Liver Patient Classification

Universitat Politècnica de Catalunya

Nicolás Jimenez Muñoz and Aina Vila Arbusà
Aprenentatge Automàtic I
Universitat Politècnica de Catalunya
nicolas.jimenez.muñoz@estudiantat.upc.edu, aina.vila.arbusa@estudiantat.upc.edu

***Abstract*—This project addresses liver disease classification using clinical and demographic data from 579 subjects (414 patients and 165 healthy controls) collected from Andhra Pradesh, India. An initial exploratory data analysis was performed to understand the distributions and relationships of the features. Several preprocessing steps were then applied to adapt and transform the data according to the requirements and assumptions of each classification model. To address class imbalance, resampling techniques such as SMOTE or class-weighted learning were employed. A wide range of algorithms were evaluated, including Random Forest, Gaussian Naive Bayes, Support Vector Machines (both linear and non-linear), and Logistic Regression, among others, with the aim of maximizing the F1-score. The final approach integrates feature engineering ensemble methods, showing high effectiveness in medical data classification and providing insights into the most clinically relevant features of liver disease.**

I. Feature Analysis

Since handling missing values and conversion of categorical features into numerical format had already been addressed, we proceeded with further data preprocessing following the initial exploratory data analysis.

A. Outlier Detection

A common preprocessing step in data analysis is the identification and potential removal of outliers. However, in medical datasets, extreme values may correspond to real and clinically relevant conditions rather than measurement errors. In the absence of guidance from medical professionals, we opted to retain these extreme observations. To understand how each feature behaves with respect to health status, we analyzed their distributions using histograms, differentiating between healthy and ill individuals. In addition to classical outlier detection techniques, such as the **Interquartile Range (IQR)** method, we also applied a more advanced model-based approach: the **Local Outlier Factor** (LOF). Despite detecting several potentially outlying observations, we chose not to remove them.

B. Feature Scaling and Transformations

Scaling and transforming variables is crucial, especially for models that are based on distance metrics or assume normally distributed features. Algorithms such as Logistic Regression, Linear and Quadratic Discriminant Analysis, K-Nearest Neighbors, Gaussian Naive Bayes, and Support Vector Machines are sensitive to both the scale and distribution of input variables. Conversely, tree-based models (such as Decision Trees, Random Forests, and Gradient Boosting) are generally invariant to these changes, since they split data based on thresholds rather than distances. An analysis of feature distributions revealed that variables, specifically `TB`, `DB`, `Alkphos`, `Sgpt`, and `Sgot`, were highly skewed. The `A/G Ratio` also exhibited moderate skewness. While the **RobustScaler** is useful in reduce the influence of outliers, it does not address skewness or non-normality.

To address skewness and improve Gaussianity, we applied the **Box-Cox transformation** and, the remaining features (`TP`, `Albumin`) were scaled with **RobustScaler**. Nevertheless, we saw that, after applying the Box-Cox transformation, the distributions of `TB` and `DB` remained far from normal. A logarithmic transformation was ineffective as well and, consequently, we employed a **Quantile Transformation**, which maps the data to a uniform distribution and then to a Gaussian, effectively normalizing the data.

C. Correlation detection

Collinearity can affect the performance and interpretability of linear models, such as Logistic Regression, LDA, and QDA. It can lead to numerical instability and unreliable coefficient estimates. To address this issue, we took two-step approach:

1) **Variance Inflation Factor (VIF):** We computed the VIF for each feature. The variable `Albumin` had an extremely high value and was consequently removed from the dataset to reduce multicollinearity.

$$\text{VIF}_j = \frac{1}{1 - R_j^2}, \tag{1}$$

   where R_j^2 is the coefficient of determination of the regression of predictor j on the other predictors.
2) **Correlation Matrix Analysis:** Even after the removal of `Albumin`, several pairs of features remained highly correlated (e.g., `TB` and `DB`) (Fig. 1). To further reduce the issue, we applied **Principal Component Analysis (PCA)**. By preserving 98% of the total variance, dimensionality was reduced to two principal components.

