**Rapport de prétraitement des données  
UCI Heart Disease**

Pipeline Scikit-learn (imputation, encodage, mise à l’échelle, modèle)

# 1. Contexte et objectif

L’objectif est de transformer les données brutes du jeu UCI Heart Disease en un format exploitable pour un modèle de classification. Le flux couvre : gestion des valeurs manquantes, encodage des variables catégorielles, mise à l’échelle des variables numériques et automatisation du processus via un Pipeline.

# 2. Données

Variables utilisées : numériques (age, trestbps, chol, thalch, oldpeak, ca) et catégorielles (sex, dataset, cp, fbs, restecg, exang, slope, thal). La cible est la variable num (niveau de maladie).

# 3. Méthodologie de prétraitement (justifications)

## 3.1 Imputation

• Numériques : imputation par la médiane. Elle est plus robuste aux valeurs extrêmes (outliers) qu’une moyenne.  
• Catégorielles : imputation par la valeur la plus fréquente (mode), cohérente avec des modalités discrètes.

## 3.2 Encodage des catégorielles

• slope : encodage ordinal avec l’ordre (downsloping < flat < upsloping), car la variable possède une hiérarchie.  
• Autres catégorielles : One-Hot Encoding avec drop='first' pour éviter la colinéarité (dummy trap) et handle\_unknown='ignore' pour rester robuste si une modalité apparaît en test mais pas en entraînement.

## 3.3 Mise à l’échelle

Deux variantes sont comparées :  
• StandardScaler : centrage-réduction (moyenne 0, écart-type 1).  
• RobustScaler : basé sur la médiane et l’IQR, plus stable en présence d’outliers (ex. cholestérol).

## 3.4 Transformations complémentaires (analyse)

• Discrétisation (KBinsDiscretizer) : conversion d’une variable continue en intervalles pour analyser des profils et simplifier certains signaux.  
• PowerTransformer (Yeo-Johnson) : réduction de l’asymétrie (skewness) sur oldpeak.  
• PolynomialFeatures (degré 2) sur un sous-ensemble de variables (ex. age, trestbps, chol) pour capturer des relations non linéaires, tout en limitant l’explosion combinatoire.

# 4. Pipeline complet (ColumnTransformer + make\_pipeline)

Le prétraitement est encapsulé dans un ColumnTransformer :  
• Bloc numérique : SimpleImputer(median) + (StandardScaler ou RobustScaler)  
• Bloc slope : SimpleImputer(most\_frequent) + OrdinalEncoder  
• Bloc nominal : SimpleImputer(most\_frequent) + OneHotEncoder  
Le tout est enchaîné à un modèle LogisticRegression dans un Pipeline unique, ce qui garantit un flux reproductible et évite les fuites d’information (les transformations sont apprises uniquement sur le train).

Le découpage train/test utilise stratify=y pour conserver la proportion des classes dans les deux ensembles, ce qui rend l’évaluation plus fiable lorsque la distribution des classes est déséquilibrée.

# 5. Résultats et interprétation

Précision (accuracy) obtenue :  
• Pipeline StandardScaler : 0.598  
• Pipeline RobustScaler : 0.603

Interprétation (bref) : le pipeline avec RobustScaler donne une légère amélioration, ce qui est cohérent si certaines variables numériques contiennent des valeurs atypiques. Les performances restent modestes : une analyse du déséquilibre de classes, des métriques complémentaires (F1, recall) et/ou des modèles plus flexibles pourraient être envisagés pour améliorer la détection.

# 6. Conclusion

Le pipeline final automatise l’ensemble du prétraitement et la classification dans un objet unique, cohérent avec les étapes précédentes (imputation médiane/mode, encodage ordinal + one-hot, mise à l’échelle). La comparaison Standard vs Robust suggère un avantage léger du scaling robuste, utile en présence d’outliers.

# Annexe – Infographie récapitulative

