Liver disease classification

Jordi Corbalan Vilaplana
UPC, Barcelona, Spain

Ona Siscart Noguer
UPC, Barcelona, Spain

***Abstract*—Early detection of liver disease is essential to improving clinical outcomes and reducing pressure on healthcare systems. This project addresses the problem as a binary classification task using clinical and demographic data. Several supervised learning models were evaluated under experimental conditions, with a focus on understanding their behavior under data constraints such as class imbalance, limited sample size, and potential feature redundancy.**
The methodology included data analysis, feature engineering, class imbalance mitigation, and the use of cross-validation for robust evaluation. Preprocessing strategies were designed to adapt to model-specific requirements, while performance assessment prioritized the F1-score to balance precision and recall, particularly due to the importance of correctly identifying sick patients.
Despite time and model restrictions, results suggest that even simple models can perform competitively when properly tuned and evaluated. The project also highlights the trade-offs between model complexity, interpretability, and performance in sensitive domains like healthcare.

***Index Terms*—medical machine learning, binary classification, supervised learning, liver disease.**

I. Introduction

This project aims to develop machine learning models capable of distinguishing between healthy individuals and patients with liver disease using clinical data. By analyzing patterns in medical measurements, the objective is to support medical decision-making through automated and accurate predictions. More specifically, given a database of biometric variables from different patients, the goal is to determine whether each individual is classified as ”healthy” (1) or ”sick” (0).

II. Data and feature analysis

A. Dataset

1) **Provenance:** Obtained from the UCI Machine Learning Repository, containing clinical and demographic data collected from patients in the North East region of Andhra Pradesh, India.
2) **Structure:** Includes records of both liver disease patients and healthy subjects, with 10 numerical features per subject.
3) **Observations:** After preprocessing performed prior to this project, the dataset contains 579 samples: 414 liver patients and 165 healthy subjects.
4) **Previous Preprocessing:** The data preprocessing, carried out before this study, involved removing samples with missing values and converting categorical variables to numerical format.

B. Exploratory data analysis

The dataset includes the following ten variables:

1) **Age:** Age of the patient.
2) **Female:** Gender of the patient (1 if Female, 0 if Male).
3) **TB:** Total Bilirubin.
4) **DB:** Direct Bilirubin.
5) **Alkphos:** Alkaline Phosphotase.
6) **Sgpt:** Alamine Aminotransferase.
7) **Sgot:** Aspartate Aminotransferase.
8) **TP:** Total Proteins.
9) **ALB:** Albumin.
10) **A/R:** Albumin and Globulin Ratio.

Only 80% of the dataset, corresponding to 463 observations, was used for training, validation and exploratory analysis. The remaining 20% constitutes the test set, which is reserved for final evaluation. Initial data exploration involved summary statistics and visualization to understand variable distributions and potential relationships with the target variable.

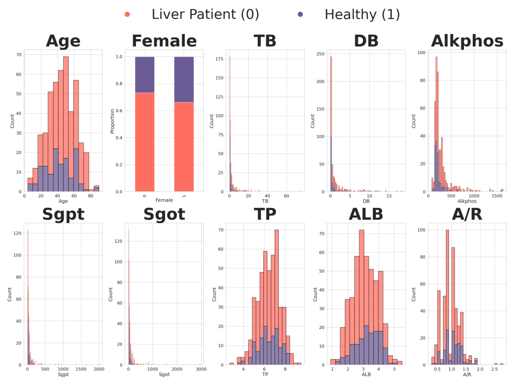

Fig. 1. Distribution of the variables in the provided dataset.

Figure 1 shows the distributions of the variables in the training dataset.

Two main observations can be made:

1) The dataset is highly imbalanced, with many more liver patients than healthy subjects. This imbalance will be addressed in later sections.
2) Several variables show skewed distributions and differ greatly in their value ranges, which may affect model performance if not properly handled.

Figure 2 displays the correlation matrix of the variables.

