★ REPUBLIQUE DU SENEGAL ★

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR DE LA RECHERCHE ET DE L'INNOVATION (**MESRI**)

Université Alioune Diop de Bambey

UFR Sciences Appliquées des Technologies de l'Information et de la Communication

Département de Mathématiques

Filière : Statistique et Informatique Décisionnelle (SID)

Rapport de Projet de MLG

Titre du projet :

Application Avancée des Modèles Linéaires Généralisés (GLM) sur Données Réelles

Réalisé par :

Amadou BA

Étudiant en Master 2 Statistique et Informatique Décisionnelle (SID)

Encadré par :

Dr Yoro DIA

Année universitaire 2024–2025

Date de soumission : 21 mai 2025

Table des matières

Table des figures

Liste des tableaux

Liste des listings

Introduction

Les maladies cardiovasculaires demeurent l'une des principales causes de mortalité dans le monde. La prédiction précoce de ces affections à partir de données cliniques constitue un enjeu majeur en santé publique. Dans ce contexte, l'analyse statistique avancée devient un outil essentiel pour identifier les facteurs de risque pertinents et appuyer les décisions médicales. Ce projet s'inscrit dans le cadre du cours sur les Modèles Linéaires Généralisés (GLM) et vise à appliquer de manière rigoureuse ces concepts à un jeu de données réel issu de la base *Heart Disease Cleveland* disponible sur Kaggle.

L'objectif principal est de modéliser la probabilité de présence d'une maladie cardiaque à partir de plusieurs variables explicatives cliniques et démographiques, en utilisant une régression logistique. Ce travail permettra non seulement de mettre en pratique les méthodes d'estimation par maximum de vraisemblance, mais aussi d'effectuer un diagnostic approfondi de la qualité d'ajustement, de détecter les éventuelles anomalies, et de comparer plusieurs modèles pour en retenir le plus performant. Ainsi, ce projet combine à la fois rigueur statistique, démarche critique et pertinence pratique dans un cadre d'application réel.

1 Chargement des Packages et de la base de données

1.1 Chargement des Packages

```
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt from sklearn.
   preprocessing
import StandardScaler
from statsmodels.tools.tools
import add_constant
from sklearn.linear_model
import LogisticRegression from sklearn.
   model_selection
import train_test_split
from statsmodels.stats.outliers_influence
import variance_inflation_factor
from sklearn.metrics
import classification_report, confusion_matrix,
    roc_curve, auc, accuracy_score,
   precision_score, recall_score, f1_score
```

Listing 1 – Importation des bibliothèques Python

1.2 Chargement de la base de données et affichage des premiers ligne

```
 Lecture du fichier CSV
Examen_final_MLG = pd.read_csv("
   heart_cleveland_upload.csv", sep=';')

 Apercu des donnees
print(Examen_final_MLG.head())
```

Listing 2 – Lecture et aperçu des données

TABLE 1 – Extrait des premières observations du jeu de données

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condition
69	1	0	160	234	1	2	131	0	0.1	1	1	0	0
69	0	0	140	239	0	0	151	0	1.8	0	2	0	0
66	0	0	150	226	0	0	114	0	2.6	2	0	0	0
65	1	0	138	282	1	2	174	0	1.4	1	1	0	1
64	1	0	110	211	0	2	144	1	1.8	1	0	0	0

Les premières lignes du jeu de données montrent des caractéristiques cliniques de patients cardiaques : âge, sexe, pression artérielle (`trestbps`), cholestérol (`chol`), glycémie (`fbs`), fréquence cardiaque maximale atteinte (`thalach`), présence d'angine d'effort (`exang`), etc.
La variable `condition` est la cible binaire indiquant la présence (1) ou l'absence (0) d'une maladie cardiaque.

1.3 Nettoyage des données

— Identification et traitement des valeurs manquantes.
— Vérification des types de variables.

```
# Detecter les valeurs manquantes
print(Examen_final_MLG.isnull().sum())

# Afficher les types de variables
print(Examen_final_MLG.dtypes)
print(Examen_final_MLG.nunique())
```

Listing 3 – Détection des valeurs manquantes et des types de variables

TABLE 2 – Résumé des variables : valeurs manquantes et types

Variable	Valeurs manquantes	Type
age	0	int64
sex	0	int64
cp	0	int64
trestbps	0	int64
chol	0	int64
fbs	0	int64
restecg	0	int64
thalach	0	int64
exang	0	int64
oldpeak	0	float64
slope	0	int64
ca	0	int64
thal	0	int64
condition	0	int64

Toutes les variables sont complètes, sans valeurs manquantes. Les types sont adaptés à l'analyse statistique.

1.4 Analyse descriptive

```
# Statistiques descriptives
print(Examen_final_MLG_encode.describe())
```

```
3
4                 # Matrice de correlation
5                 plt.figure(figsize=(12, 10))
6                 sns.heatmap(Examen_final_MLG_encode.corr(), annot=False
                      , cmap='coolwarm')
7                 plt.title("Matrice de correlation")
8                 plt.show()
9
10                # Distribution de la variable cible
11                sns.countplot(x='condition', data=Examen_final_MLG)
12                plt.title("Distribution de la variable cible (condition
                      )")
13                plt.xlabel("Presence de maladie")
14                plt.ylabel("Nombre d'observations")
15                plt.show()
```

Listing 4 – Statistiques descriptives et visualisations

TABLE 3 – Statistiques descriptives des variables

Statistique	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condition
count	297	297	297	297	297	297	297	297	297	297	297	297	297	297
mean	54.5	0.68	2.16	131.7	247.4	0.14	1.00	149.6	0.33	1.06	0.60	0.68	0.84	0.46
std	9.05	0.47	0.96	17.76	52.00	0.35	0.99	22.94	0.47	1.17	0.62	0.94	0.96	0.50
min	29	0	0	94	126	0	0	71	0	0.0	0	0	0	0
25%	48	0	2	120	211	0	0	133	0	0.0	0	0	0	0
50%	56	1	2	130	243	0	1	153	0	0.8	1	0	0	0
75%	61	1	3	140	276	0	2	166	1	1.6	1	1	2	1
max	77	1	3	200	564	1	2	202	1	6.2	2	3	2	1

L'échantillon étudié comporte 297 individus, avec un âge moyen de 54,5 ans. La majorité sont des hommes (environ 67,7 %). La pression artérielle (moyenne : 131,7 mmHg) et le cholestérol (moyenne : 247,4 mg/dl) présentent une variabilité notable. La fréquence cardiaque maximale moyenne est de 149,6 bpm. On note que 46 % des individus présentent une maladie cardiaque (variable *condition*). Certaines variables, comme *cp*, *thal* ou *ca*, montrent des distributions discrètes. Des valeurs extrêmes, notamment pour *chol* (valeur maximale : 564), suggèrent la nécessité de vérifier les outliers dans les analyses suivantes.