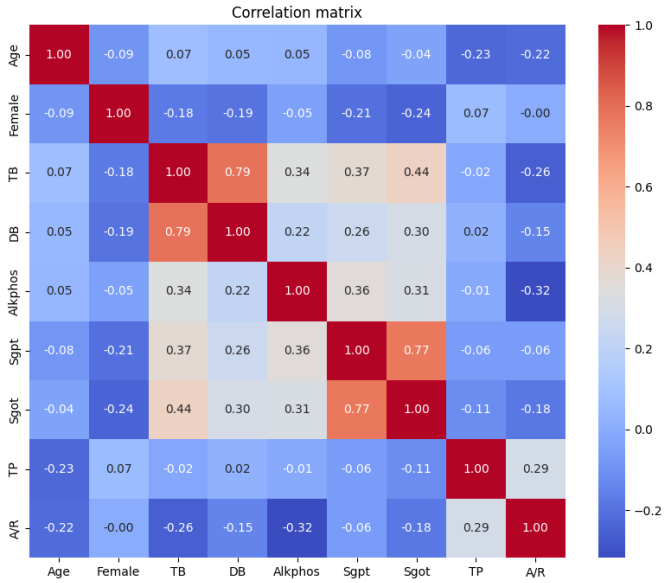
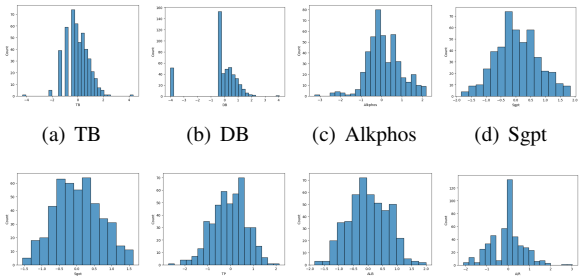


Fig. 1. Correlation matrix without albumin.

D. Data for Model Comparison

Following this initial pre-processing, we prepared four different versions of the dataset to evaluate the impact of various pre-processing strategies on model performance:

- **Box-Cox transformed data:** Only skewed features were transformed using Box-Cox; no scaling or decorrelation.
- **Box-Cox + scaled data:** Box-Cox transformation followed by scaling of all features.
- **Box-Cox + scaled + decorrelated data:** In addition to the previous steps, multicollinearity was addressed.
- **Raw data:** The original dataset with no transformations applied.



(a) TB (b) DB (c) Alkphos (d) Sgpt

(e) Sgot (f) TP (g) ALB (h) A/R

Fig. 2. Feature distributions

II. Classification Methdods

To address the binary classification we evaluated a variety of supervised learning algorithms, including both parametric and non-parametric approaches. The classifiers used were: **Linear and Quadratic Discriminant Analysis (LDA and QDA)**, **K-Nearest Neighbors (KNN)**, **Gaussian Naive Bayes**, **Logistic Regression**, **Linear Regression**, **SVM (Linear and Non-linear** and **Decision Tree**. These were chosen to represent a range of assumptions and capacities to model complex patterns. Linear models are interpretable and robust under normally distributed inputs; tree and kernel based methods were chosen for their flexibility in capturing nonlinear interactions.

A. Ensemble methods

In order to improve predictive performance and reduce model variance, ensemble learning methods were used. These combine predictions of multiple base estimators creating a stronger model. It has been proved that these are particularly advantageous for structured data with moderate dimensionality and complex feature interactions such as in medical data contexts. The considered ensemble approaches are:

- **Random Forest:** Bagging-based method that builds multiple decision trees using randomly resample subsets of features at each split. Predictions are aggregated via majority vote. It reduces variance and is robust to overfitting, outliers and feature scaling.
- **XGBoost:** Gradient boosting implementation that trains trees sequentially, where each new tree focuses on correcting the errors of previous ones. Widely used in tabular predictive modeling.
- **AdaBoost:** Builds an ensemble by training a sequence of weak classifiers, giving more weight to misclassified samples at each iteration. The final prediction is a weighted vote of all models. It is effective at reducing bias but can be sensitive to noisy data.
- **Stacking:** Combines predictions from multiple diverse base models using a meta-model that learns how to best combine them. Often yields superior performance.

