0.0	0	120	354	0.0	1.0	163	1.0	0.6	2	0	2	1.0

```
# for nulti-categorical variables, they need one-hot encoding (transform them into
from sklearn.preprocessing import OneHotEncoder
multi_cat = ['cp', 'slp', 'caa', 'thall']
def OneHotEncoding(df, enc, categories):
    transformed = pd.DataFrame(enc.transform(df[categories]).toarray(), columns=enc.get_feature_names_out(categories))
    return pd.concat([df.reset_index(drop=True), transformed], axis=1).drop(columns=categories)

enc_ohe = OneHotEncoder()
enc_ohe.fit(data[multi_cat])
data = OneHotEncoding(data, enc_ohe, multi_cat)
```

```
# standarize continuous data
from sklearn.preprocessing import StandardScaler
numeric_var
scaler = StandardScaler()
scaler.fit(data[numeric_var])

data[numeric_var] = scaler.transform(data[numeric_var])
data.head()
```

	age	sex	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	output	...	slp_0	slp_1	slp_2	caa_0	caa_1	caa_2	caa_3	thall_1	thall_2	th
0	0.949794	1.0	145	-0.261285	1.0	-1.026850	0.018826	0.0	1.084022	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	
1	-1.928548	1.0	130	0.067741	0.0	0.973852	1.636979	0.0	2.118926	1.0	...	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	
2	-1.485726	0.0	130	-0.822564	0.0	-1.026850	0.980971	0.0	0.307844	1.0	...	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	
3	0.174856	1.0	120	-0.203222	0.0	0.973852	1.243374	0.0	-0.209608	1.0	...	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	
4	0.285561	0.0	120	2.080602	0.0	0.973852	0.587366	1.0	-0.382092	1.0	...	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	

5 rows × 24 columns

4. Séparation des données test et données d'entraînement

L'on sépare les données selon le ratio 8 : 2.

```
from sklearn import model_selection

y = data['output']
x = data.drop('output', axis=1)

x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.2, stratify=y)

print('Le jeu de données d\'entraînement contient ' + str(x_train.shape[0]) + ' observations avec ' + str(x_train.shape[1]) + ' v
print('Le jeu de données de test contient ' + str(x_test.shape[0]) + ' observations avec ' + str(x_test.shape[1]) + ' variables.'
```

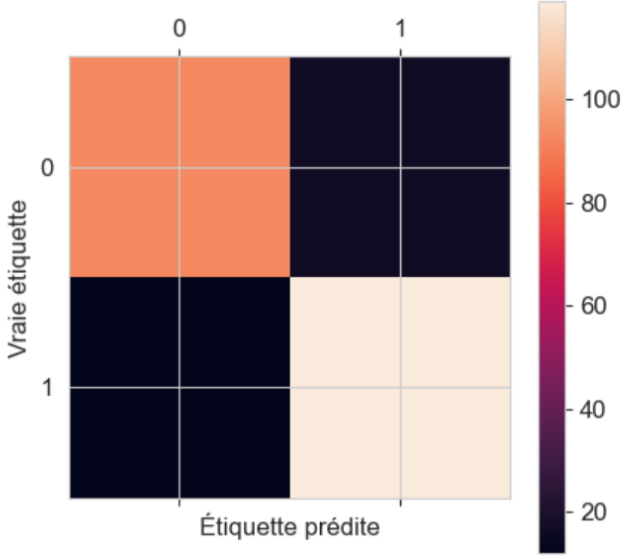
```
Le jeu de données d'entraînement contient 241 observations avec 23 variables.
Le jeu de données de test contient 61 observations avec 23 variables.
```

5. Modèle d'entrainement

Concernant ce travail, l'on a testé plusieurs modèles différents pour essayer d'en tirer le meilleur possible. Dans ce cas, les résultats par modèles sont les suivant :