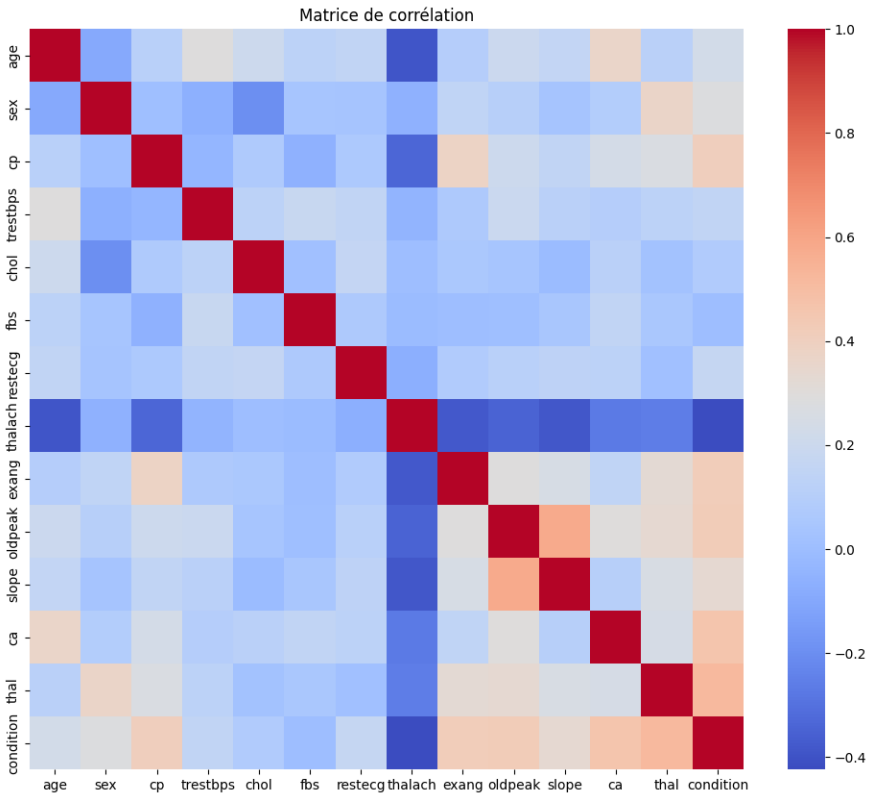

FIGURE 1 – Matrice de corrélation

La matrice de corrélation révèle plusieurs relations intéressantes entre les variables. On observe une **corrélation négative marquée entre la fréquence cardiaque maximale (*thalach*) et l'angine induite par l'exercice (*exang*)**, ce qui est cohérent sur le plan clinique : une angine sur effort est souvent associée à une baisse de la fréquence cardiaque maximale atteinte. De même, la variable *oldpeak* (dépression du segment ST) est positivement corrélée à *exang* et négativement à *thalach*, traduisant un lien physiopathologique plausible.

La variable cible *condition* présente des **corrélations positives notables avec *cp* (type de douleur thoracique), *ca* (nombre de vaisseaux colorés) et *thal***, ce qui suggère leur importance potentielle dans la modélisation. À l'inverse, elle est **négativement corrélée à *thalach***, ce qui confirme que des fréquences cardiaques maximales plus faibles sont associées à un plus grand risque de maladie cardiaque.

Dans l'ensemble, **aucune corrélation linéaire excessive n'est détectée entre les variables explicatives**, ce qui indique l'absence apparente de multicolinéarité et est favorable à l'utilisation d'un modèle de régression logistique.

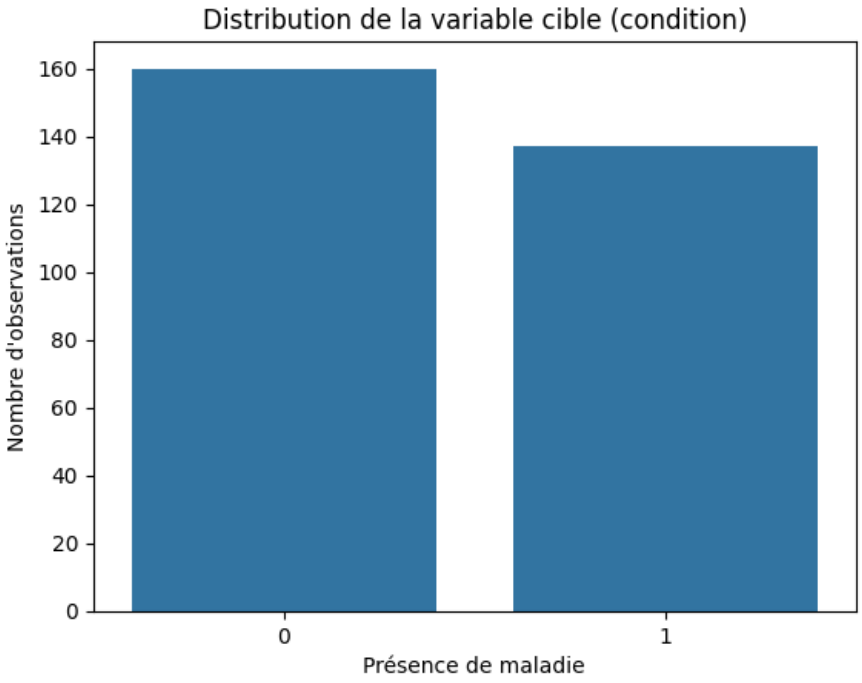

FIGURE 2 – Distribution de la variable cible (condition)