As shown in Figure 2, certain variables are highly correlated. Notably, DB (Direct Bilirubin) and TB (Total Bilirubin),

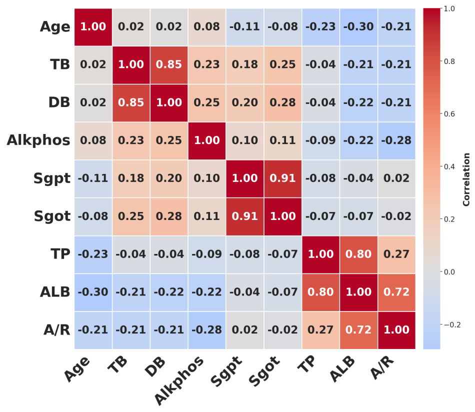

Fig. 2. Correlation matrix of the variables in the training dataset.

as well as Sgpt (ALT) and Sgot (AST), show strong positive correlations. ALB (Albumin) is moderately correlated with TP and A/G ratio. These correlations suggest some redundancy in the data, which could be taken into account for feature selection or dimensionality reduction.

C. Feature Engineering

The main goal was to compress highly correlated variables and enrich the dataset with mathematically informed features. Below are the key engineered variables considered (not necessarily included in final models):

- **SGPT/SGOT (corr. 0.91)**: ratio between liver enzymes ALT and AST to reduce redundancy and enhance interpretability.
- **B-ratio (DB/TB) (corr. 0.85)**: proportion of direct bilirubin relative to total, summarizing liver function efficiency.
- **Globulins = TP – ALB (corr. 0.80)**: non-albumin protein content, indicative of inflammation or immune response.
- **Log-Likelihood to Each Class**: By approximating each class density using Kernel Density Estimation (KDE), the log-likelihood of each sample under both class distributions was incorporated as a new feature, aiming to capture class-specific structure in the feature space.

Feature selection was conducted independently for each model using RFE and weight-based importance, in accordance with the assumptions of each classifier.

D. Data preprocessing

The preprocessing approach consisted of implementing different techniques, that could then be selectively applied to models that required specific data transformations or assumptions satisfied.

1) Missing and duplicated values: No missing values were detected during the Exploratory Data Analysis. However, 10 duplicate rows were found and removed, as identical biometric records across patients were deemed unlikely and likely erroneous.

2) Outlier treatment: After carrying out an IQR-based analysis of outliers, we found a large percentage of them. We attributed this to the imbalance in the data and assumed that the outliers might correspond to healthy patients. Regardless, due to the high proportion of outliers and the limited size of the available dataset, no outlier treatment was applied.

3) Transformations: Two diferent types of transformations were implemented as part of the preprocessing stage.

a) The logarithm transformation: was implemented in order to center variables presenting a skew above 1. The reasoning behind this transformation was that certain models, such as Logistic Regression or Support Vector Machines, are quite sensitive to skewed distributions, although they do not assume normality.

b) The scaling of variables: was performed using both Box-Cox and standard scaling, applied only to numerical features and excluding the categorical `Female` column. These transformations allowed models assuming normality to benefit from Box-Cox, while others used standard scaling. To prevent data leakage, both scalers were implemented within pipelines and fitted only on training data during cross-validation. For Box-Cox, the lambda parameter was optimized per variable via cross-validation. The log transform was not explicitly applied, as the grid search would naturally select $\lambda = 0$ if it were optimal.

4) Resampling: Class Imbalance Treatment in the pre-processing stage: Handling class imbalance was identified as a critical step due to the marked disproportion in the dataset. Techniques such as SMOTE were evaluated but did not provide satisfactory improvements considering the risks of oversampling. Under sampling was also considered; however, it was discarded as the limited dataset size required maximizing the use of all available data. Consequently, the final approach adopted model-internal weighting by setting the `class_weight='balanced'` option.

III. Classification Methods

In order to tackle the problem of diagnosing sick liver patients, several of the classification methods and models seen in class were tried, both simple and combined in ensembles:

- **Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA)**: Generative models, which assume a normal distribution of the data and take into account the covariance of classes. The only tuning that was carried out was for the QDA regularization parameter.
- **K-Nearest Neighbors (KNN)** : Non-parametric classifier based on distance between points. Simple model, but sensitive to data scaling and noise. Tuning was made for hyperparameters k and the distance used.
- **Gaussian Naive Bayes (GNB)**: Generative classifier that assumes normality and conditioonal independence between features within a class.
- **Logistic Regression**: A linear discriminative model, uses a sigmoid function to estimate class probabilities. It was

particularly useful for testing the impact of resampling techniques. It is sensitive to data scaling and skew.

