CS-233 – Milestone 1 Report

Heart-Disease Classification with KNN, Logistic Regression and K-Means

Nour Alaoui Ismaili   Ghali Elouahdani   Nacer Saadani
373068_378613_381326

1 Introduction

This project (MS1) focuses on classifying patients with heart disease using supervised and unsupervised algorithms. The dataset contains 297 records with 13 clinical features. Labels 0–4 denote severity, so the task is 5-class classification. We re-implemented in pure `NumPy`, three classic algorithms:

- k-Nearest Neighbors (KNN) – lazy, non-parametric.
- Multi-class Logistic Regression – iterative, linear, probabilistic.
- K-Means – fully unsupervised, distance-based.

The task is five-class classification (labels 0–4). Because the class distribution is very skewed (Table 1), we report *both* global accuracy and macro F1-score. The entire pipeline (data $\rightarrow$ metrics) is orchestrated by `main.py`, respecting the skeleton provided by the course.

Class	0	1	2	3	4
Train count	128	41	30	30	8

Table 1: Training-set label counts (labels as floats).

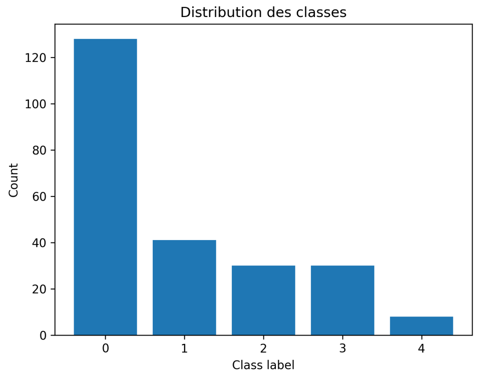

Figure 1: Visualization of Table 1.

1.1 Data preparation

We load the archive `features.npz`, which contains 297 patients with 13 clinical features. `main.py` first shuffles the 237-row training array with a fixed seed (`np.random.seed(100)`); when the flag `--test` is absent it keeps 20 % of these rows as a validation set, so hyper-parameter tuning is performed on data that faithfully imitates the hidden test distribution without ever seeing it. For every feature we compute the mean μ and standard deviation σ on the 80 % training slice, then apply the z-score transformation

$$x' = \frac{x - \mu}{\sigma}$$

to the training, validation, and test splits. This normalisation is indispensable, because raw medical attributes span very different ranges (e.g. cholesterol 126–564 vs. the binary variable `sex`). All preprocessing is carried out once in `main.py`, guaranteeing that every model receives identically scaled inputs.

1.2 KNN

During `fit`, our KNN simply memorises the entire training set (features and labels). At prediction time, for each test sample it computes the Euclidean distance to every stored training point, uses `np.argsort` to select the k nearest neighbours, and then applies `np.bincount` to choose the most frequent label among them. We evaluated several values of k on the validation split (see Fig. 2) to identify the optimal setting.

1.3 Logistic Regression

Logistic regression for multiclass problems learns a set of linear parameters and an intercept, then converts the resulting scores into class probabilities via a numerically stable softmax transformation. Parameters are fitted by minimizing the L_2-penalized cross-entropy loss using batch gradient descent. The step size and total number of iterations are determined by monitoring performance on an independent validation split.

1.4 K-Means

K-Means groups samples by iteratively alternating between two steps: (1) assigning each point to the nearest cluster center, and (2) updating each center to the mean of its assigned points. This process repeats until the centers no longer move significantly. Once fitted, we assign a class label to each centroid based on the majority label of its training members, and predict new samples by finding the closest labeled centroid.

2 Results

KNN — hyper-parameter study

After implementing KNN, we ran our model on the validation set to fine-tune the hyperparameter k, the number of neighbors. We evaluated values of k from 2 to 15, measuring validation accuracy and macro F1-score for each setting. Figure 2 shows how performance varies with k: the validation accuracy reaches its maximum of **68.1%** at $k = 12$, which also corresponds to the highest macro F1-score of **0.414**. Performance remains relatively stable from $k = 6$ to $k = 15$, but $k = 12$ clearly provides the best trade-off between both metrics. Considering this, we selected $k = 12$ for our final KNN configuration, yielding a validation accuracy of 68.1% and a macro F1-score of 0.414.

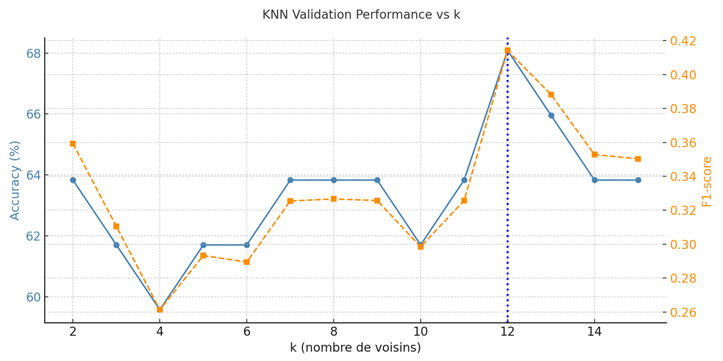

Figure 2: Validation accuracy (blue) and macro F1 (orange) vs. k.