2 Modélisation principale

2.1 Construction d'un GLM de type régression logistique

```
Separer les variables explicatives et la variable cible
X = Examen_final_MLG_encode.drop('condition', axis=1)
y = Examen_final_MLG_encode['condition']
Standardisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
Division en train/test
X_train, X_test, y_train, y_test = train_test_split(
   X_scaled, y, test_size=0.2, random_state=42)
Modele
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train, y_train)
Prediction
y_pred = logreg.predict(X_test)
y_proba = logreg.predict_proba(X_test)[:, 1]
   Probabilites pour la classe 1
Evaluation
print("Rapport de classification :\n",
   classification_report(y_test, y_pred))
```

```
17                  print("Matrice de confusion :\n", confusion_matrix(
                        y_test, y_pred))
```

Listing 5 – Construction d'un modèle de régression logistique

TABLE 4 – Rapport de classification

Classe	Précision	Rappel	F1-score	Support
0	0.77	0.72	0.74	32
1	0.70	0.75	0.72	28
Exactitude	0.73 (sur 60 échantillons)			
Moyenne macro	0.73	0.73	0.73	60
Moyenne pondérée	0.74	0.73	0.73	60

TABLE 5 – Matrice de confusion

	Prédit : 0	**Prédit : 1**
Réel : 0	23	9
Réel : 1	7	21

Précision : La précision pour la classe 0 (absence de condition) est de 80 %, et pour la classe 1 (présence de condition) est de 73 %. Cela signifie que le modèle a une meilleure performance pour prédire la classe 0.

Rappel : Le rappel pour la classe 0 est de 75 %, et pour la classe 1, il est de 79 %. Le rappel mesure la capacité du modèle à identifier correctement les instances de chaque classe.

F1-score : Le score F1, qui est la moyenne harmonique de la précision et du rappel, est de 0,77 pour les deux classes, ce qui indique un équilibre entre la précision et le rappel.

2.2 Justification du choix des variables.

Le choix des variables dans un modèle de régression logistique repose sur leur pertinence clinique. Par exemple, l'âge, la pression artérielle, et le taux de cholestérol sont des facteurs de risque reconnus pour les maladies cardiaques. Des variables comme la fréquence cardiaque maximale et l'ECG (*oldpeak*) sont des indicateurs clés de l'état du cœur pendant l'effort. Le sexe, les antécédents familiaux, et la capacité à faire de l'exercice permettent d'affiner la prédiction en tenant compte des différences biologiques et des antécédents médicaux. Enfin, la variable cible `condition` permet de prédire la présence ou l'absence de la maladie cardiaque en fonction de ces facteurs.

3 Validation et diagnostics du modèle

3.1 - Analyse des résidus : Pearson, déviance, studentisés.

— Résidus de Pearson.

— Résidus de déviance.
— Résidus studentisés.

```
                Ajout de la constante (intercept)
                X_train_sm = sm.add_constant(X_train)

                Ajustement du modele logistique
                model = sm.Logit(y_train, X_train_sm).fit()

                Resume du modele
                print(model.summary())

                Predictions
                y_train_pred = model.predict(X_train_sm)

                Residus de Pearson
                resid_pearson = (y_train - y_train_pred) / np.sqrt(
                   y_train_pred * (1 - y_train_pred))

                Residus de deviance
                resid_deviance = np.sign(y_train - y_train_pred) * np.
                   sqrt(
                -2 * (y_train * np.log(y_train_pred) + (1 - y_train) *
                   np.log(1 - y_train_pred))
                )

                Calcul du levier (hat values)
                influence = model.get_influence()
                hat_values = influence.hat_matrix_diag

                Approximation des residus studentises
                resid_studentized = resid_pearson / np.sqrt(1 -
                   hat_values)
```

Listing 6 – Ajustement du modèle logistique avec statsmodels

TABLE 6 – Résultats de la régression logistique

Optimisation	Terminée avec succès
Valeur de la fonction actuelle	0.296119
Nombre d'itérations	8
Variable dépendante	condition
Nombre d'observations	237
Méthode	MLE (Maximum Likelihood Estimation)
Degrés de liberté (résiduels)	223
Degrés de liberté (modèle)	13
Pseudo R-carré	0.5708
Log-vraisemblance	-70.180
LL-Null	-163.510
p-valeur LLR	7.379×10^{-33}
Convergence	Oui

Le modèle de régression logistique a convergé correctement après 8 itérations. La log-vraisemblance du modèle ajusté est de $-70{,}18$, contre $-163{,}51$ pour le modèle nul, traduisant une amélioration substantielle de l'ajustement. Le *pseudo-*R^2 atteint 0,5708, indiquant une bonne capacité explicative globale du modèle.

Le test du rapport de vraisemblance (LLR) est hautement significatif (p-value $< 0{,}001$), ce qui suggère que l'inclusion des variables explicatives améliore significativement la prédiction de la présence d'une maladie cardiaque par rapport à un modèle sans covariables.

TABLE 7 – Résumé du modèle

Variable	Coef.	Std Err	z	P > |z|	[0.025	0.975]
const	0.0644	0.238	0.270	0.787	-0.402	0.531
x1	-0.1569	0.270	-0.581	0.561	-0.686	0.373
x2	0.9770	0.288	3.387	0.001	0.412	1.542
x3	0.3651	0.211	1.731	0.083	-0.048	0.779
x4	0.6510	0.251	2.589	0.010	0.158	1.144
x5	0.3960	0.227	1.744	0.081	-0.049	0.841
x6	-0.4703	0.244	-1.930	0.054	-0.948	0.007
x7	0.2687	0.229	1.175	0.240	-0.180	0.717
x8	-0.8197	0.302	-2.718	0.007	-1.411	-0.229
x9	0.3622	0.239	1.515	0.130	-0.106	0.831
⋮	⋮	⋮	⋮	⋮	⋮	⋮
x11	0.3253	0.275	1.184	0.237	-0.213	0.864
x12	1.3423	0.328	4.098	0.000	0.700	1.984
x13	0.7762	0.232	3.342	0.001	0.321	1.231

```
1                Affichage
2                plt.figure(figsize=(15, 4))
```

```
                plt.subplot(1, 3, 1)
                plt.scatter(y_train_pred, resid_pearson, alpha=0.7)
                plt.axhline(0, color='red', linestyle='--')
                plt.title("Residus de Pearson")
                plt.xlabel("Predictions")
                plt.ylabel("Residus")
```