- **Support Vector Machines (SVM)**: A powerful linear classifier that maximizes the margin between classes. Sensitive to outliers and scaling. Tuning involved kernel selection and regularization parameters.
- **Tree-based classifiers**: Decision Trees, Random Forests, and Extra Trees were employed. These models require minimal preprocessing and handle non-linear relationships well. Hyperparameters such as tree depth and number of estimators were tuned.
- **Ensemble methods: Hard and Soft Voting**: Voting ensembles were created to combine predictions from multiple classifiers.
- **Stacking**: A more advanced ensemble technique where the outputs of base classifiers are used as inputs for a meta-classifier. This can improve performance by leveraging the strengths of multiple models.

Although all of the aforementioned models were tried, some were discarded after an initial fit as they were deemed inadequate for the problem at hand.
Additionally, given the imbalance in our data, different methods of resampling and dealing with imbalance were attempted:

- `class weight = 'balanced'`: For models that support this option (such as SVMs, Random Forests, and Logistic Regression), class weighting was implemented. As discussed in class, handling class imbalance directly within the model is often more effective than relying solely on data resampling techniques.
- **Resampling: SMOTE** As mentioned, SMOTE was used as a method of oversampling of the minority (healthy) class, but was eventually discarded.

An additional way to deal with class imbalance is using the appropriate metrics, which we explore in the following section.

IV. Evaluation metrics

The main evaluation metric used was the **F1-score**

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1}$$

, as it is the most appropriate for unbalanced datasets. F1-scores were computed both via cross-validation and on the entire training set. In addition, precision and recall were examined to better interpret the F1 results. This allowed to keep tracking of how much the model was really overfitting the training dataset.

Accuracy was considered as a secondary metric.
Given the health-related nature of the data, F1-scores were also computed per class, as correctly identifying sick patients was deemed particularly important. As a result, precision was given slightly more weight in our analysis, since it reflects how reliable a "healthy" (class 1) prediction is. Minimizing the risk of falsely classifying a sick patient as healthy was a key priority.

V. Experiments

A. Experimental Procedure

An iterative procedure was followed to determine the optimal configuration and tuning for each model. Initially, a preliminary fit (using a general gridsearch over hyperparameters) of all candidate models was performed using default parameters and minimal preprocessing to assess their suitability for the task. Based on these results, models showing potential were further refined through targeted hyperparameter tuning, feature engineering, and feature selection techniques such as Recursive Feature Elimination (RFE).

The preprocessing pipeline was adapted according to the specific assumptions and requirements of each model type, ensuring compatibility with their underlying mechanisms (e.g., scaling for distance-based models or regularization-aware transformations). Feature engineering steps included the creation and evaluation of new variables as well as the analysis of inter-feature relationships to enhance model performance.

*1) **Model tunning and comparison**:*

a) Pipeline implementation for fitting and tunning models: The use of pipeline structures was adopted to streamline the model training and evaluation process while minimizing the risk of common mistakes such as data leakage. By integrating preprocessing steps (such as scaling and feature selection) directly into the pipeline alongside the estimator, it was ensured that all transformations were fitted exclusively on the training data within each cross-validation fold.

b) Hyperparameter Tuning: A custom function was implemented to perform hyperparameter tuning using grid search with cross-validation, selecting the configuration that achieved the highest macro-averaged F1 score on the validation folds.

c) Cross-Validation Strategy for comparing models: Once the best hyperparameters were found, models needed to be compared. A stratified 10-fold cross-validation approach was employed to ensure that each fold preserved the original class distribution. This method was preferred over a single fixed validation split as it provides a more robust performance estimate, makes better use of the our limited data, and reduces the variance of evaluation metrics across different data partitions. A 5·k-fold cross-validation strategy was also considered; however, it was discarded since the computational cost reduction did not justify the slight differences observed in the resulting performance metrics.

d) Best model Comparisons: Among the evaluated metrics, special attention was given to the macro-averaged F1 score obtained through cross-validation, as it provides a robust estimate of the model's generalization ability. Additionally, the relationship between cross-validation F1 macro and the F1 macro computed on the entire training set was considered to assess potential overfitting or underfitting. As previously mentioned, particular importance was also placed on the per-class F1 scores, given the clinical nature of the problem and the critical need to correctly identify positive (diseased) cases.

B. Summary of experiments

*a) **Initial Selection**:* Simple models like GNB, Logistic Regression, and QDA performed well in the initial evaluation, likely due to their low complexity and limited need for tuning. RF and SVM showed signs of overfitting but were retained for their potential with further hyperparameter optimization. KNN, LDA, and Decision Trees were discarded due to low performance (F1 macro below 0.6) or redundancy (RF is already including decision trees). The models selected for further development are QDA, GNB, Logistic Regression, SVM, RF, and ETC. Additionally, SMOTE was discarded due to generating synthetic clinical data that seemed unrealistic and because it performed worse than using class weight balancing.

