BRAC
UNIVERSITY

Inspiring Excellence

CSE422: Artificial Intelligence
Project Report
Project Title: Prediction of Heart Disease

Group No : _13_, CSE422 Lab Section : _04_, Summer 2025

ID	Name
23101314	Aditi Roy Adri
23101302	Shiva Prasad Sarkar

Table of Contents

1. Introduction

This project builds and compares multiple ML models to predict heart disease from tabular patient data. We implement a full pipeline: EDA → preprocessing → stratified split → model training (Neural Network ,KNN,Naive Bayes, Logistic Regression)→ evaluation → unsupervised K-Means exploration → reporting. The report is structured per the course template.

2. Dataset Description

2.1 Features & Target

- **Rows:** 920
- **Columns:** 16
- **Non-predictive columns :** Id , dataset , ca (ids are unique and so keeping it will add noise and will overfit ; values in the dataset are mostly constant categorical values and have no variance ; And ca column has maximum values missing so drooping it is the best option.).
- **Target:** Num with values {0,1,2,3,4}.
- **Frequencies:** 0: 411, 1: 265, 2: 109, 3: 107, 4: 28
- **Feature count used:** 20 (after dropping id,dataset,ca and treating num as target).

```
Shape of dataset: (920, 16)
```

The problem type is classification.The target column is **num** with **integer, discrete** values **{0, 1, 2, 3, 4}** and the following class frequencies:
0: 411, **1: 265**, **2: 109**, **3: 107**, **4: 28**. These values are encoded in categories, rather than continuous values.

Feature Types:

Quantitative: ['age', 'chol', 'oldpeak', 'thalch', 'trestbps']

Categorical (binary): ['exang', 'fbs', 'sex']

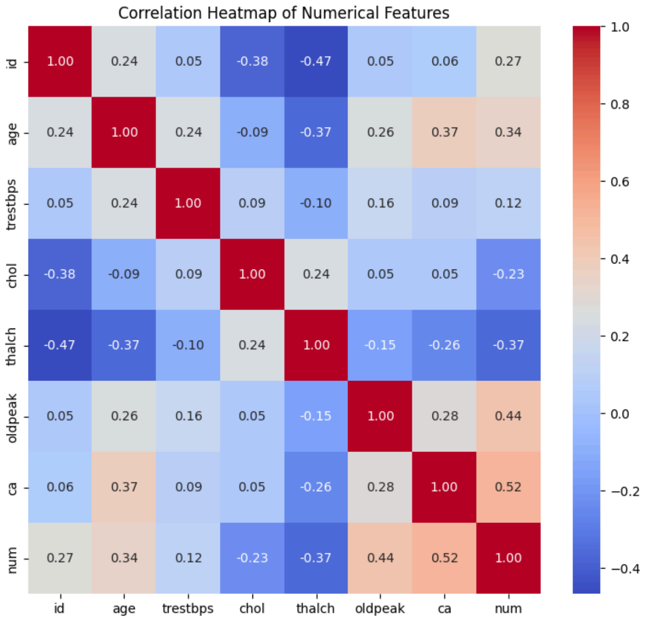
Categorical (multi-level): ['ca', 'cp', 'restecg', 'slope', 'thal']

Yes. Most ML algorithms require numeric input, so categorical variables must be encoded.

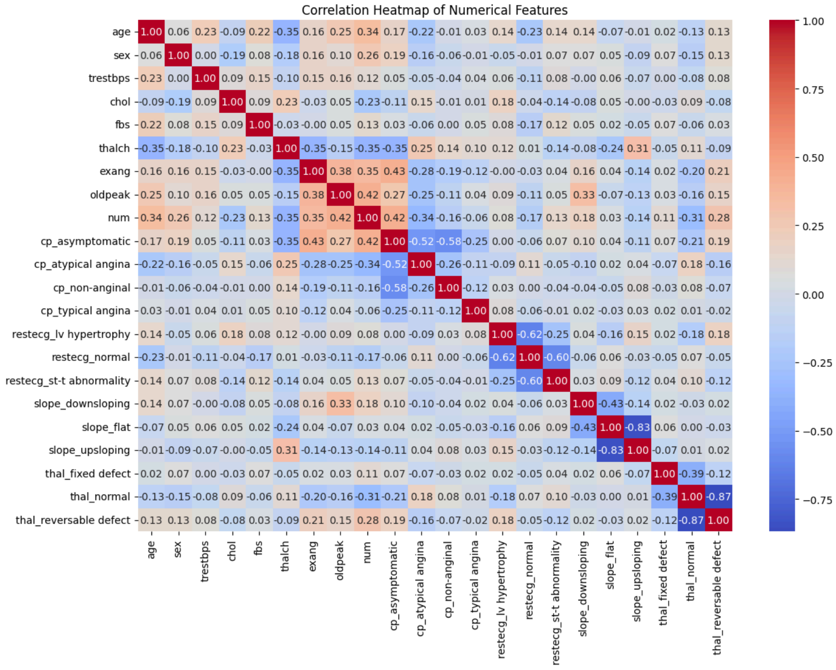
- **Binary example:** sex (Male/Female). If left as text, models cannot process it. Encoding turns it into sex_Male = {0,1}, allowing the model to learn that being male increases heart disease risk.