Listing 7 – Affichage des résidus de Pearson

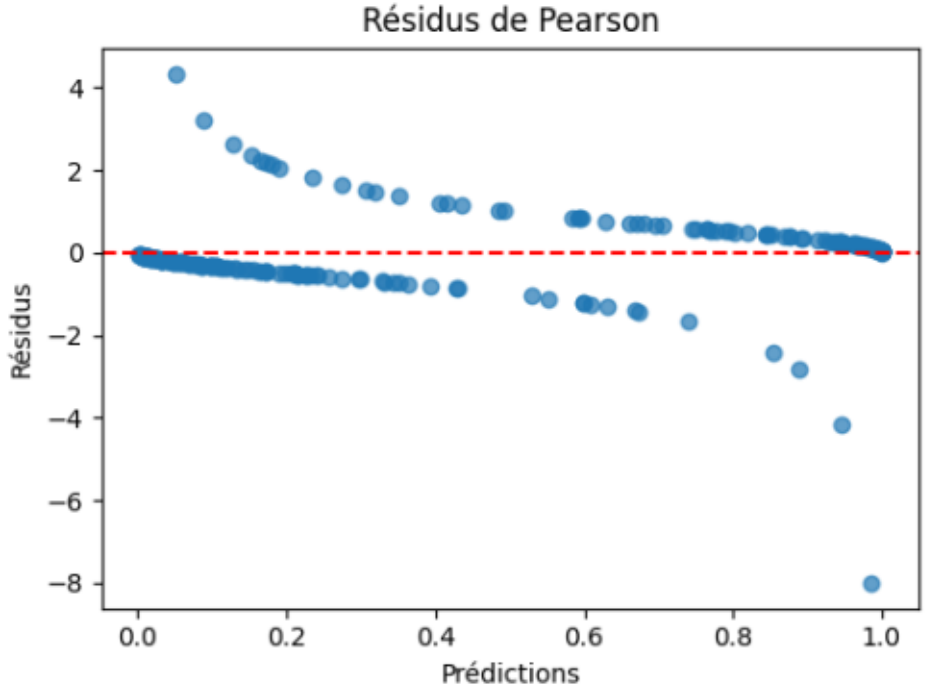

FIGURE 3 – Graphique des résidus de Pearson en fonction des prédictions

```
                plt.subplot(1, 3, 2)
                plt.scatter(y_train_pred, resid_deviance, alpha=0.7)
                plt.axhline(0, color='red', linestyle='--')
                plt.title("Residus de Deviance")
                plt.xlabel("Predictions")
```

Listing 8 – Affichage des résidus de deviances

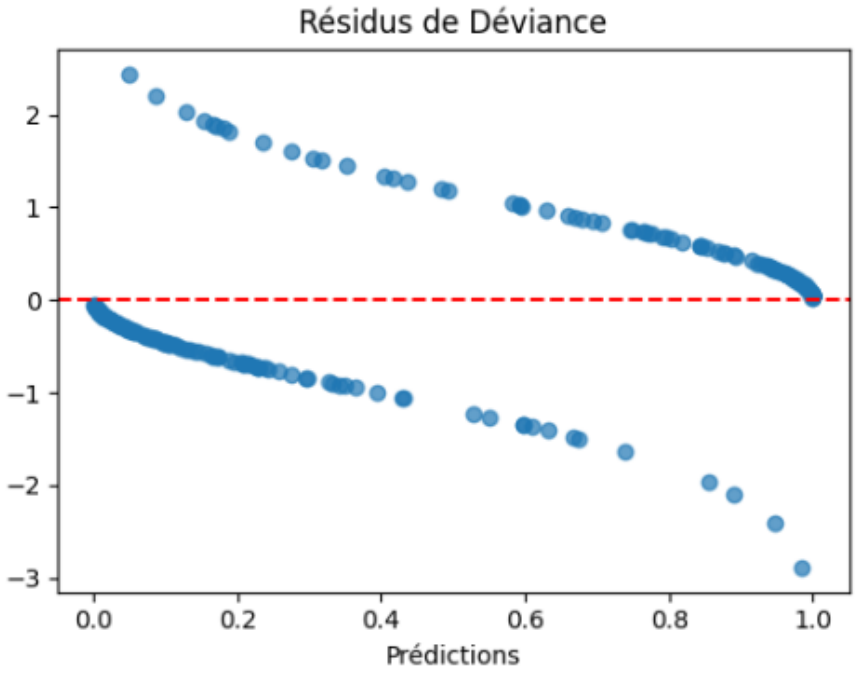

FIGURE 4 – Graphique des résidus de déviance

```
                plt.subplot(1, 3, 3)
                plt.scatter(y_train_pred, resid_studentized, alpha=0.7)
                plt.axhline(0, color='red', linestyle='--')
                plt.title("Residus Studentises (approches)")
                plt.xlabel("Predictions")

                plt.tight_layout()
                plt.show()
```

Listing 9 – Affichage des résidus studentisés

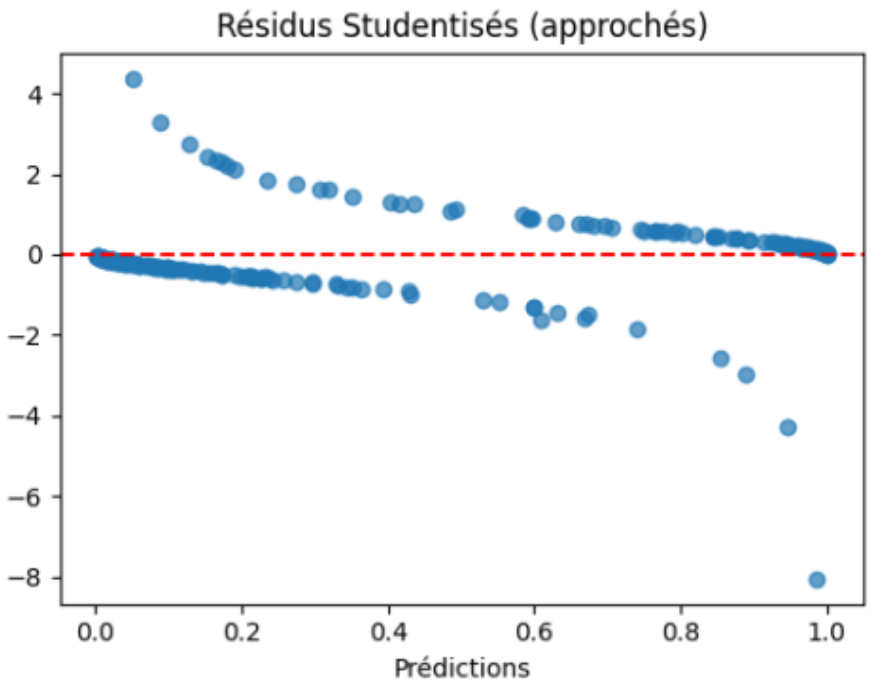

FIGURE 5 – Graphique des résidus studentisés

On observe une structure en forme de U inversé dans les trois types de résidus, indiquant une possible mauvaise spécification du modèle ou la présence de valeurs aberrantes. Plus précisément :