K-Means — hyper-parameter study

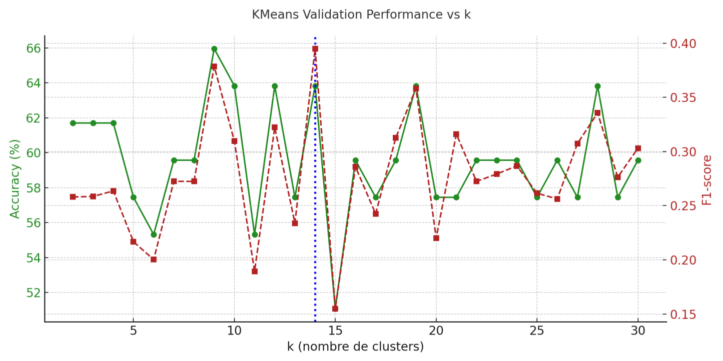

Figure 3: Validation accuracy and macro F1 vs. number of clusters K.

After implementing KMeans, we ran our model on the validation set to fine-tune the number of clusters k. We evaluated values of k from 2 to 30, measuring validation accuracy and macro F1-score for each setting. Figure 3 shows how performance varies with k: while validation accuracy peaks at **65.96%** for $k = 9$, the macro F1-score reaches its maximum of **0.3949** at $k = 14$. Performance fluctuates significantly across values of k, showing that KMeans is sensitive to this hyperparameter. Considering the best trade-off between accuracy and F1-score, we selected $k = 14$ as the final configuration, achieving a validation accuracy of 63.83% and a macro F1-score of 0.3949.

Logistic Regression — grid search

After implementing our multinomial logistic regression, we performed a grid search over learning rates

$$\eta \in \{10^{-5}, 5\times10^{-5}, 10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}\}.$$

and iteration counts

$$T \in \{50, 100, 200, 500, 1000\}.$$

Figure 4 shows that for very small learning rates ($\eta \leq 10^{-5}$), validation accuracy remains uniformly low (38.30%) regardless of T. As η increases to 5×10^{-4}, accuracy jumps to 46.81% at $T = 50$, and ultimately peaks at 65.96% in two configurations: ($\eta = 5 \times 10^{-3}$, $T = 50$) and ($\eta = 5 \times 10^{-4}$, $T = 500$). Macro F1 follows a similar trend, rising from 0.21 at the extremes to a maximum of 0.3434 at these same points. Balancing peak performance against training cost, we selected

$$\eta = 5 \times 10^{-4}, \quad T = 500,$$

which yields a validation accuracy of **65.96%** and a macro F1-score of **0.3434** for our final logistic-regression model.

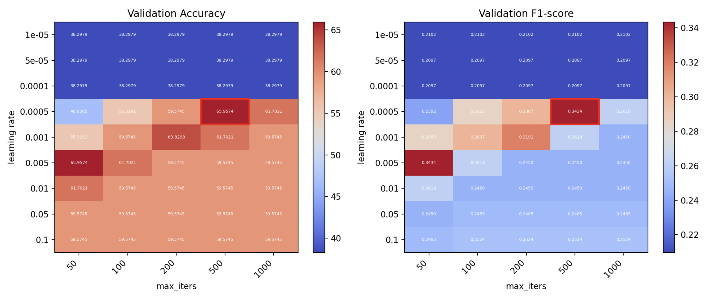

Figure 4: LogReg validation grids (accuracy left, F1 right).

Method	Grid	Best	Val F1
KNN	k 2–15	12	0.4140
LogReg	η, T	5x10^{-4}, 500	0.3434
K-Means	K 2–30	14	0.3949

Table 2: Selected hyper-parameters.

Conclusion

Model	Train Acc	Train F1	Test Acc	Test F1
KNN	56.5	0.24	56.7	0.28
LogReg	55.3	0.20	53.3	0.18
K-Means	54.9	0.17	55.0	0.19

Table 3: Final metrics on hidden test set.

Table 2 summarizes the hyper-parameters selected by validation F1: KNN with $k = 12$ (val F1 = 0.4140), logistic regression with $\eta = 5 \times 10^{-4}$, $T = 500$ (val F1 = 0.3434), and K-Means with $K = 14$ (val F1 = 0.3949). Applied to the hidden test set (Table 3), KNN achieves the highest test accuracy (56.7%) and test F1-score (0.28). K-Means follows with 55.0% accuracy and 0.19 F1, while logistic regression attains 53.3% accuracy and 0.18 F1. On the training set, KNN records 56.5% accuracy and 0.24 F1, logistic regression 55.3% accuracy and 0.20 F1, and K-Means 54.9% accuracy and 0.17 F1, indicating minimal over-fitting across all methods. These results show that KNN offers the best balance of overall accuracy and class-balanced performance; K-Means remains a strong unsupervised competitor; and logistic regression, though simple, yields lower metrics on this task.