Model	F1 Macro (CV)	F1 Macro (Train)
ETC	0.644	0.787
SVM	0.641	0.834
GNB_best	0.632	0.644
Logreg_best	0.623	0.638
Logreg_smote	0.619	0.637
QDA_best	0.618	0.698
RF	0.607	0.989
Tree	0.599	0.832
KNN_best	0.569	0.731
LDA	0.558	0.603

TABLE I
INITIAL SELECTION RESULTS

Initial selection results based on cross-validation and training macro F1 scores (see Table I). The full set of evaluation metrics for all models can be consulted in the project's notebooks.

*b) **Simple Models: QDA, Logistic Regression, and GNB**:* [1]

For the simple models (QDA, Logistic Regression, and GNB), we retained the hyperparameters obtained from the initial basic grid search. Due to their simplicity and limited capacity for tuning, no additional hyperparameter optimization was performed beyond the initial grid search.

Instead, we focused on assessing the impact of the newly engineered features. A targeted grid search was conducted to evaluate whether including or excluding these features improved model performance. The results showed that adding the engineered variables generally led to better performance; however, removing the originally correlated features resulted in performance degradation. Therefore, the best-performing approach was to include some of the newly engineered variables (different subset for each model) without discarding any of the original correlated features.

The preprocessing steps applied to these models were as follows:

- **QDA and GNB**: Normal features were Box-Cox transformed, as normality is assumed.
- **Logistic Regression**: Skewed features were log-transformed as no Box-Cox transformation was applied. Additionally, features were scaled.

[1]Further details on the final hyperparameters and selected features for each model can be found in the project notebooks.

*c) **Support Vector Classifier**:* The SVM model was tuned with a primary focus on selecting the most appropriate kernel and addressing multicollinearity and noise through feature engineering. Separate grid searches were conducted for linear, RBF, and polynomial kernels. Although the RBF kernel is often effective, it showed signs of overfitting in this case. The polynomial kernel consistently outperformed the others and was therefore selected.
To mitigate SVMs' sensitivity to multicollinearity, different combinations of highly correlated features were dropped during tuning. The best results were obtained when removing the variables `DB` and `Sgpt` after including their respective derived features. Further attempts at feature engineering—including the use of ratios between correlated features and the addition of cluster-based variables—did not yield performance improvements. Feature importance from a random forest was also considered for variable selection, but results were inconclusive. Ultimately, the most effective strategy involved removing collinear variables.
The final SVM model used a polynomial kernel and a reduced feature set obtained by dropping correlated variables. This makes sense, because as seen in class, more variables would imply unnecessary noise.

*d) **Random Forest**:* Random Forest required careful hyperparameter tuning to prevent overfitting, particularly regarding tree depth and the number of estimators. Despite its feature selection capability, the best performance was achieved by retaining even low-importance variables such as *Female*. The model also exhibited a decrease in F1 when including features like the log-likelihood variable, which strongly influenced tree splits; this was addressed by considering both feature importance and the impact on performance metrics when evaluating feature inclusion.

*e) **Extra Trees Classifier**:* Since Random Forest and Extra Trees use similar feature handling, no new features were added beyond those in the Random Forest analysis. The existing features were sufficient.

Extra Trees is prone to overfitting but was controlled by hyperparameter tuning. Over-regularizing can harm performance because the model already uses high randomness; too much restriction leads to underfitting.

The final Extra Trees model outperformed by little the best Random Forest, likely due to its increased randomization reducing variance and improving generalization.

*f) **Ensembles**:*
We combined different subsets of the six best-performing models, selecting combinations based on confusion matrix analysis to exploit complementary error patterns rather than exhaustively testing all subsets.

Voting
Both hard and soft voting were evaluated across these subsets. Hard voting provided the best overall performance. Complex models tended to make fewer errors on the minority class,

while simpler models exhibited different error patterns. Their combination allowed for balancing these differences, improving generalization.

Stacking

Stacking was performed using logistic regression as the meta-model due to its favorable trade-off between simplicity and performance. All base models were included in the stacking ensemble, and a grid search was conducted to optimize the meta model's hyperparameters. Despite this, stacking did not surpass the best hard voting ensemble in terms of performance, and performed quite badly.