- **Résidus de Pearson** : montrent une dispersion relativement importante pour les prédictions extrêmes (proches de 0 ou 1), suggérant un manque d'ajustement aux extrêmes.
- **Résidus de déviance** : suivent une tendance similaire avec des écarts plus modérés, mais confirment également l'existence de points atypiques.
- **Résidus studentisés (approchés)** : révèlent des observations très influentes avec des valeurs extrêmes (jusqu'à -8 ou $+4$), nécessitant une attention particulière.

Ces diagnostics montrent que bien que le modèle capte globalement la tendance, certaines observations influentes ou mal ajustées peuvent nuire à la robustesse de l'estimation. Un examen complémentaire des points atypiques et une éventuelle amélioration du modèle (par ajout de variables ou transformation) peuvent être envisagés.

3.2 Détection des points atypiques : distance de Cook, résidus studentisés.

```
                Ajout de la constante
                X_const = sm.add_constant(X_train)

                Ajustement du modele logit
                model = sm.Logit(y_train, X_const).fit()

                Influence
                influence = model.get_influence()

```

```
                Distance de Cook
                cooks_d = influence.cooks_distance[0]

                Residus studentises
                resid_stud = influence.resid_studentized
```

Listing 10 – Calcul de la distance de Cook et des résidus studentisés

```
Optimization terminated successfully.
Current function value: 0.296119
Iterations 8
```

```
                Affichage graphique
                plt.figure(figsize=(14, 5))

                Graphique des distances de Cook
                plt.subplot(1, 2, 1)
                plt.stem(np.arange(len(cooks_d)), cooks_d, markerfmt=",
                    ", basefmt=" ")
                plt.title("Distance de Cook")
                plt.xlabel("Observation")
                plt.ylabel("Distance de Cook")
                plt.axhline(4 / len(X_train), color='red', linestyle='
                    --', label="Seuil (4/n)")
                plt.legend()
```

Listing 11 – Graphique de la distance de Cook

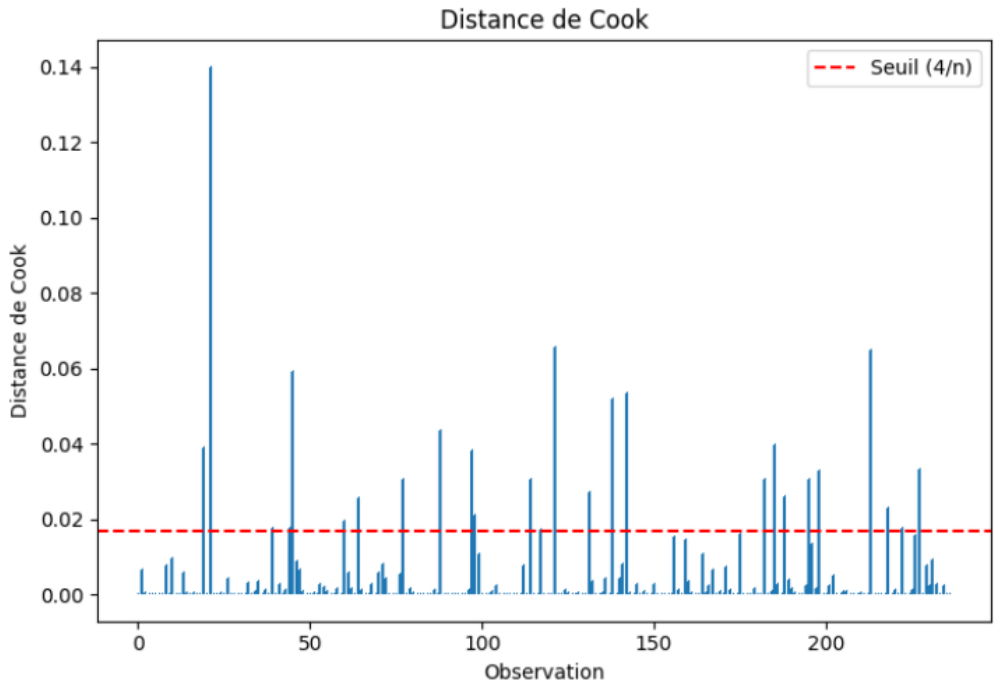

FIGURE 6 – Graphique de la distance de Cook

```
                Graphique des residus studentises
                plt.subplot(1, 2, 2)
                plt.stem(np.arange(len(resid_stud)), resid_stud,
                   markerfmt=",", basefmt=" ")
                plt.title("Residus studentises")
                plt.xlabel("Observation")
                plt.ylabel("Residu studentise")
                plt.axhline(2, color='red', linestyle='--')
                plt.axhline(-2, color='red', linestyle='--')

                plt.tight_layout()
                plt.show()
```