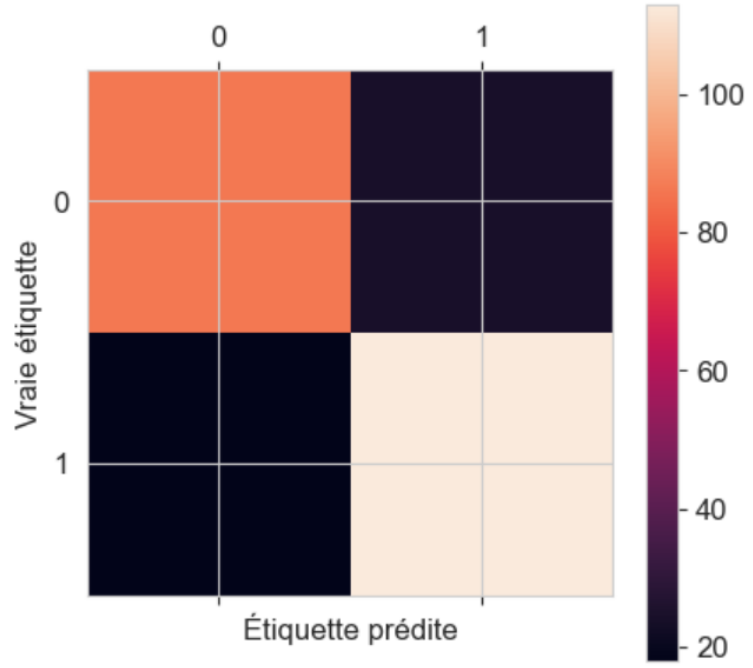
- Régression logistique :

```
L'exactitude du classifieur logistique est de 87.967%
Pour le classifieur logistique, l'exactitude est de 84.65% (75.89% ~ 93.41%)
```



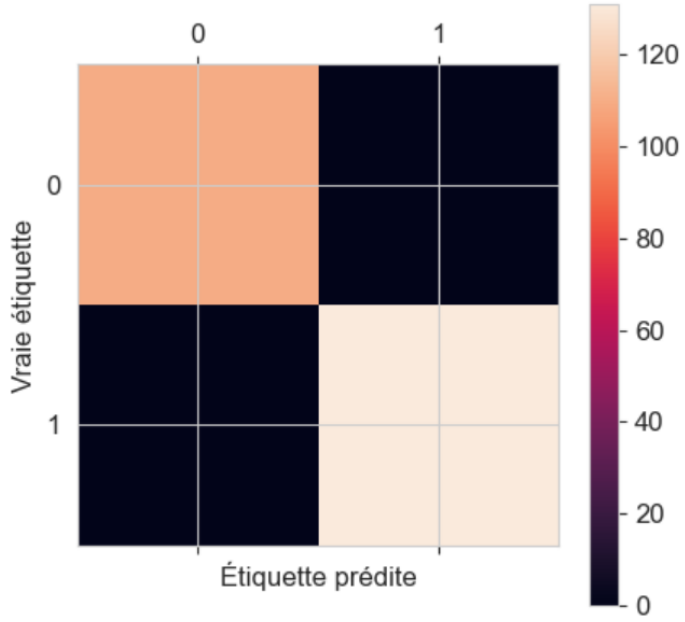
- Classification KNN

```
Pour KNN, l'exactitude est de 75.92% (62.76% ~ 89.07%)
```



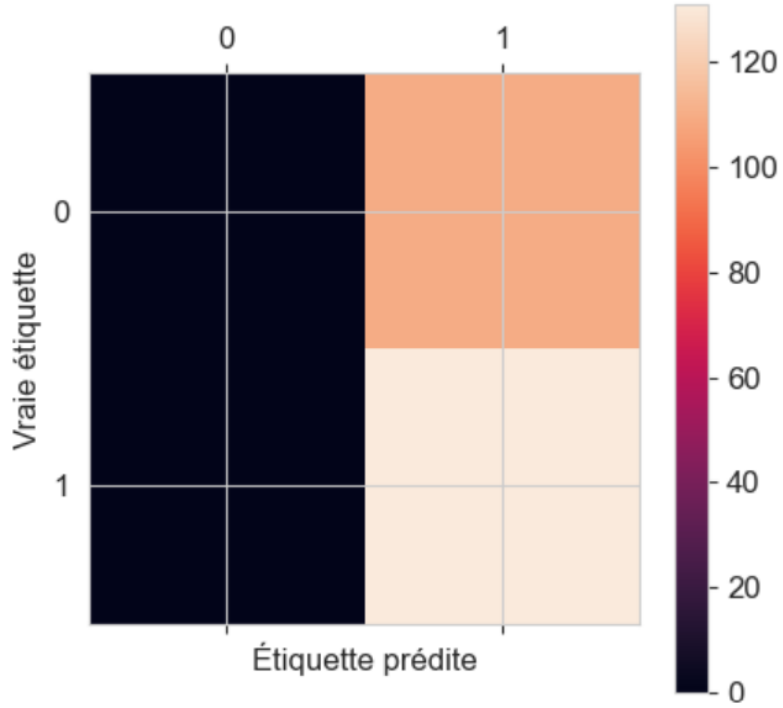
- Ramdom Forest

```
Pour Random Forest, l'exactitude est de 85.9% (79.45% ~ 92.35%)
```