- **Multi-category example:** (chest pain type: typical, atypical, non-anginal, asymptomatic). If kept as {0,1,2,3}, the model may assume 3 > 1. With One-Hot Encoding, it becomes four columns (cp_typical, cp_atypical, etc.), each a yes/no (1/0) indicator, so no false ordering is imposed.

- The result of the Co-relation before data preprocessing :



After preprocessing and encoding :



Insights from Heatmap: Here from the heatmap we see that both 'slope_flat', 'slope_upsloping has strong correlation , so we can exclude one of the features . The same goes for ‘thal reversible effect’ and ‘thal normal’ both have high correlation , so can also exclude one of these features.

2.2 Imbalanced Dataset

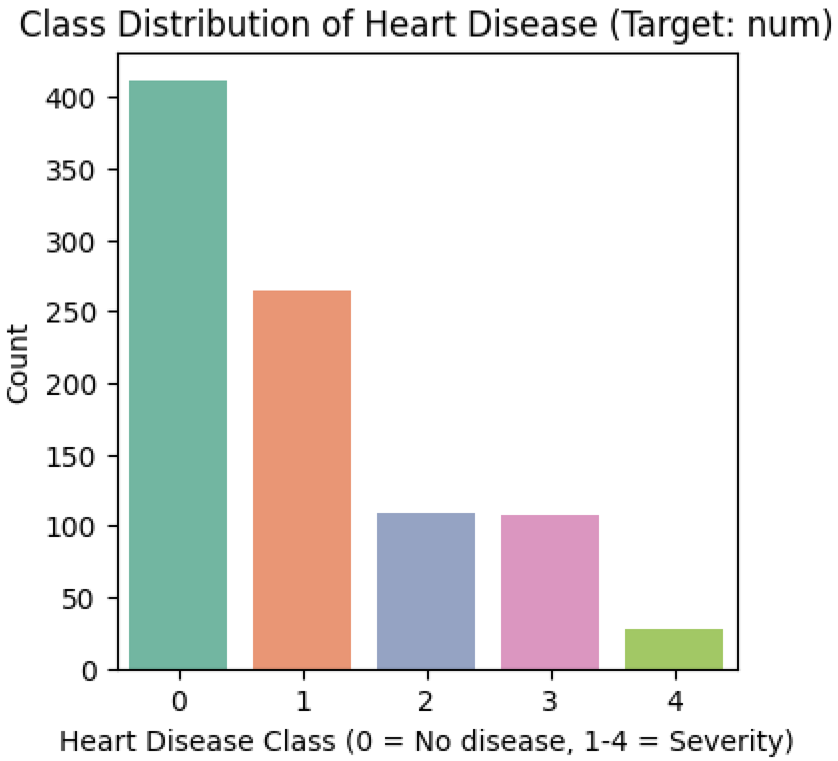
- **Multiclass:** Highly skewed (class 4 rare with 28 samples).

- **Binary (num>0):** Positive ≈ 509 vs negative 411 (mild imbalance).

The target variable num in the heart disease dataset takes five unique values {0, 1, 2, 3, 4}, representing increasing severity of disease. The class distribution is highly unequal: the majority class (0, no disease) contains 411 samples, while the rarest class (4) contains only 28 samples. This imbalance means that certain severity levels are **underrepresented**.

Imbalanced datasets are problematic because most machine learning algorithms assume relatively balanced class distributions. When imbalance is present, models tend to be **biased toward majority classes**, achieving high accuracy by overpredicting common categories while performing poorly on minority ones. In our case, without handling imbalance, a model might predict “no disease” for most patients and still achieve a deceptively high accuracy, while failing to detect higher severity levels that are clinically critical.

This imbalance justifies simplifying the problem to **binary classification (disease vs. no disease)**, where the distribution is more balanced (411 vs. 509).



2.3: Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA), specifically focusing on:

- **Identifying important relationships** within the data.

- **Imbalanced Dataset** analysis: Checking if any classes have an unequal distribution.
- **Correlation analysis** using visual tools like heatmaps.

Key Insights for EDA :

- **Identifying Relationships**:
  - EDA should uncover relationships between the features (e.g., age, cholesterol, resting blood pressure) and the target variable (num for heart disease)
  - This involves looking at distributions, pairwise relationships, and patterns within the data (scatter plots, histograms).