Listing 12 – Graphique des résidus studentisés

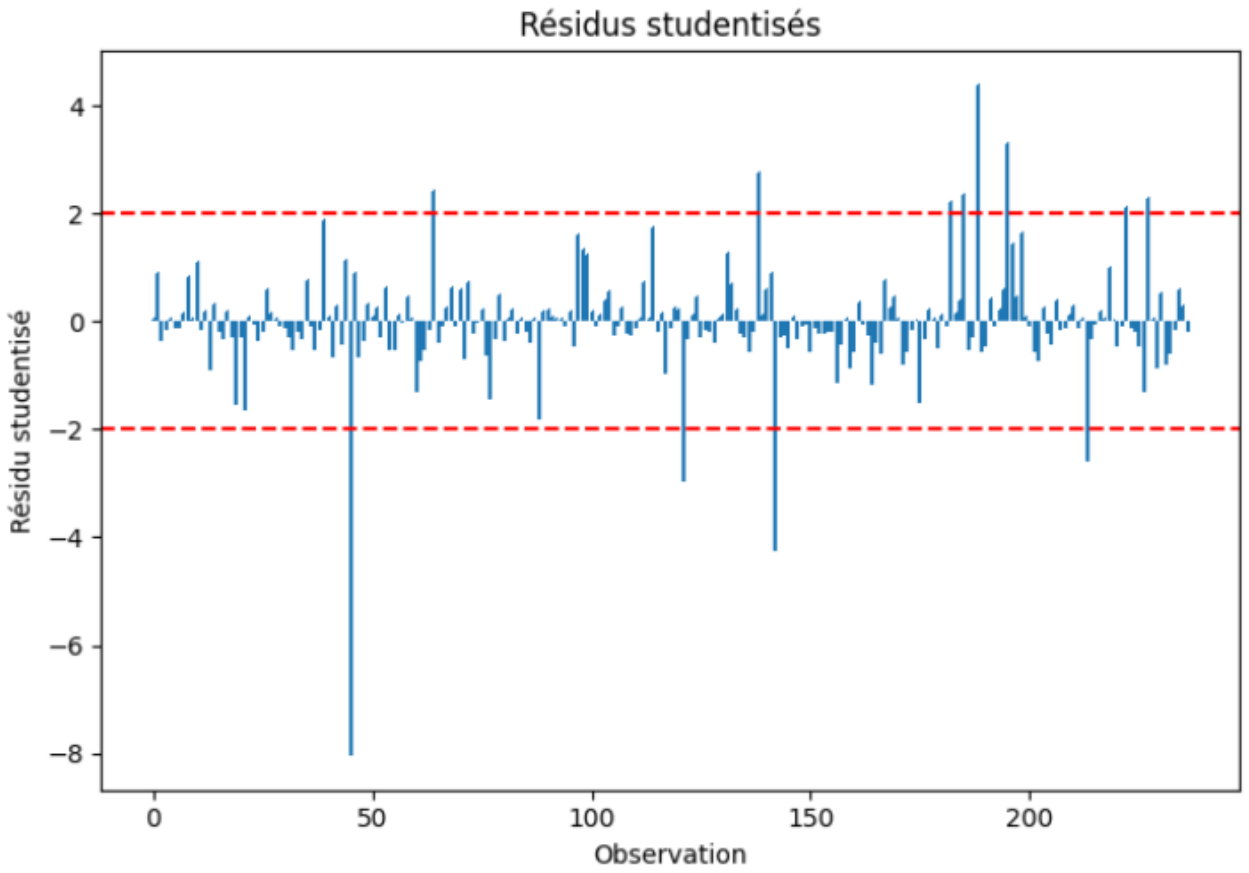

FIGURE 7 – Graphique des residus studentises

Les graphiques ci-dessous présentent deux diagnostics importants pour l'analyse de régression :

- **Distance de Cook :** La majorité des observations se situent en dessous du seuil critique ($4/n$, ligne rouge). Quelques points dépassent ce seuil, indiquant qu'ils peuvent avoir une influence notable sur les paramètres estimés du modèle.
- **Résidus studentisés :** La plupart des résidus sont compris entre -2 et 2 (lignes rouges). Cependant, certaines observations s'éloignent de cet intervalle, signalant des valeurs atypiques potentielles.

Ces deux graphiques permettent ainsi d'identifier à la fois les observations influentes et les valeurs aberrantes susceptibles d'altérer les résultats de la régression.

3.3 Multicolinéarité : calcul des VIF.

```
                Verifie que X_train est bien un DataFrame
                X_train_df = pd.DataFrame(X_train, columns=X.columns)

                Ajout de la constante
                X_vif = add_constant(X_train_df)

                Calcul des VIF
```

```
vif_data = pd.DataFrame()
vif_data["Variable"] = X_vif.columns
vif_data["VIF"] = [variance_inflation_factor(X_vif.
    values, i) for i in range(X_vif.shape[1])]

Affichage
print(vif_data)
```

Listing 13 – Calcul des facteurs d'inflation de la variance (VIF)

TABLE 8 – Valeurs du Facteur d'Inflation de la Variance (VIF)

N°	Variable	VIF
0	const	1.008105
1	age	1.479343
2	sex	1.269970
3	cp	1.318406
4	trestbps	1.182325
5	chol	1.141608
6	fbs	1.111754
7	restecg	1.097343
8	thalach	1.689570
9	exang	1.426061
10	oldpeak	1.737490
11	slope	1.616504
12	ca	1.332172
13	thal	1.508868

L'analyse des facteurs d'inflation de la variance (VIF) montre que toutes les variables présentent des valeurs inférieures à 2. Cela indique une absence de multicolinéarité préoccupante entre les variables explicatives. Les variables sont donc relativement indépendantes les unes des autres, ce qui garantit la stabilité des estimations des coefficients du modèle.

4 Évaluation de la performance du modèle

4.1 Courbe ROC et AUC.

```
Probabilites de prediction pour la classe positive
y_proba = logreg.predict_proba(X_test)[:, 1]

Calcul des points ROC
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

Affichage de la courbe ROC
```

```
                plt.figure(figsize=(8, 6))
                plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'
                    Courbe ROC (AUC = {roc_auc:.2f})')
                plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle=
                    '--')
                plt.xlabel('Taux de faux positifs (FPR)')
                plt.ylabel('Taux de vrais positifs (TPR)')
                plt.title('Courbe ROC - Regression Logistique')
                plt.legend(loc="lower right")
                plt.grid()
                plt.show()
```