B. Data imbalance

Furthermore, given the significant imbalance in class distribution, where the majority class corresponds to patients with liver disease, we implemented several techniques to address the issue.

- **Synthetic Oversampling (SMOTE):** This technique generates new minority-class instances interpolating between existing samples in the feature space.
- **Class Weighting (`class_weight='balanced'`):** Setting this parameter automatically assigns weights inversely proportional to class frequencies in the training set. It increases the penalty for misclassifying minority-class instances, forcing the model to treat both classes more equally.

III. Evaluation Metrics

To analyze the performance of the classification models, we used different evaluation metrics that measure several aspects of predictive quality. The results for each model are summarized in a comparative table, where the main criterion for raninkg is the macro-averaged F1 score, as this was also the official metric used in the Kaggle competition. The selected evaluation metrics are:

- **Accuracy:** The proportion of correctly classified instances out of the total number of samples. Nevertheless,

it has to be taken into consideration that it sometimes can be misleading in imbalanced datasets like ours.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

- **F1 Score (Macro-Averaged):** This is the most important one, since it is the metric used in the Kaggle competition. It provides a balance between precision and recall. It is very informative in class imbalance scenarios with class imbalance, since it considers both false positives and false negatives.

$$F1_{macro} = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{3}$$

- **Precision (Macro-Averaged):** The amount of correctly predicted positive cases among all predicted positives.

$$\text{Precision}_{macro} = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i} \tag{4}$$

- **Recall (Macro-Averaged):** The proportion of actual positive cases that were correctly identified.

$$\text{Recall}_{macro} = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FN_i} \tag{5}$$

Macro-averaging was selected over micro-averaging to prevent the dominant class from influencing the evaluation, ensuring a fair evaluation of model performance across both majority and minority classes.

IV. Experiments

This section describes the experimental methodology applied to evaluate the performance of the classifiers on the dataset. Models were trained using the raw and transformed dataset (after preprocessing), and, where appropriate, experiments were repeated on a decorrelated feature set to validate model assumptions.

Performance was measured using 5-fold cross-validation and evaluated using the metrics previously defined.

Additional improvements were carried out in the 5 better performing models through data resampling, hyperparameter tuning and ensemble methods (check it out in First Model Selection of the code).

A. Basic models

- **Linear Discriminant Analysis (LDA):** The model was trained using the *LinearDiscriminantAnalysis* class from scikit-learn on the transformed training data, both with and without the removal of correlated features, in order to satisfy the model's assumptions. This allowed us to evaluate the impact of feature correlation on model performance.
- **Quadratic Discriminant Analysis (QDA):** Similar to LDA, *QuadraticDiscriminantAnalysis* was applied to both the full and uncorrelated transformed datasets. This experiment was conducted to evaluate whether the increased flexibility of QDA led to improved classification performance.
- **Gaussian Naive Bayes (GNB):** Was evaluated using the transformed dataset and, as the previous ones, with a 5-fold cross-validation. Since the model does not support hyperparameter tuning, its performance was mainly tested across different data representations (with PCA, without PCA).
- **Logistic Regression:** The *LogisticRegressionCV* class was employed for regularization parameter tuning over 20 logarithmically spaced values of C, using 10-fold cross-validation. The optimal value obtained, was used to retrain the final model and, in order to balance date, 'class_weight='balanced'' was also employed. Since logistic regression assumes linear separability and feature independence, experiments were also repeated on the decorrelated dataset.
- **Support Vector Machine (Linear Kernel):** This was trained using *SVC*, with a linear kernel and 'class_weight='balanced''.
- **Support Vector Machine (RBF Kernel):** In this case, the RBF kernel was employed to capture non-linear relationships. A grid search was used to tune C and γ using 5-fold cross-validation with macro-averaged F1 score. Then the best model was validated on a separate validation set.
- **K-Nearest Neighbors (KNN):** A *GridSearchCV* procedure was first used to identify the optimal number of neighbors (*k*) and distance metric, using a 5-fold cross-validation. The configuration that achieved the best scores, excluding $k = 1$ to reduce the risk of overfitting, was $k = 7$ with the *Euclidean* distance metric. These were used in the final model.
- **Decision Tree:** A *DecisionTreeClassifier* was trained on the raw training dataset using 'class_weight='balanced'' to address class imbalance. A hyperparameter grid search was conducted over tree depth, minimum samples split, minimum leaf size, and maximum features. The optimized model was validated on a separate set.