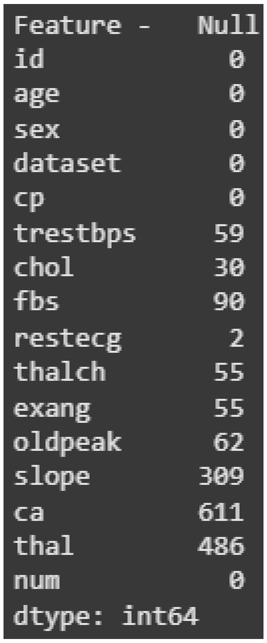
- **Correlation Analysis**:

- **Heatmap**: A correlation heatmap will help you visualize the relationships between numerical variables. High correlations might suggest redundant features, while low correlations could indicate independent predictors.

We used the insights of EDA to work smoothly in our data-preprocessing steps.

3. Dataset Pre-processing:

A) Missing values :



```
Feature -   Null
id             0
age            0
sex            0
dataset        0
cp             0
trestbps      59
chol          30
fbs           90
restecg        2
thalch        55
exang         55
oldpeak       62
slope        309
ca           611
thal         486
num            0
dtype: int64
```

ca (611 missing) → This is very high (≈ 66% of the dataset missing).

thal (486 missing) → Also high (≈ 53% missing).

slope (309 missing) → Moderate missingness (≈ 34%).

fbs (90 missing), **oldpeak (62)**, **trestbps (59)**, **thalch (55)**, **exang (55)**, **chol (30)** → Light to moderate missingness (3–10%).

restecg (2 missing) → will remove associated rows.

First we will handle the case of very high missing values. In the preprocessing stage, two features, **ca** (66% missing) and **thal** (53% missing), showed excessive missing data, making reliable imputation difficult and potentially introducing bias. Although they had some correlation with the target, retaining them could mislead the model due to poor data quality. Therefore, both features were dropped, as standard practice suggests removing variables with more than **50%** missing values. This ensures the dataset remains clean, consistent, and better suited for building a robust multi-class classification model.

Secondly, we deal with moderately high missing value slopes. As it is moderately high(34%), we will not drop it. We will use mod imputation to solve this issue, as the data is categorical, mean and median makes no sense here. And we do the same for all the missing values in other categorical features, as they have less number of missing values. So, we will use the **mode** of the data to fix the missing value issue.

Mean imputation is used for normally distributed features (thalch, trestbps). Skew
Median imputation is used for skewed features (chol, oldpeak).

B) Categorical encoding:

Sex (Binary: male = 1 , female = 0):
→ Here Binary encoding is done, since there are only two categories.
Fbs (Binary: true = 1 , false = 0) - only two categories
exang (Binary: yes = 1 , no = 0)- only two categories
Cp : (multiple categorical val) : We have done one hot encoding here making each category a different column.
Slope of ST Segment (slope: 0–2): Used One-Hot Encoding technique here to preserve all distinctions.
Resting ECG (restecg: 0–2): Used One-Hot Encoding technique here to preserve all distinctions.
Thal : used one hot encoding.

c) Feature Engineering:

The numerical features in our dataset (`'age', 'chol','trestbps','thalch','oldpeak'`

) are measured in **different ranges and units**. For example:

- age ranges from ~45–60
- chol (cholesterol) ranges from 100–600,
- oldpeak is mostly between 0–6

If left unscaled, features with larger numeric ranges dominate models that rely on distances or gradients.

Solution:

We applied **StandardScaler** from scikit-learn to rescale numeric features. StandardScaler transforms each feature to have:

Mean = 0
Standard deviation = 1

As we decided to use KNN, Logistic regression, Naive Bayes and NN :

- **KNN** uses Euclidean distance → all features must be on the same scale.

- **Logistic Regression & Neural Networks** use gradient descent → converge faster and more stable with standardized inputs.
- Naive Bayes : We used it just for testing and tuning purposes.

- StandardScaler is **less sensitive** to outliers, which is useful since chol and oldpeak are skewed. Unlike minmaxscaler that works poorly in these cases.

- Tree-based models (DecisionTree/RandomForest) don't require scaling, but our main models do, so we standardize for consistency.

Effect:
After standardization, each numerical feature is centered at zero with comparable scale, ensuring **no single feature dominates the model learning**.

