Milestone 1 Report: Heart Disease Prediction

Leonardo Matteo Bolognese Ilias Bouraoui Yann Gaspoz

April 20, 2025

1 Introduction

Heart disease prediction is critical for early intervention. We address the 5-class classification task on a 297-sample dataset with 13 features by implementing and comparing three methods: logistic regression, KNN, and K-means clustering. This report details data preparation, methodology, hyperparameter selection, and results.

2 Methodology

2.1 Data Preparation

From 237 training and 60 test samples, we further split the training into 80/20 for validation when tuning. Numerical features (age, trestbps, chol, thalach, oldpeak) were standardized (zero mean, unit variance). Categorical features were retained without encoding transformations beyond ordinal representation.

2.2 Logistic Regression

We implement multiclass logistic regression with softmax, optimizing cross-entropy via batch gradient descent. We tested learning rates $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$ and iteration caps of 100–1000. Convergence was typically achieved within 700 iterations at $\text{lr} = 10^{-2}$.

2.3 KNN

Our KNN implementation stores normalized training samples and, during inference, calculates Euclidean distances between each test sample and the training set. The predicted label is chosen through majority voting among the k nearest neighbors. This logic is wrapped inside a custom `KNN` class, which handles normalization based on training statistics and uses a helper function to perform lazy predictions. To determine the best value for k, we apply a validation-based tuning strategy, as detailed below.

2.4 K-Means Clustering

We apply K-means with k clusters and assign class labels by majority vote in each cluster. Before clustering, we normalize the data and reuse the same statistics during prediction for consistency. We used the elbow method (see Fig. 2) and validation-based tuning for $k = 1 \dots 15$, selecting $k = 4$ (highest macro F1). Multiple random initializations (`n_init=10`) ensured robust clustering by retaining the solution with the lowest inertia. The experiments were conducted using custom command-line flags `--elbow` to generate the inertia plot (Fig. 2) and `--tune_kmeans` to automatically select the best k based on validation F1 score (Fig. 3). The command `--n_init` can also be used to choose the wanted number of random initializations.

3 Experiments and Results

3.1 Hyperparameter Tuning

3.1.1 KNN

We tuned the hyperparameter k using validation accuracy. For $k \in \{1, 3, 5, \dots, 21\}$ (we use only odd values to avoid ties), we trained the model and computed accuracy on the validation set. The optimal value was found to be $k = 3$ (using `--tune_knn`), which yielded the highest validation accuracy. The final model used this value for testing. Since KNN is non-parametric, no fitting occurs beyond memorizing the training data; tuning primarily impacts the neighborhood sensitivity and decision boundary smoothness.

3.1.2 Logistic Regression

For easier testing, max_iters has been tuned with different lr values. For $max\_iters \in \{100, 200, \dots, 5000\}$ and $lr \in \{0.00001, 0.0001, \dots, 0.1\}$ we computed the training and testing accuracies and evaluated them to retrieve the correct hyper-parameters. From there, we took the optimal lr value to be 0.01 with max_iters being around 700, see Fig. 1 for more information.

3.1.3 KMeans

We use KMeans for unsupervised clustering, assigning labels by majority vote within each cluster. Data is nor-

malized before fitting, and the same statistics are reused at test time. To determine the optimal number of clusters k, we applied both the elbow method and macro-F1 validation tuning, selecting $k = 4$ as a trade-off between compactness and label alignment. The relevant plots are shown in Fig. 2 and Fig. 3.

3.2 Validation Plots

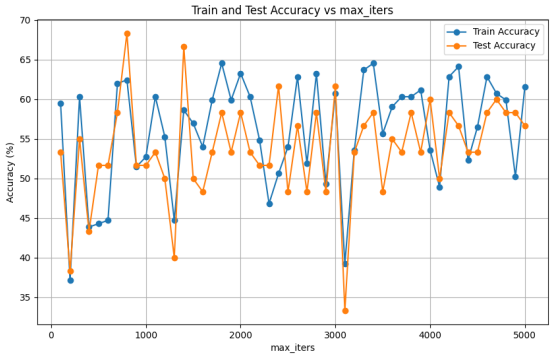

Figure 1: Training and validation accuracy across learning rates and iteration caps. The optimal configuration is found at lr $= 10^{-2}$ with approximately 700 iterations.