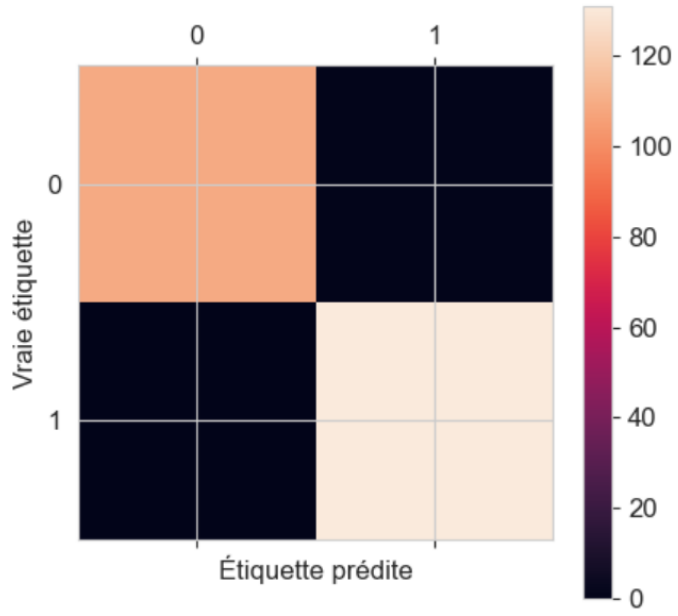
- Classification SVC

```
Pour SVC, l'exactitude est de 54.35% (53.27% ~ 55.43%)
```



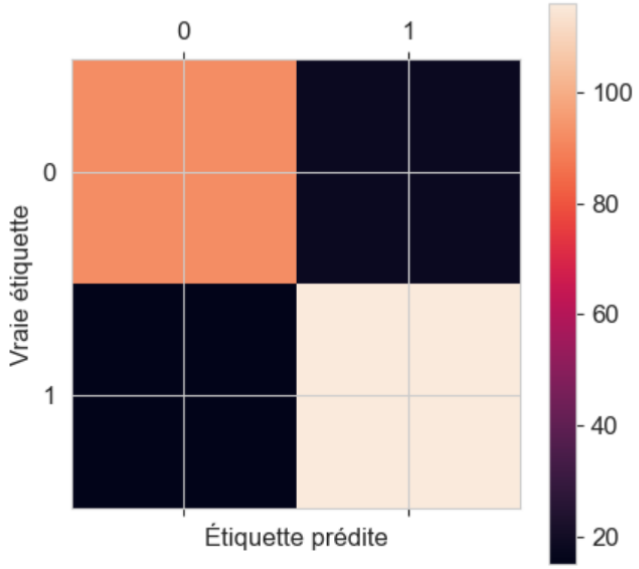
- Classification GB

```
Pour GB Classifier, l'exactitude est de 81.73% (66.12% ~ 97.35%)
```



- Classifieur Naive Baiye

```
Pour le classifieur Naive Bayes, l'exactitude est de 83.0% (74.6% ~ 91.4%)
```



L’on remarque pour ce cas que chaque modèle présente des caractéristiques différentes à un point où chacun présente des précisions plus ou moins importante mais l’on retient que le meilleur modèle est celui de la régression logistique avec 87,967%.

II. Remarque et commentaire

Avec comme modèle prépondérant celui de la régression logistique, l'on conclut que celui-ci est le meilleur modèle pour l'entraînement. L'on pourrait optimiser les hyperparamètres pour arriver à d'autres précision mais cela ne ferait que nuire au performance si n'est pas bien maîtiser.