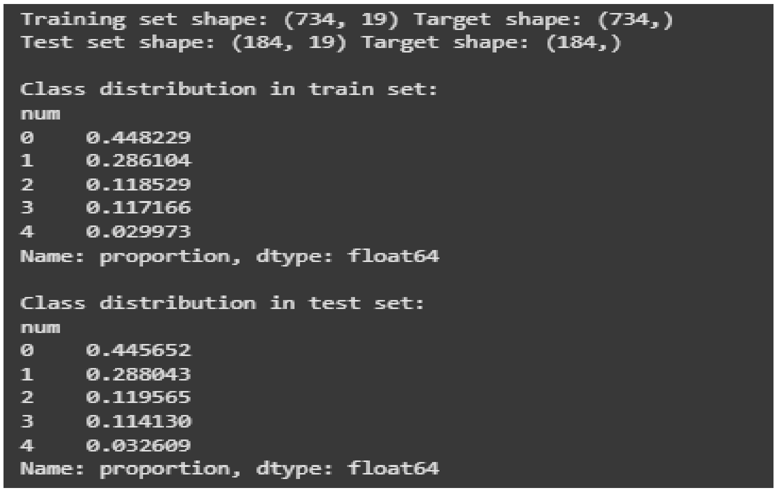
4. Dataset Splitting:

To evaluate our models fairly, we split the dataset into **training and testing sets**. The training set is used to build the model, while the testing set is held back to evaluate performance on unseen data.

- We used a **80/20 stratified split**

80% training data (used to fit the models).

20% testing data (used for final evaluation).



```
Training set shape: (734, 19) Target shape: (734,)
Test set shape: (184, 19) Target shape: (184,)

Class distribution in train set:
num
0    0.448229
1    0.286104
2    0.118529
3    0.117166
4    0.029973
Name: proportion, dtype: float64

Class distribution in test set:
num
0    0.445652
1    0.288043
2    0.119565
3    0.114130
4    0.032609
Name: proportion, dtype: float64
```

Why not stratified?
The target variable (0,1,2,3,4) is not perfectly balanced in the given data set. A simple random split could create uneven class distributions, where one set contains far fewer diseased patients. Though its a target we did not use the encoding process here which will increase our accuracy in a significant level.

Validation set:
We did not use a separate validation set because:

- For models like KNN and Logistic Regression, we use **cross-validation** to tune hyperparameters.

5. Model training & testing (Supervised)

We framed the task as binary classification; after cleaning/encoding, we standardized numeric features and used a stratified 80/20 train–test split.

Trained four models: K-Nearest Neighbors (tuned k), Naive Bayes, Logistic Regression (L2-regularized), and a Neural Network (MLP) with ReLU, sigmoid output, and EarlyStopping.

6. Model Selection / Comparison Analysis

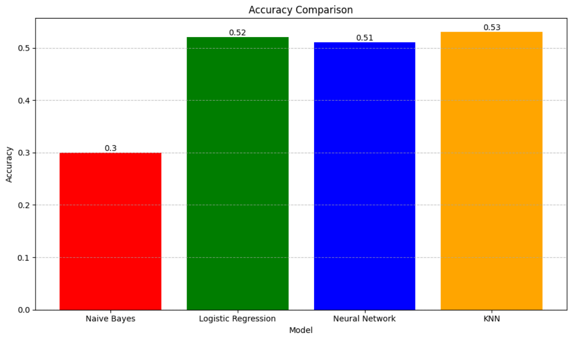
We trained and evaluated three supervised learning models: KNN, Logistic Regression, and Neural Network (MLP). Their performance was compared using accuracy, precision, recall, F1-score, confusion matrices, and ROC–AUC curves.

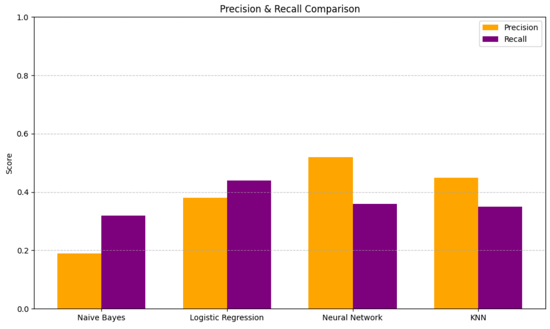
a. Accuracy Comparison

- Three models achieved similar accuracy (~50–60%). While one model achieved 30%
- Logistic Regression and KNN slightly performed better than the Neural Network and naive bayes performed badly.
- Indicates that the dataset is not highly nonlinear — linear models (like Logistic Regression) are sufficient.