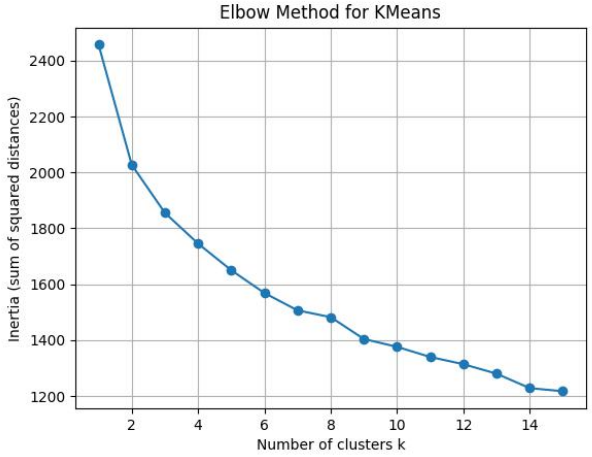

Figure 2: Elbow method to select k for K-means, for $k \in \{1, \ldots, 15\}$. A clear elbow appears at $k = 4$, beyond which inertia decreases more slowly, suggesting diminishing returns in clustering quality. Although other values such as $k = 9$ could also perform well, they lie beyond the elbow (more information is given in Fig. 3).

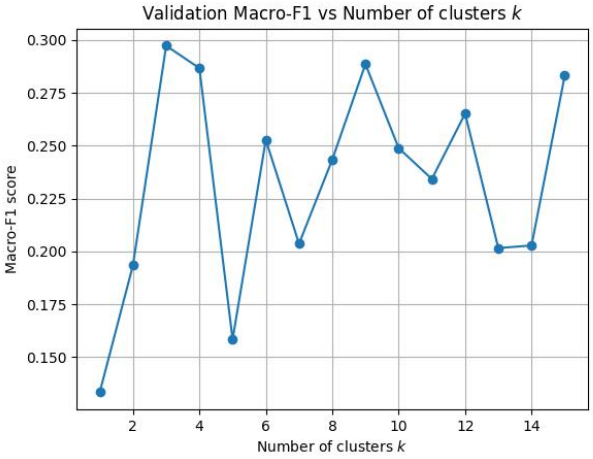

Figure 3: Validation macro-F1 score for $k \in \{1, \ldots, 15\}$ in K-means clustering. While $k = 9$ shows strong performance, we select $k = 4$ to remain consistent with the elbow method and avoid overfitting to validation noise.

3.3 Performance Metrics

Table 1 summarizes train/test accuracy and macro F1-score. See Section 4 for interpretation.

Method	Train Acc (%)	Test Acc (%)	Train F1	Test F1
KNN ($k = 3$)	73.4	58.3	0.540	0.305
Logistic Regression	63.3	58.3	0.329	0.28
K-means ($k = 4$)	59.1	58.3	0.243	0.226

Table 1: Evaluation on training and test sets.

3.4 Runtime Analysis

Training and prediction times were measured using Python's `time` module. KNN had the fastest training time at 0.004 s and predicted in 0.001 s, reflecting its lazy learning nature. Logistic regression was trained in around 0.1 s and predicted in under 0.001 s, while K-means required around 0.006 s for training and similarly under 0.001 s for prediction. All methods run efficiently on CPU given the small dataset size.

4 Discussion and Conclusion

KNN achieved good accuracy at 58.3%, benefiting from its adaptability to local data structures. However, as a nonparametric method, it would struggle in high-dimensional settings where distances lose meaning.

Logistic regression had a lower F1 score, but it remains a reliable and interpretable baseline. It is efficient in low-resource contexts but may miss complex patterns.

K-means clustering performed well without labels. With tuning, it reached good accuracy comparable to the accuracy of logistic regression.