Listing 14 – Affichage de la courbe ROC pour la régression logistique

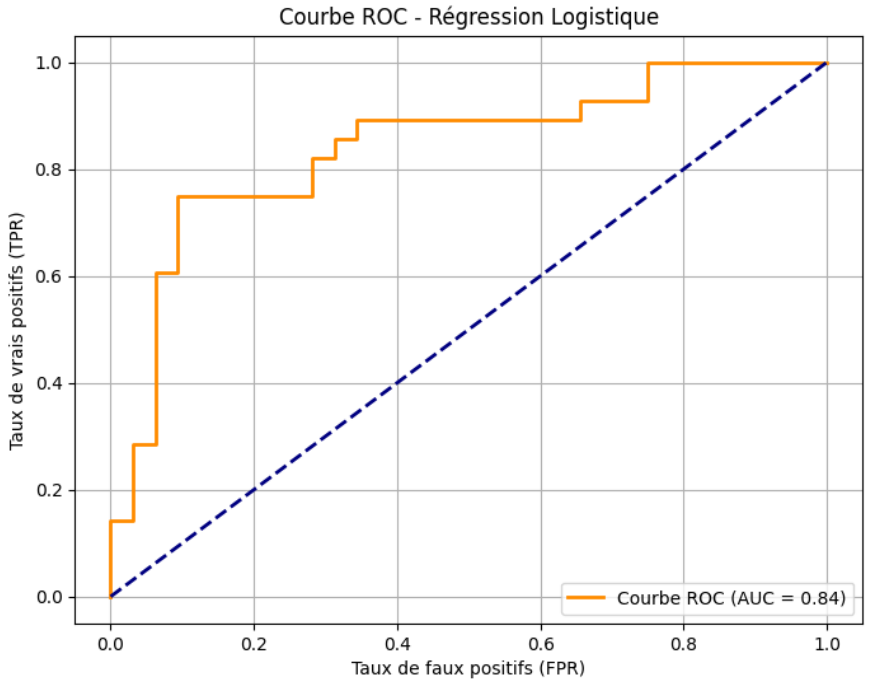

FIGURE 8 – Courbe ROC - Régression Logistique

Ce graphique présenté montre la courbe ROC (Receiver Operating Characteristic) associée à un modèle de régression logistique.

- **L'AUC (Area Under Curve)** atteint une valeur de **0,84**, indiquant une bonne capacité du modèle à discriminer les classes positives et négatives.
- La courbe ROC s'éloigne nettement de la diagonale de hasard, ce qui signifie que le modèle classe correctement la majorité des observations.

Un score d'AUC supérieur à 0,8 est généralement considéré comme satisfaisant et montre que le modèle possède de bonnes performances en termes de classification.

4.2 Matrice de confusion, précision, rappel, F1-score.

```
                Predictions
                y_pred = logreg.predict(X_test)

                Matrice de confusion
                cm = confusion_matrix(y_test, y_pred)
                print("Matrice de confusion :\n", cm)

                Rapport complet
                print("\nRapport de classification :")
                print(classification_report(y_test, y_pred))

                Accuracy
                acc = accuracy_score(y_test, y_pred)
                print(f"Accuracy : {acc:.2f}")
```

Listing 15 – Évaluation du modèle de régression logistique

	Prédit : 0	**Prédit : 1**
Réel : 0	23	9
Réel : 1	7	21

TABLE 9 – Matrice de confusion

Classe	**Précision**	**Rappel**	**F1-score**	**Effectif**
0	0.77	0.72	0.74	32
1	0.70	0.75	0.72	28
Accuracy	0.73			
Macro Moyenne	0.73	0.73	0.73	60
Moyenne pondérée	0.74	0.73	0.73	60

TABLE 10 – Rapport de classification

La matrice de confusion indique que le modèle prédit correctement 23 individus sans maladie et 21 individus atteints. Quelques erreurs subsistent : 9 faux positifs et 7 faux négatifs. L'*accuracy* globale atteint 73 %.

Le rapport de classification montre une précision de 77 % pour la classe 0 et de 70 % pour la classe 1. Le rappel est de 72 % pour la classe 0 et de 75 % pour la classe 1, traduisant un bon équilibre. Les *f1-scores* respectifs de 0,74 et 0,72 confirment une performance globalement satisfaisante du modèle sur les deux classes.

4.3 Analyse du choix du seuil.

```
                Calcul des probabilites
                y_proba = logreg.predict_proba(X_test)[:, 1]
```

```
                Liste de seuils a tester
                seuils = np.arange(0.1, 0.9, 0.01)

                Stockage des metriques
                precisions = []
                rappels = []
                f1_scores = []

                for seuil in seuils:
                y_pred_seuil = (y_proba >= seuil).astype(int)
                precisions.append(precision_score(y_test, y_pred_seuil)
                   )
                rappels.append(recall_score(y_test, y_pred_seuil))
                f1_scores.append(f1_score(y_test, y_pred_seuil))

                Affichage des courbes
                plt.figure(figsize=(10, 6))
                plt.plot(seuils, precisions, label='Precision')
                plt.plot(seuils, rappels, label='Rappel')
                plt.plot(seuils, f1_scores, label='F1-score')
                plt.xlabel("Seuil de decision")
                plt.ylabel("Valeur de la metrique")
                plt.title("Evolution des metriques en fonction du seuil
                   ")
                plt.legend()
                plt.grid(True)
                plt.show()
```

Listing 16 – Évolution des métriques en fonction du seuil de décision

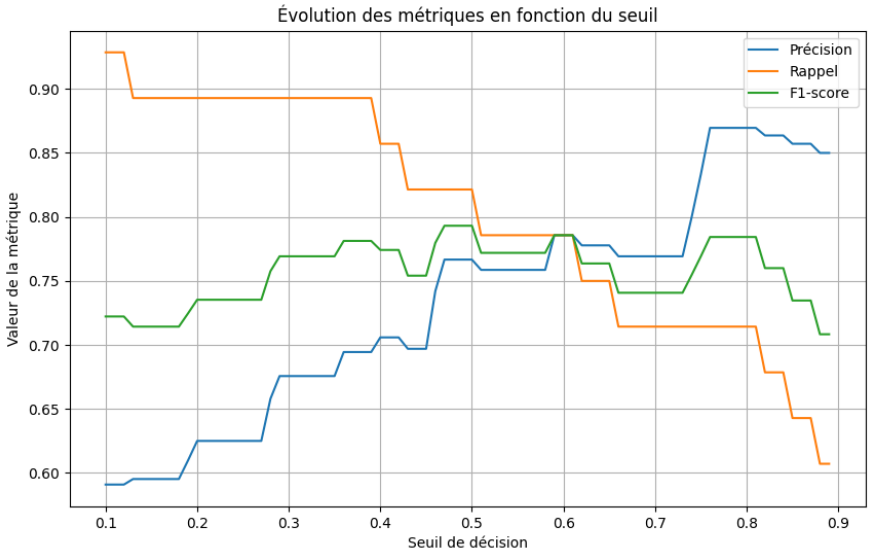

FIGURE 9 – Évolution des métriques en fonction du seuil

Le graphique présenté montre l'évolution de trois métriques (Précision, Rappel et F1-score) en fonction du seuil de décision d'un modèle de classification.