b. Precision, Recall, and Comparison

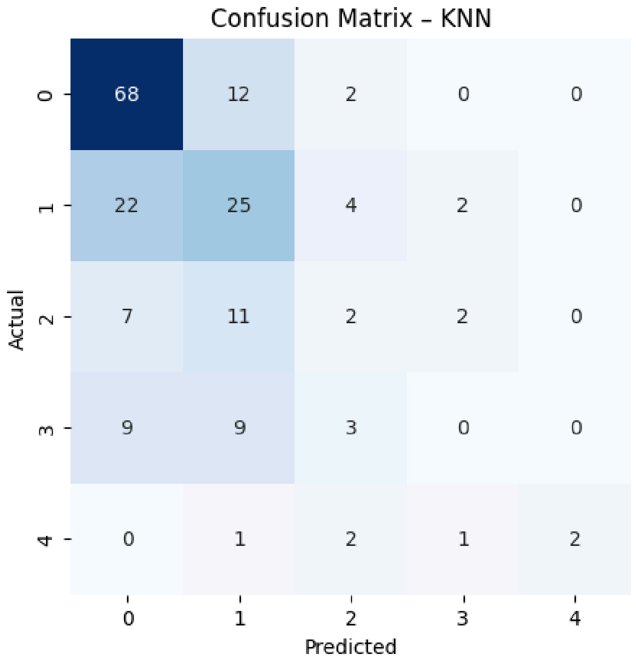
- KNN: Highest precision (~0.60), meaning it produced fewer false positives.
- Logistic Regression: Balanced precision and recall (~0. each), making it the most stable overall
- Neural Net: Competitive, but slightly lower recall.
- Naive Bayes : Performance not good



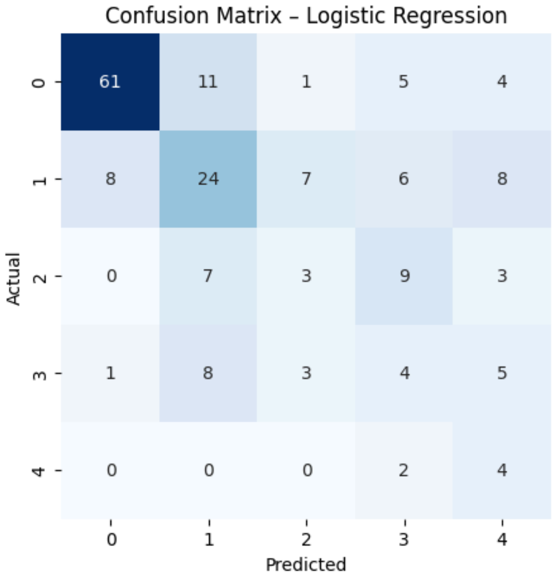


c. Confusion Matrices

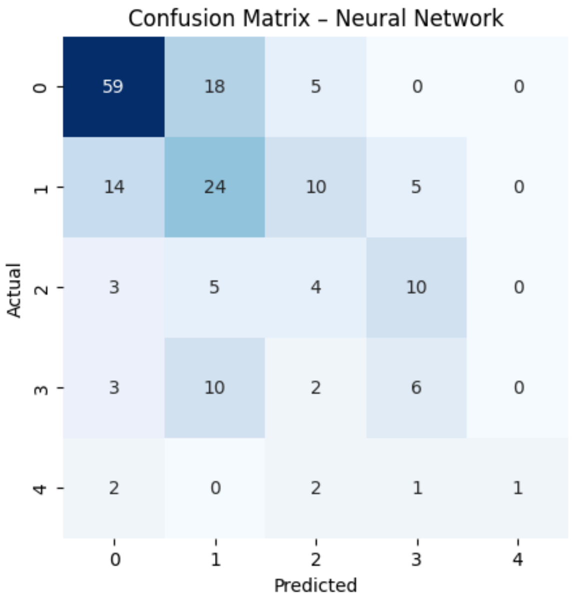
- KNN
  Correctly classified test samples, with a few more false negatives compared to Logistic.



- Logistic Regression

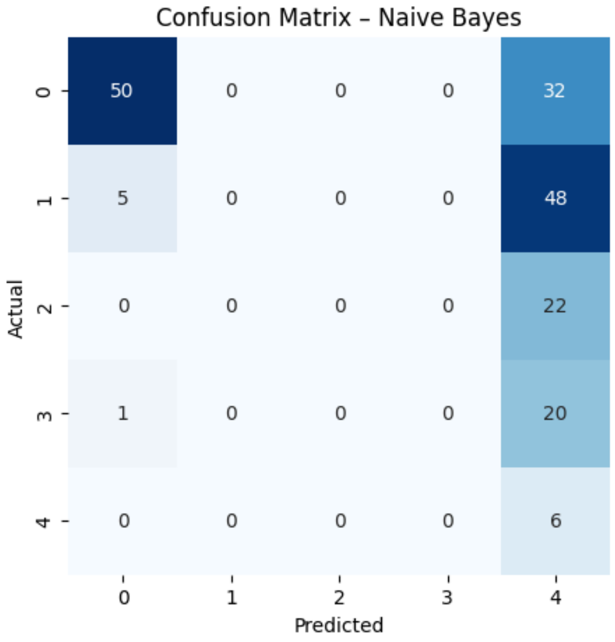


- Neural Network:



Actual \ Predicted	0	1	2	3	4
0	59	18	5	0	0
1	14	24	10	5	0
2	3	5	4	10	0
3	3	10	2	6	0
4	2	0	2	1	1

- Naive Byes :



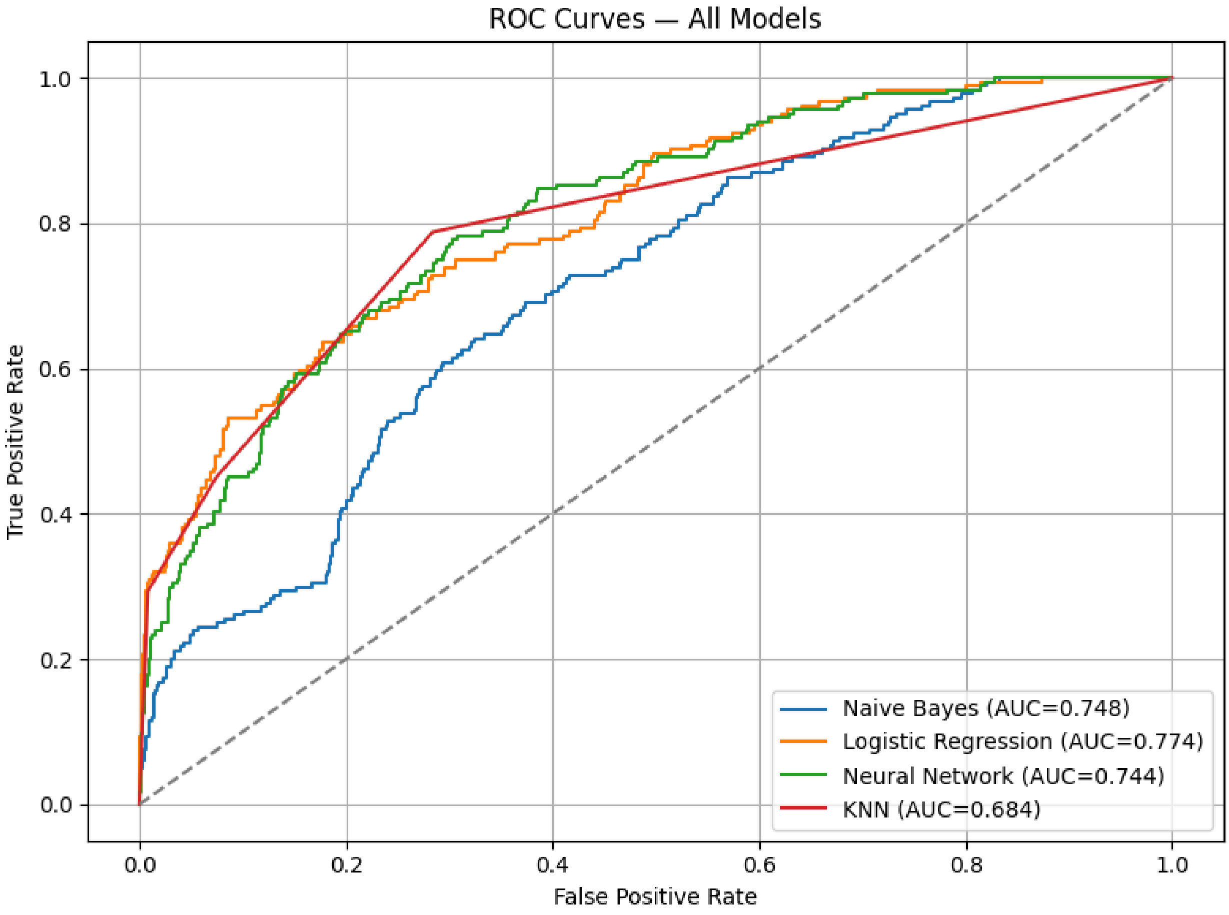
Actual \ Predicted	0	1	2	3	4
0	50	0	0	0	32
1	5	0	0	0	48
2	0	0	0	0	22
3	1	0	0	0	20
4	0	0	0	0	6