B. Ensemble methods

- **Random Forest:** A *RandomForestClassifier* was initially trained with 100 trees using 'class_weight='balanced'' to correct for imbalance. The model's OOB score gave us an internal estimate of generalization error. In order to insist on data balancing, SMOTE was applied too. A grid search was performed to tune the number of estimators, maximum depth, split criteria, and class weighting. The best model was evaluated using macro-averaged F1 score and validated on a held-out set.
- **Gradient Boosting:** A *GradientBoostingClassifier* was applied to the raw training dataset. A grid search was used to tune critical parameters including the number of estimators, learning rate, maximum depth, and minimum samples for split and leaf nodes. The best configuration

was then trained on the complete training set and evaluated on the validation set. This model was enhanced by resampling with SMOTE and repeating a better hyperparameter tuning.

- **AdaBoost (with SMOTE):** An *AdaBoostClassifier* using shallow decision trees as weak learners was trained on SMOTE-resampled data. A 5-fold *stratifiedKFold* cross-validation procedure was used to adjust the number of estimators and the learning rate. The final configuration was selected based on the highest macro-averaged F1 score.
- **Stacking (Gradient Boosting + AdaBoost):** This stacking experiment combined Gradient Boosting and AdaBoost as base learners, both trained on raw SMOTE-balanced features. A *Logistic Regression* model was used as the meta-learner. This setup allowed for the integration of various decision boundaries in a unified model.
- **Stacking 2 (GNB + SVM + Random Forest):** A second stacking ensemble combined Gaussian Naive Bayes, Linear SVM and a tuned Random Forest model. All of them were trained on normalized and SMOTE-balanced data. Logistic Regression served again as the meta-learner. The objective for this setup was to combine generative and discriminative strengths.
- **Stacking 3 (GNB + Random Forest):** To test the effect of model simplicity and reduce the risk of overfitting, a lightweight stacking variant was wet up using only GNB and Random Forest was evaluated. This configuration was particularly useful to assess the advantages of including SVMs in the ensemble. As with previous setups, SMOTE was applied before training, and predictions were passed to a Logistic Regression meta-learner.

V. Results

Model	Accuracy	F1 Macro	Precision Macro	Recall Macro
Random Forest (SMOTE)	0.623656	0.653664	0.604342	0.625421
Log Regression	0.672973	0.653237	0.665689	0.702695
Stacking Ensemble (GNB + SVM + RF)	0.655914	0.651531	0.610837	0.626263
SVM (Linear)	0.662162	0.649857	0.680543	0.718149
Stacking Ensemble (GNB + RF)	0.623656	0.646917	0.604342	0.625421
Gradient Boosting (SMOTE)	0.612903	0.644140	0.573118	0.585017
Random Forest (tuned)	0.666667	0.638405	0.640094	0.666667
Log Reg Uncorrelated Data	0.667568	0.635006	0.638573	0.661545
Stacking Ensemble (GB + AdaBoost)	0.612903	0.634083	0.573118	0.585017
GNB	0.670270	0.629992	0.629791	0.649057
Log Reg + SMOTE	0.655914	0.624242	0.625356	0.648148
SVM (Linear + SMOTE)	0.645161	0.624127	0.634276	0.662458
LDA	0.743243	0.616191	0.707419	0.610872
KNN1	0.678378	0.610775	0.609373	0.614465
QDA	0.651351	0.589366	0.587825	0.595597
GaussianNB (SMOTE)	0.602151	0.582859	0.599907	0.621212
Tuned SVM	0.612903	0.582752	0.589353	0.606902
Decision Tree	0.648649	0.580996	0.579120	0.585085
AdaBoost + SMOTE	0.602151	0.573764	0.582633	0.599327
Random Forest	0.708108	0.573233	0.612518	0.571968
Decision Tree Tuned 2	0.591398	0.569444	0.584722	0.602694
SVM (Non-linear)	0.621622	0.562950	0.562861	0.569093
Gradient Boosting	0.655914	0.550725	0.558532	0.549663
AdaBoost	0.666667	0.545626	0.561111	0.546296
KNN2	0.667568	0.545097	0.557160	0.546541
LDA Uncorrelated	0.694595	0.498462	0.585482	0.522282
Decision Tree Tuned	0.516129	0.481605	0.495798	0.494949
QDA Uncorrelated Data	0.705405	0.473766	0.569795	0.515454