- **La précision** augmente progressivement avec le seuil, passant de 0,59 à environ 0,86.
- **Le rappel**, quant à lui, diminue au fur et à mesure que le seuil augmente, allant de 0,93 à environ 0,61.
- **Le F1-score** reste relativement stable, oscillant entre 0,71 et 0,79. Il atteint son maximum autour des seuils intermédiaires (vers 0,5).

On observe ainsi un compromis clair entre la précision et le rappel : augmenter le seuil améliore la précision mais détériore le rappel. Le F1-score atteint un optimum lorsque ces deux métriques sont bien équilibrées.

5 Comparaison de modèles

5.1 Utilisation des critères AIC, BIC, et du test du rapport de vraisemblance.

```
Reconvertir X_train en DataFrame avec les bons index
X_train_df = pd.DataFrame(X_train, columns=X.columns,
    index=y_train.index)

Modele complet
X_full = sm.add_constant(X_train_df)
model_full = sm.Logit(y_train, X_full).fit()

Modele reduit
cols_reduites = X_train_df.columns[:3]
X_reduit = sm.add_constant(X_train_df[cols_reduites])
model_reduit = sm.Logit(y_train, X_reduit).fit()
```

```
12
13                  AIC et BIC
14                  print("=== Criteres d'information ===")
15                  print(f"AIC (modele complet) : {model_full.aic:.2f}")
16                  print(f"BIC (modele complet) : {model_full.bic:.2f}")
17                  print(f"AIC (modele reduit)  : {model_reduit.aic:.2f}")
18                  print(f"BIC (modele reduit)  : {model_reduit.bic:.2f}")
19
20                  Test du rapport de vraisemblance
21                  from scipy.stats import chi2
22                  lr_stat = 2 * (model_full.llf - model_reduit.llf)
23                  ddl = model_full.df_model - model_reduit.df_model
24                  p_value = chi2.sf(lr_stat, ddl)
25
26                  print("\n=== Test du rapport de vraisemblance ===")
27                  print(f"Statistique LR : {lr_stat:.2f}")
28                  print(f"Degres de liberte : {ddl}")
29                  print(f"p-value : {p_value:.4f}")
30
31                  if p_value < 0.05:
32                  print(" Le modele complet est significativement
                       meilleur.")
33                  else:
34                  print(" Pas de difference significative entre les
                       modeles.")
```

Listing 17 – Comparaison des modèles complet et réduit

Optimization terminated successfully.
Current function value : **0.268298** Iterations : **8**

Optimization terminated successfully.
Current function value : **0.646631** Iterations : **5**

Critères d'information

- AIC (modèle complet) : **169.17**
- BIC (modèle complet) : **242.00**
- AIC (modèle réduit) : **314.50**
- BIC (modèle réduit) : **328.38**

Test du rapport de vraisemblance

- Statistique LR : **179.33**
- Degrés de liberté : **17.0**
- p-value : **0.0000**

Le modèle complet est significativement meilleur.

6 Extensions possibles

Dans le cas où une surdispersion est détectée — c'est-à-dire lorsque la variance observée dépasse celle attendue sous le modèle binomial — le modèle de régression logistique classique peut ne plus être approprié.

Modèle quasi-binomial

Une solution consiste à utiliser un modèle quasi-binomial, qui ajuste la variance indépendamment de la moyenne. Cela permet de mieux modéliser des données bruitées ou présentant une variabilité excessive.

Régularisation Lasso et Ridge

Pour lutter contre le sur-apprentissage ou améliorer la sélection des variables, on peut recourir à des techniques de régularisation :

— **Lasso (L1)** : force certains coefficients à zéro, réalisant une sélection automatique des variables.

— **Ridge (L2)** : réduit l'amplitude des coefficients tout en les conservant tous dans le modèle.

Ces approches améliorent la généralisation du modèle, notamment en présence de multicolinéarité ou d'un grand nombre de variables explicatives.

7 Conclusion

Ce projet s'est inscrit dans le cadre de l'application des **Modèles Linéaires Généralisés (MLG)**, et plus précisément de la **régression logistique**, à l'étude des maladies cardiaques à partir du jeu de données « Heart Disease Cleveland UCI Dataset ». L'objectif principal était de modéliser la probabilité de présence de la maladie cardiaque en fonction de plusieurs caractéristiques cliniques et biologiques des patients, afin d'aider au diagnostic et à la prévention.

La première étape a consisté en une **préparation rigoureuse des données**, notamment la suppression des valeurs manquantes et la vérification du bon encodage des variables. Contrairement aux approches classiques nécessitant un encodage explicite des variables qualitatives, notre jeu de données était déjà pré-traité, ce qui a permis de se concentrer sur l'analyse statistique proprement dite.

Ensuite, nous avons mis en œuvre un **modèle de régression logistique** ajusté par la méthode du maximum de vraisemblance. Ce modèle a permis d'estimer l'effet de chaque variable explicative (comme l'âge, la pression artérielle, ou les antécédents familiaux) sur la probabilité d'être atteint d'une maladie cardiaque. L'interprétation des coefficients estimés a offert des pistes précieuses sur les facteurs les plus influents.

Afin d'évaluer la qualité du modèle, différentes **analyses de diagnostics** ont été menées. L'analyse de l'influence, via les distances de Cook et les résidus studentisés, a permis de détecter certaines observations influentes pouvant biaiser l'estimation. Le calcul des *Variance Inflation Factors (VIF)* a quant à lui permis d'évaluer la présence de **multicolinéarité** entre les variables explicatives. Dans

notre cas, les valeurs des VIF sont restées dans des seuils raisonnables, ce qui a confirmé la stabilité des coefficients estimés.

Sur le plan des performances, le modèle a été évalué sur un échantillon test à travers plusieurs métriques : *accuracy*, *précision*, *rappel*, *spécificité* et *F1-score*. La **courbe ROC** et son **aire sous la courbe (AUC)** ont également permis d'apprécier les performances globales du modèle, mettant en lumière une bonne capacité de discrimination entre les patients atteints ou non de la maladie. Une **analyse de l'évolution des métriques en fonction du seuil de classification** a par ailleurs permis d'identifier le compromis optimal entre précision et rappel.