CS-233 Project (2025) - Milestone 1 Report

Introduction

This report, written in collaboration by Alessandro Annibale Cioffi (358488), David Gorgiev (362628), and Sacha Heizmann (363265), details the steps taken to classify the Heart Disease Dataset[1] using three classification methods seen in CS-233: k nearest neighbours, k means and logistic regression. The goal is to predict the presence of heart disease at varying degrees (0, 1, 2, 3, 4), based on a set of 13 features[2]. We will start by explaining how we have pre-processed the data before training, followed by an exposition of the results (i.e. accuracy and F1-score) obtained for each method, and finally, an interpretation of the results.

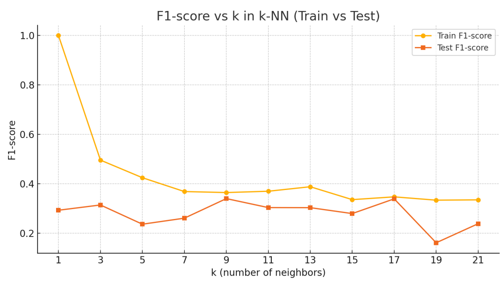
Methodology

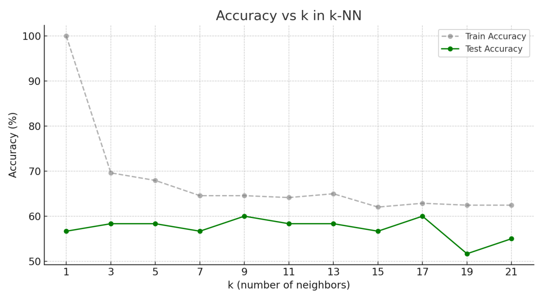
We analysed the dataset by loading it and printing out its features, leading to the identification of numerical features (age, trestbps, chol, thalach, oldpeak, ca) and categorical features (sex, cp, fbs, restecg, exang, slope, thal). We noticed strong class imbalance in the training data (128, 41, 30, 30, 8), which we addressed using a class weighting scheme, so as to prevent model bias toward the majority class.
We handled numerical feature pre-processing by subtracting the mean and dividing by the standard deviation, so as to ensure all features contributed equally during the training process. As for categorical features, we transformed them using one-hot encoding.
Finally, a bias term (i.e. a column of 1) was added to let the model learn an offset from the origin; this proves to be beneficial for the decision boundary, as it makes it more flexible compared to a model that does not use one (in those cases, the decision boundary would have to pass through the origin).
To address the class imbalance, an inverse frequency weighting approach was considered and implemented, as described in the k-NN slides. This way, underrepresented classes were weighed more than overrepresented ones. The weights were normalised to sum to the total number of samples, preserving the relative importance between classes.

Results

KNN

The k-NN model was tested with different values of k ranging from 1 to 21.





A small value of k = 1 of course resulted in perfect training accuracy and F1-score but overfitted. As we increased k, the model became more robust, though with slightly less flexibility on the training set.
The best test performance in terms of F1-score (0.339782) and accuracy (60.000%) was obtained with k = 9 and k = 17. This means that a slightly higher k provides a better balance between overfitting and underfitting.
To choose between 9 and 17, we performed 5-fold cross-validation for both k = 9 and k = 17.
The results were **0.3563** for k = 9 and **0.3144** for k = 17.
Cross-validation clearly shows that k = 9 generalizes better. It scores better than k = 17 and provides the best results. Hence, we selected **k = 9** as the final configuration for the k-NN classifier.

K-Means

The choice of the optimal hyperparameter was made through the test of every k from 1 to 100 using a loop. Three major candidates stood out: 7, 48 and 98.
Seven clusters seems to be the optimal choice for efficiency and precision among small integers. Nevertheless, the F1-scores obtained are too low.

	k = 7	k = 48	k = 98
Train Accuracy	**59.916**	68.776	**73.418**
Train F1	**0.283**	0.458	**0.619**
Test Accuracy	**51.667**	**60.0**	58.333
Test F1	**0.187**	0.318	**0.345**

[1] https://archive.ics.uci.edu/dataset/45/heart+disease, last accessed on 15.04.2025.
[2] *Idem*.

