CS-433: Project 1
Medical Dataset

Our code

Alireza Sakhaeirad
EPFL
SCIPER:373331

Alireza Abdollahpoorrostam
EPFL
SCIPER:380830

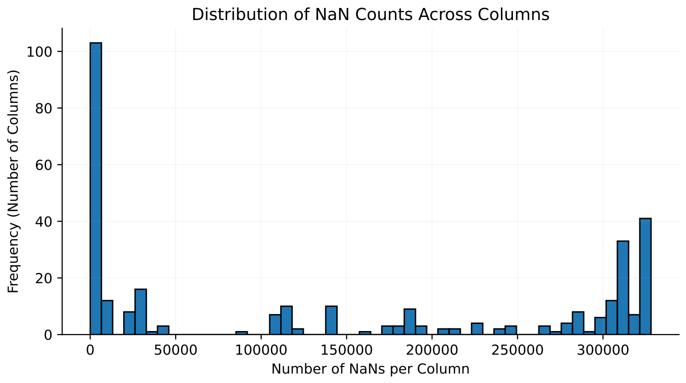
Moein Nasiri
EPFL
SCIPER:390183

1 Data Processing

This section provides a concise analysis of our data. The total number of `NaN` values in the *training* data is $47,175,779$, representing 44.79% of the dataset. Similarly, the total number of `NaN` values in the *test* data is $15,724,379$, also accounting for 44.79% of the dataset. Additionally, There are 239 columns containing `NaN` values, which we remove..

Statistic	Value
Mean NaNs per column	146,965.04
Max NaNs in a column	328,103
Columns with all NaNs	0
Columns with no NaNs	82

(a) NaN Summary Statistics (Per Column)



(b) `NaN` distribution.

Figure 1: Overview of missing-value statistics and their distribution across dataset columns.

Fixing Data Imbalance. To address the strong class imbalance in the training set, we oversampled the minority (positive) class to match the number of majority (negative) samples. Specifically, we computed binary labels as $y_{\text{train\_binary}} = (y_{\text{train}} + 1)/2$ and determined the counts of positive and negative samples. The minority class (28,975 samples) was replicated until its size equaled that of the majority class (299,160 samples), yielding a balanced dataset with a total of 588,910 samples and a near-unity ratio of $1.03 : 1$. Random shuffling ensured that no ordering bias remained. This balanced dataset was then used for all subsequent training experiments.

Model Selection and Cross-Validation. To ensure fair model comparison and mitigate overfitting, we implemented a systematic *model selection pipeline* integrating k-fold cross-validation. Specifically, the training data were partitioned into $k = 5$ folds, with each fold serving as a validation set once while the remaining $k - 1$ folds were used for training. The average performance across folds provided an unbiased estimate of generalization capability and guided hyperparameter tuning for each classifier. This framework was fully automated, enabling consistent evaluation across all models and simplifying the selection of optimal configurations (e.g., regularization strength, number of estimators, or hidden layer sizes).

Feature Normalization. To ensure consistent feature scaling and improve model convergence, we implemented two normalization techniques: `MinMaxScaler` and `StandardScaler`. The `MinMaxScaler` linearly rescales each feature to a specified range-defaulting to $[0, 1]$, using the transformation

$$x' = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})} \cdot (r_{\max} - r_{\min}) + r_{\min},$$

thereby preserving relative distances between samples. The `StandardScaler`, on the other hand, standardizes features by removing the mean and scaling to unit variance via

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

where μ and σ denote the feature-wise mean and standard deviation. Both classes were implemented from scratch to maintain full control over numerical stability (e.g., handling zero variance features) and to support seamless integration with the experimental pipeline.

2 Training

We used `Logistic Regression`, `KNN`, `Random Forest`, and `MLP` classifiers for training.

Logistic Regression. We trained a regularized `logistic regression` model using three-fold cross-validation to select the optimal regularization parameter $C \in \{0.1, 1.0, 10.0\}$. The grid search identified $C = 10.0$ as the best configuration, achieving a mean cross-validation accuracy of approximately 0.495. Evaluation on the held-out validation set yielded an overall accuracy of 0.492. As shown in the classification report, the model exhibited a strong bias toward the positive class, achieving a recall of 1.00 and an F_1-score of 0.66 for the positive class, while completely failing to detect the negative class (precision and recall equal to 0.00). The macro-averaged F_1-score was 0.33, indicating poor generalization and highlighting that the model struggles to separate the two classes effectively despite the balanced training data.