VI. Results

The following table shows the best tuned models obtained, sorted by decreasing Cross-validation F1 Score.

Model	F1 Macro (CV)	F1 Macro (Train)
Voting	0.652	0.699
GNB	0.645	0.654
RF	0.644	0.749
ETC	0.644	0.787
LogReg	0.634	0.640
SVC	0.634	0.702
QDA	0.627	0.688
Stacking	0.512	0.635

TABLE II
F1 TRAINING AND CV SCORES FOR BEST MODELS

Firstly, all the final tuned models achieved similar cross-validation F1 scores, around 60%. Training F1 scores were also close to the CV scores, indicating no severe overfitting. This performance ceiling of 65 % may reflect limitations in the dataset or flaws in our approach, and could potentially be improved with further experimentation.

The highest cross-validation score was obtained by a hard voting classifier combining the best GNB, QDA, ETC, SVC, and Logistic Regression models.

However, the runner-up—Gaussian Naive Bayes—was significantly simpler, with only a 0.007 difference in cross-validation F1.

A dilemma arose when trying to select the best model, as one is much simpler but makes a slightly higher number of mistakes—something that is highly relevant when the purpose is classifying sick patients.

However, the gain from using a much more complex model is so small that it was deemed unnecessary. Additionally, we consider that, in today's society, it would be unwise to rely solely on data-based models to diagnose serious diseases such as liver illness, and so the classification should serve only as a preliminary guide. Thus, we considered the trade-off of simplicity in exchange for a few more mistakes when classifying sick patients to be acceptable. The selected model was therefore the best Gaussian Naive Bayes found, with the following confusion matrix for the training dataset. We can clearly see most patients from each class are correctly classified.

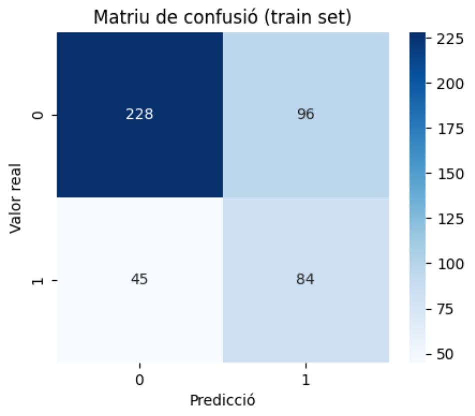

Fig. 3. Confusion matrix of the best model (Gaussian Naive Bayes).

VII. Conclusions

This project provided valuable insights into model selection and evaluation on a clinical dataset with limited sample size and class imbalance. The use of only course-covered algorithms and the time constraints of the competition restricted the exploration of more advanced techniques.

Some methodological limitations were identified: the feature engineering process could have been refined to extract more meaningful information and reduce redundancy; hyperparameter tuning lacked a two-stage refinement; and log-likelihood features were applied without prior scaling, potentially affecting some models like Random Forest. Moreover, only SMOTE was explored for resampling.

In terms of ensembling, efforts focused on complementarity analysis using confusion matrices, but only a few stacking meta-models were tested, and no exhaustive combination evaluation was performed.

After testing all models, the Gaussian Naive Bayes (GNB) emerged as the best trade-off between simplicity and performance. While ensembles offered slight improvements, they introduced excessive complexity and reduced interpretability.

Despite these constraints, the project achieved solid model development and highlighted key differences in algorithm behavior, overfitting tendencies, and feature dependencies. Future work could expand on feature engineering, explore more robust resampling and boosting methods, and develop smarter ensembles—especially given the strong performance already achieved by the simple GNB model.

References

[1] Kankal, S.S., & Kasar, S. (2025). *A Systematic Review on Machine Learning Techniques for Liver Disease Detection*. Biomedical Materials & Devices. https://doi.org/10.1007/s44174-025-00341-1

[2] Livers. (2021). *Liver Disease Detection Using Machine Learning Techniques*. Livers, 1(4), 294–312. https://doi.org/10.3390/livers1040023

[3] Giannini, E.G., Testa, R., & Savarino, V. (2005). *Liver enzyme alteration: a guide for clinicians*. CMAJ, 172(3), 367–379.

[4] Vidal Manzano, J. (2024–2025). *Transparències del curs d'Aprenentatge Automàtic 1*. Grau en Ciència i Enginyeria de Dades, Universitat Politècnica de Catalunya (UPC).