TABLE I
Model comparison based on Accuracy, F1 Macro, Precision Macro and Recall Macro.

A. Parametrized Gaussian models

- **LDA and QDA:** QDA outperforms LDA in terms of Recall Macro, but performs worse on the other metrics. Although LDA achieves higher accuracy, the highest among all models, this metric is less informative given the class imbalance; therefore, F1-score provides a more reliable performance measure. Both models show a notable drop in F1 Macro compared to accuracy, indicating that their performance is uneven across classes (likely worse on the minority class). Dimensionality reduction, aimed at addressing uncorrelated features, further deteriorated the results.
- **GNB:** This model achieves a comparatively high F1 Macro score, highlighting its ability to maintain a balanced trade-off between precision and recall across classes. Despite its simplicity and the strong assumption of feature independence, GNB performs competitively, which suggests that it can still be effective even when its assumptions are not fully met.

B. Unparametrized models

- **KNN1 and KNN2:** KNN1 (with k = 1) outperforms KNN2 (k = 7) in terms of F1 Macro on the validation set. However, using k = 1 often leads to overfitting by modeling noise in the training data. This hypothesis was confirmed when evaluating the model on the Kaggle test set, where KNN1 exhibited a significant drop in performance compared to KNN2.

C. Unknown density function models

- **Logistic Regression:** Show to have great F1 Macro, although performance slightly decreases when trained on the uncorrelated dataset. This behavior is expected: removing correlated variables often results in the loss of useful information relevant to classification.
- **SVM:** The linear SVM outperformed the Non-linear version significantly, suggesting that data is better separated by a linear decision boundary. We then performed the hyperparameter tuning on the non-liner SVM. While it lead to some improvement in evaluation metrics, it still showed relatively poor results compared to the linear model.
- **Decision Tree:** The untuned version achieved better performance compared to both tuned versions. Specifically, its F1 Macro was higher, despite not benefiting from a tuned search of hyperparameters. Since Decision Trees can already capture complex patterns without heavy optimization, in some situations, the gain is marginal or even negative because of the bias-variance trade-off.

D. Ensemble methods

- **Random Forest:** The basic yielded an Out-of-Bag (OOB) score of 0.7108, corresponding to an OOB error of 0.2892. This estimate suggests that the model generalizes reasonably well to unseen data. After hyperparameter tuning, we observed a clear improvement in the F1 Macro

score. Interestingly, this tuning process also led to a slight reduction in Accuracy. This is expected behavior when working with imbalanced datasets: accuracy tends to be biased toward the majority class, while F1 Macro gives equal weight to each class, providing a fairer evaluation. Therefore, the observed trade-off reflects a shift toward a more balanced classifier that avoids overfitting to dominant labels. The biggest improvement is achieved when using SMOTE for data balancing.