After applying feature normalization using the `MinMaxScaler`, we retrained a regularized `logistic regression` model with three-fold cross-validation to tune the regularization strength $C \in \{10, 100, 1000\}$. The best configuration was found at $C = 1000$, achieving a mean cross-validation accuracy of approximately 0.723. On the held-out validation set, the model reached an overall accuracy of 0.725. Both classes were predicted with comparable performance, yielding precision and recall values of around $0.70 - 0.75$ and a macro-averaged F_1-score of 0.73. These results indicate a significant improvement over the unnormalized version, showing that proper feature scaling greatly enhanced the model's stability and discrimination capability across both classes.

K-Nearest Neighbors. We next experimented with a `K-Nearest Neighbors (KNN)` classifier using a five-fold cross-validation setup. The model was evaluated over a grid of hyperparameters including the number of neighbors $k \in \{3, 5, 11\}$ and weighting schemes `{'uniform', 'distance'}`. Due to the high dimensionality and large size of the balanced dataset, we restricted training to a random subset comprising one-third of the total samples to ensure computational feasibility. Despite this adjustment, training with the `KNN` classifier proved to be excessively time-consuming. Consequently, we decided to discontinue training with this method and transition to the `Random Forest` classifier, which offers a more scalable and efficient alternative for large-scale data.

Random Forest. Following the limitations encountered with the `KNN` classifier, we trained a `Random Forest` model composed of 300 decision trees, each constrained to a maximum depth of 300. The model employed the Gini impurity criterion, used the square-root rule for feature selection at each split, and enabled bootstrapping for improved generalization. Training was parallelized across all available CPU cores. Evaluation on the validation set demonstrated excellent performance, with an overall accuracy of 0.988. The classifier achieved nearly perfect discrimination between classes, yielding precision, recall, and F_1-scores all close to 0.99 for both positive and negative samples. The macro- and weighted-average F_1-scores were also 0.99, indicating strong predictive capability and robustness across classes.

Multilayer Perceptron (MLP) Classifier. We further trained a `Multilayer Perceptron (MLP)` model to capture potential nonlinear relationships among features. The network architecture consisted of two hidden layers with 128 and 64 neurons, respectively, each using the `ReLU` activation function. Training was performed using the `Adam` optimizer with a learning rate of 10^{-3}, an L_2 regularization coefficient (α) of 10^{-4}, and a maximum of 50 iterations. To enhance convergence stability, early stopping was enabled with a 10% validation split and a tolerance of 10^{-4}. The model was trained with shuffled mini-batches and a fixed random seed for reproducibility, aiming to balance expressive power with generalization and provide a robust nonlinear baseline for comparison against the linear and tree-based classifiers.

The MLP achieved an overall validation accuracy of approximately 0.796, with balanced precision and recall across both classes. The macro- and weighted-average F_1-scores of 0.80 indicate that the model successfully captured nonlinear relationships while maintaining good generalization between positive and negative samples (See Table 1).

We **further trained a `MLP` model within a preprocessing pipeline incorporating the** `MinMaxScaler` to ensure consistent feature scaling before training. The MLP achieved an overall validation accuracy of approximately 0.796, with well-balanced precision and recall across both classes. The macro- and weighted-average F_1-scores of 0.80 indicate that the model effectively leveraged nonlinear feature interactions and benefited from MinMax normalization, resulting in consistent and robust predictive performance across the dataset (See Table 1).

Table 1: Comparison of MLP classifier performance on the validation set, with and without `MinMax` normalization.

Metric	**Without MinMax Normalization**			**With MinMax Normalization**		
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Class 0	0.75	0.68	0.72	0.82	0.76	0.79
Class 1	0.70	0.77	0.73	0.77	0.83	0.80
Accuracy		0.725			0.796	
Macro avg	0.73	0.73	0.73	0.80	0.80	0.80
Weighted avg	0.73	0.73	0.72	0.80	0.80	0.80