Forty-eight clusters offer the best trade-off between speed and precision. It has indeed the best accuracy on the test set and very good F1-scores and accuracy overall.
Finally one can observe that ninety-eight clusters outputs better F1-scores overall, but is the slowest and has lower accuracy on the test set. Because this can be a sign of overfitting, we selected k=48 to be the optimal choice of hyperparameter.

Logistic Regression
The logistic regression implementation uses a vectorized approach inspired by the 4th Python exercise set and several online resources[3][4][5]. Key features include sample weight support for handling class imbalance, and numerical stability optimizations by subtracting the maximum value from the scores before applying softmax[3]. Error computation follows standard practice: calculating the difference between softmax probabilities and one-hot encoded labels[4], then applying sample weights when provided. The gradient is computed as the matrix product of transposed training data and the error, driving weight updates to minimize cross-entropy loss. Implementation challenges included mastering array element-wise operations, vectorizing the softmax computation, and implementing the error weighting scheme. After experimenting with various learning rates (0.1, 0.01, 0.001, 0.0001) and iteration counts (100, 250, 500, 750, 1000, 1250, 1500, 2500), the most notable configurations are shown in the following table:

Learning Rate	Weights	Max Iters	Test Accuracy	Test F1-Score	Notes
0.0001	Yes	100	65.000%	0.464091	Best overall performance
0.0001	Yes	250	61.667%	0.411830	Strong runner-up
0.1	No	100	60.000%	0.349244	Best high learning rate config
0.001	No	2000	56.667%	0.279683	Overfitting case
0.01	Yes	250	30.000%	0.191538	Worst test accuracy
0.1	Yes	2000	48.333%	0.161111	Worst F1-score

The mid-range max iteration values (500-1500), along with intermediate learning rates (0.1, 0.01, 0.001) yielded mixed results, which were typically worse than those shown. They were thus omitted.

Interpretation and Conclusion

The k-NN method showed clear trends as we varied the value of k. With small values, the model achieved perfect accuracy on the training set but performed worse on the test set, indicating overfitting. As we increased k, the model started to generalize better and the test results got better — but only up to a certain point. At k=9, we saw the best balance between how well the model did on both the training and test sets even though the results are stable for k from 1 to 21. After that, when k got too large, the performance slightly dropped again, probably because the model became too simple. Using cross-validation, we found that k=9 gave the best overall F1-score (0.3563), so that's the value we chose.

K-Means unsupervised method showed that even though there are 5 true classes in the data set, modeling the data with more clusters helped reveal substructures, leading to better performance. This suggests that the class labels may represent broader categories that internally contain multiple distinct data distributions. F1-score and accuracy trends show that performance improves as K increases, but only up to a point. After that, test performance drops as the model overfits the training data.

In logistic regression, excessive iterations led to poorer performance due to overfitting. For most learning rates, unweighted models performed better, with one crucial exception: at 0.0001 learning rate, weighting combined with low iteration counts (100-250) produced exceptional test accuracy and F1 scores. This optimal configuration addresses the dataset's class imbalance problem, while the low learning rate enables stable convergence.
Notably, our full experiments revealed that at 0.0001 learning rate, weighted models consistently outperformed unweighted ones across all iteration counts (100-2500), showing steady improvement in F1 scores (0.464 to 0.380). In contrast, higher learning rates (0.1, 0.01) showed significant performance instability regardless of weighting, with test F1 scores varying by up to 0.15 between consecutive iteration settings.
For this method, we recommend learning rate 0.0001 with class weighting and 100-250 iterations for optimal heart disease classification, as this configuration balances training efficiency with the highest test performance metrics.

We briefly conclude by observing that logistic regression yields the best results, with an outstanding F1 score of 0.464 and test accuracy of 65%. For the heart disease dataset, we would thus recommend this supervised classification method with the optimal parameters discussed prior, due to efficient class imbalance handling through weighting, as well as clear decision boundary delimitation between heart disease categories.

---

[3] https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/, last accessed on 15.04.2025
[4] https://peterroelants.github.io/posts/cross-entropy-softmax/, last accessed on 15.04.2025
[5] https://medium.com/@koushikkushal95/logistic-regression-from-scratch-dfb8527a4226, last accessed on 17.04.2025