- **Gradient Boosting:** The basic model performed poorly in F1 Macro score. Nevertheless, after tuning and taking into consideration data imbalance applying SMOTE, the model achieved a substantial increase in F1 Macro. As in the previous case, this increase came with a moderate reduction in overall **accuracy**, again reflecting the expected trade-off when shifting the model's focus away from the majority class.
- **AdaBoost:** This underperformed across evaluation metrics, both before and after applying SMOTE. Its sensitivity to noisy data and its limited capacity to model complex relationships may explain its relatively poor results in comparison to other ensemble methods.
- **Stacking with Raw Data:** We evaluated multiple combinations of classifiers using raw input data. Notably, the combination of **AdaBoost and Gradient Boosting** yielded the best overall results. Despite AdaBoost's individual poor performance, using it in the ensemble appears to provide complementary decision boundaries that Gradient Boosting alone does not capture. This likely enhanced the diversity of the base models, which is a known factor in improving ensemble performance. The model outperformed other combinations on the private Kaggle submissions. This supports the idea that carefully selected, even individually weak, learners can boost ensemble performance when their errors are uncorrelated.
- **Stacking with Normalized Data:** The stacking ensemble combining Gaussian Naive Bayes (GNB), Random Forest (RF), and Support Vector Machine (SVM) demonstrates superior performance compared to the ensemble stacking only GNB and RF. Specifically, the inclusion of SVM leads to higher F1 Macro scores, indicating a better balance between precision and recall across all classes. This improvement is also reflected in the slightly increased accuracy and recall metrics. It can be attributed to the increased diversity of the base models in the ensemble. While GNB provides a probabilistic approach and RF contributes a strong tree-based ensemble, SVM adds a distinct linear decision boundary, enriching the model's ability to capture complex patterns in the data. This diversity enables stacking method to generalize better and reduces the likelihood of overfitting to specific data characteristics.

VI. Conclusions

Among the tested models, **Random Forest** achieved the best performance metrics in the local validation sets, showing robustness and reliable predictive power. However, despite its strong local results, on the Kaggle leaderboard other methods showed to perform better, suggesting that local validation alone may not fully capture the model's generalization capability on unseen data.

Stacking ensembles combining **Gaussian Naive Bayes (GNB)**, **Random Forest (RF)**, and **Support Vector Machines (SVM)** showed promising results during local cross-validation. Surprisingly, these performed worse on the Kaggle platform, indicating potential overfitting or discrepancies between the local validation data distribution and the test data.

Gradient Boosting models, while not achieving the top scores locally, they ranked very well on Kaggle. This suggests that it may generalize better to the test set despite showing less impressive local metrics. This highlights the importance of evaluating multiple metrics and external validation.

Interestingly, **Gaussian Naive Bayes** performed badly according to local evaluation metrics but achieved relatively good results on Kaggle. This emphasizes that some classifiers might capture patterns not evident in local validation, especially in imbalanced and complex datasets.

Overall, this project highlights the critical need for careful **model selection**, **hyperparameter tuning**, and **multiple evaluation strategies**. Although local metrics provide valuable information, the final performance in external datasets can differ significantly. Ensemble methods, particularly stacking, can be promising but require cautious validation to avoid overfitting.

Future work could focus on improving generalization through better cross-validation strategies, feature engineering, and exploring additional ensemble combinations.

References

[1] J. Vidal, *Chapter 4: Non-parametric classifiers*, Lecture notes, Universitat Politècnica de Catalunya, 2025.

[2] J. Vidal, *Chapter 5: Logistic regression*, Lecture notes, Universitat Politècnica de Catalunya, 2025.

[3] J. Vidal, *Chapter 6: Support vector machines*, Lecture notes, Universitat Politècnica de Catalunya, 2025.

[4] J. Vidal, *Chapter 7: Classification and regression trees*, Lecture notes, Universitat Politècnica de Catalunya, 2025.

[5] J. Vidal, *Chapter 8: Ensemble methods*, Lecture notes, Universitat Politècnica de Catalunya, 2025.

[6] J. Vidal, *Chapter 9: Unsupervised learning*, Lecture notes, Universitat Politècnica de Catalunya, 2025.

[7] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-Interscience, 2004.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.