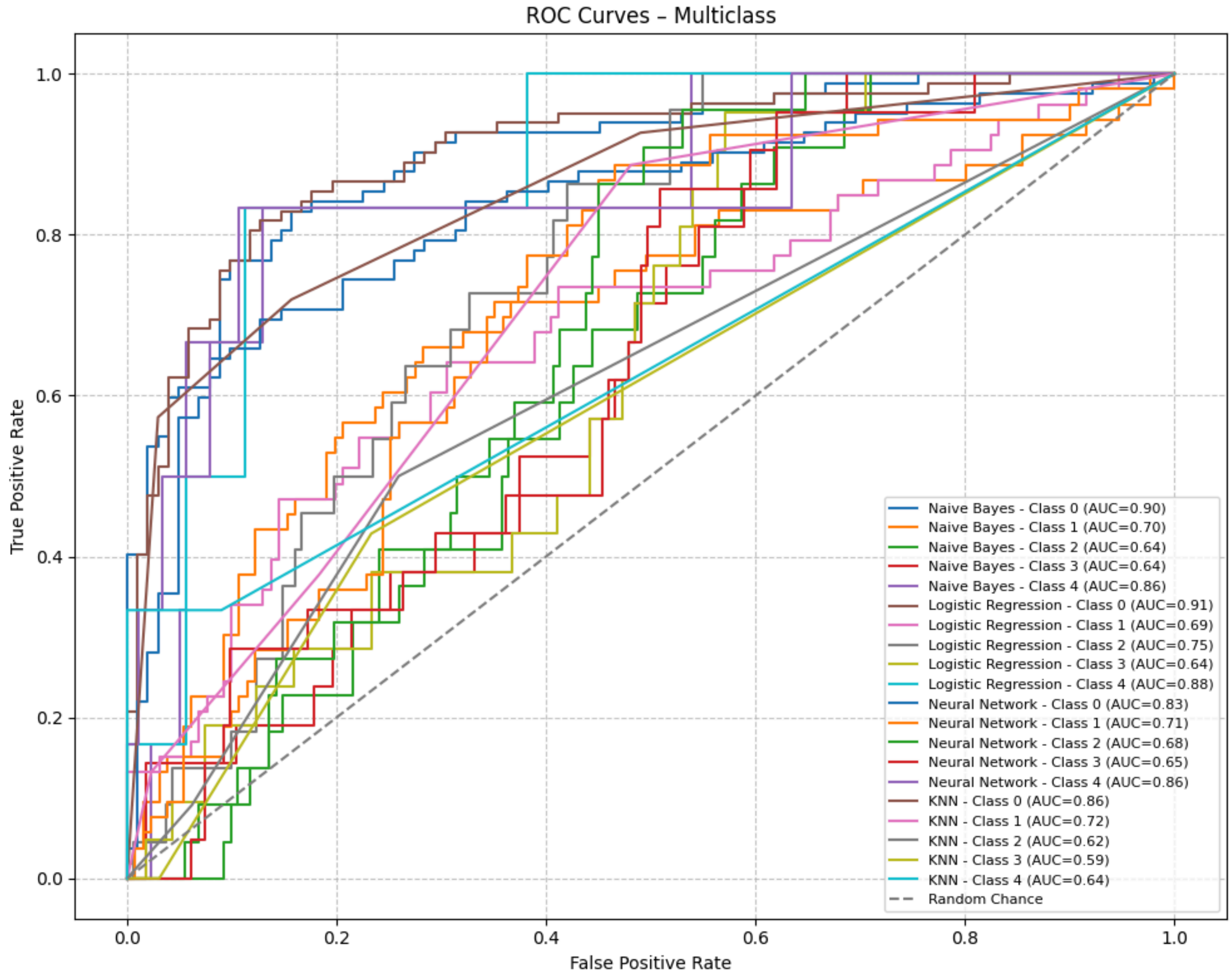
d. ROC Curves & AUC Scores

- Logistic Regression: AUC = 0.774 (best performance).
- KNN: AUC = 0.684
- Neural Net: AUC = 0.744.

- Naive Bayes - Auc = .748

All models are showing strong discriminative ability. Logistic Regression edges out slightly.

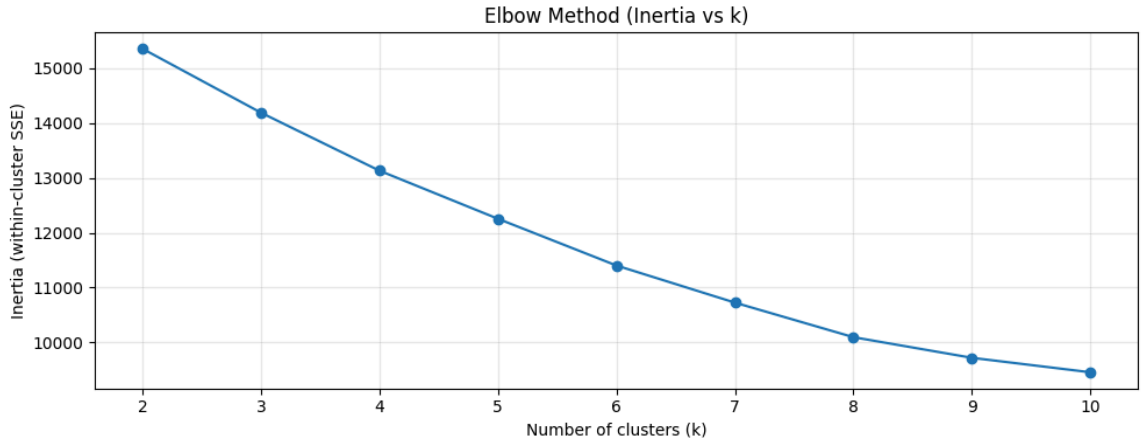


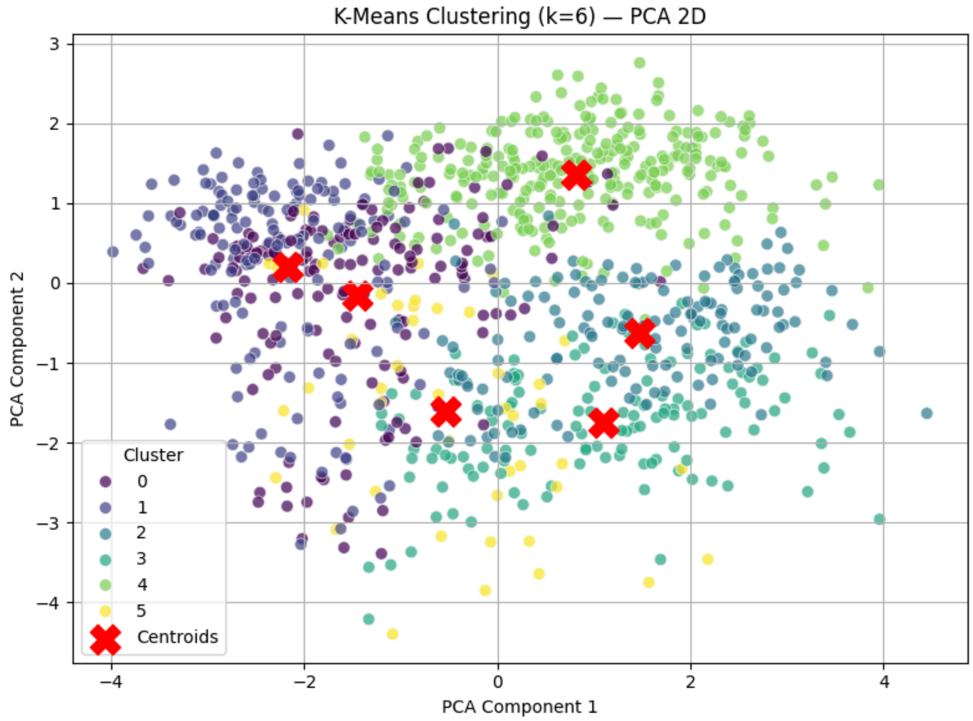


Logistic Regression performed most consistently across all metrics, with the best AUC and balanced precision/recall. KNN is a strong alternative, particularly when prioritizing precision. Neural Network did not outperform simpler models, suggesting the dataset size/complexity does not justify deeper architectures.

7. K-Means Clustering (Unsupervised Learning)

We applied K-Means clustering to the heart disease dataset as an unsupervised task. Using the Elbow method, the optimal cluster number was around k==6. After fitting K-Means and visualizing with PCA, patients grouped into clusters based on feature similarity, revealing natural patterns without using labels.





8. Conclusion

In this study, we explored various supervised learning techniques—such as Logistic Regression, K-Nearest Neighbors (KNN), Naive Bayes, and Neural Networks—to forecast heart disease, alongside an unsupervised K-Means clustering approach that uncovered various patterns in the dataset. These patterns roughly aligned with the presence or absence of heart disease, though the target variable itself was not simplified into binary categories. Instead, we retained its original multiclass format (ranging from 0 to 4), which introduced challenges due to significant class imbalance—specifically, 44% of the samples belonged to class 0, while the remaining were unevenly distributed across classes 1 to 4.

This imbalance contributed to lower overall model accuracy, particularly for models that struggled to generalize across the less-represented classes. While encoding the target variable into binary form (e.g., 0 for no disease and 1 for any presence of disease) showed potential for improving accuracy, we chose to maintain the multiclass structure to preserve clinical granularity. Among the models, Logistic Regression demonstrated simplicity and interpretability, fitting well with the classification task, whereas Neural Networks captured more complex patterns but demanded considerable computational resources and fine-tuning.

The overall performance of the models was moderate, reflecting the inherent complexity and ambiguity in medical data, where patient features often overlap and lack distinct separability. Key challenges included handling missing values, managing the skewed class distribution, and determining the most meaningful classification strategy. This project ultimately highlighted both the potential and the limitations of applying machine learning in healthcare contexts, emphasizing the need for careful data preprocessing, balanced model selection, and a commitment to interpretability—especially when outcomes impact real-world health decisions.