Liver Patient Prediction Using Machine Learning

1st David González Acén
Polytechnic University of Catalonia (UPC)
Barcelona, Spain
david.gonzalez.acen@estudiantat.upc.edu

2nd Angel Morales Cirera
Polytechnic University of Catalonia (UPC)
Barcelona, Spain
angel.morales@estudiantat.upc.edu

***Abstract*—This project applies machine learning to a medical classification task. The goal is to predict whether a person suffers from a liver disease or not, based on several clinical measurements. We work with a dataset that contains information about several liver-related indicators for a number of patients. We try different classification models, including Discriminant Analysis, Naive Bayes, Logistic Regression, K-Nearest Neighbors, Random Forest, and Support Vector Machines (SVM).**

After testing and comparing them using F1-score as the main evaluation metric, we find many models with a very good performance, but we consider the Support Vector Machine to be the best. To obtain the best possible configuration, we tuned carefully its hyperparameters using a grid-search with cross-validation, and finally tested it on unseen data. The results show that SVM has a very good performance, especially when the data is scaled properly and we balance the classes correctly. This confirms that SVM is a good option for this kind of medical prediction problem. Overall, the project shows how machine learning can be very helpful in real world health problems.

I. Introduction

This project should be considered under the context of a competition between different groups in the Kaggle InClass platform. The main goal of this project is to find a machine learning (ML) model capable of achieving a high score in this competition, which compares each team's model based on the F1-score, a metric computed from the recall and precision, that takes into account class imbalance.

This competition presents a realistic problem: predicting liver disease using machine learning methods, given a dataset with 579 observations: 414 belong to patients with liver problems and 165 to healthy subjects. Each observation contains 10 features, most of them related to liver function. These include age, gender, and several blood test indicators like Total Bilirubin, Alkaline Phosphotase, and Albumin/Globulin ratio.

The dataset was already preprocessed to remove rows with missing values and to convert categorical variables into numerical ones. Our task is to build a classifier that can tell if a patient is sick or healthy based on these features.

To do this, we test different models and compare them, aiming to find the model that performs best. Our tools have been Scikit-Learn, a high-level module for Python that contains a diversity of ML algorithms, among other things.

II. Feature analysis

A previous study on the field of hepatic diseases has been carried out in order to have an initial idea of the effects of the variables, along with the standard values for healthy patients. Also, normality and outliers have been checked, and the pairplot matrix has been analysed.

In general, variables do not follow a normal distribution, and the Box-Cox transformation has been applied before fitting any models that assume normality. We have discovered that liver disease affects males twice as often as females [1]. In terms of bilirubin, total and direct bilirubin are expected to be found at doses less than 0.3 and 2 mg/dL in adults, respectively, and higher values may be related to liver disease [2]. Alkaline phosphotase is expected to be lower than 650 for children, and less than 270 for adults. Value vary between ages and gender, and if too high may be related to liver disease [3]. Sgpt and sgot have normal values that vary across laboratories, but can be approximated by the range (5, 60) units/liter for both. Very high values are closely related to liver disease [4]. Lastly, proteins (tp), albumin (alb), and albumin-globulin ratio (a/r) can also be useful in detecting liver disease, but there is not a clear distinction between normal and dangerous values. Unusually low values may be related to liver disease [5].

There are some variables with a high linear correlation, and we will have to take this into account for some of our models.

III. Classification methods

Due to the size of the dataset being quite little, the best option is using shallow classification algorithms (supervised learning). We have considered some generative classifiers: linear and quadratic discriminant analysis and Naive Bayes; K nearest neighbors (KNN), discriminative classifiers: logistic regression, support vector machine (SVM) and decision trees.

Those methods have been trained using K-fold cross-validation to select hyper-parameters as well as to estimate the generalization performance. We will get into details on the experiments section. Also, we have considered some models built using ensembles: bagging (and random forests), boosting and stacking.

IV. Evaluation Metrics

Since the competition's leaderboard is based on the F1-score, it has been our main evaluation metric. Additionally, because liver-patients are labeled as 0 (negative class) and healthy individuals as 1 (positive class), we have also taken into account the specificity, which aims to measure the relative number of false positives. In this case, a false positive means

Identify applicable funding agency here. If none, delete this.

classifying an ill subject as a healthy one, which would be a dangerous situation in a real scenario.

Defining:

- TP: True Positives — healthy subjects correctly classified.
- TN: True Negatives — liver-patients correctly classified.
- FP: False Positives — liver-patients incorrectly classified as healthy.
- FN: False Negatives — healthy subjects incorrectly classified as liver-patients.

Precision, shown in Equation 1, measures the proportion of predicted positives that are actually correct:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

Recall, or sensitivity, quantifies the proportion of actual positives that are correctly identified (Equation 2):

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

Specificity, defined in Equation 3, captures the proportion of actual negatives correctly classified:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3}$$

The F1-score is the harmonic mean of precision and recall (Equation 4), providing a balanced measure when both are important:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

The F1-score can also be computed directly from TP, FP, and FN values using Equation 5:

$$\text{F1 Score} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{5}$$

V. Experiments

A. Dataset splits

We aim to select one final model. These means that we will need to use some of the data to compare the models, so we will need:

- A training split. For each model we will divide this training split further, because we will use some type of validation for every model that requires it. The training phase will allow to get a training and in some cases a validation error, capable to predict the test error.
- A ”comparison” split. Similar to a test split, but we will use it to select the model. This means we are using this information and is not a test set, properly said. Nevertheless, models will not use this information when they are fitted.
- A test set. This will be the public set provided at the Kaggle page. To avoid overfitting, we won't use the information provided by the test set, unless extreme cases (such as an incomprehensible drop in the F1-score). The final private test set will then be the real and only test set.

B. Training performance

For some models, it is interesting to be able to compare different versions before checking the performance on the ”comparison” set. In order to do this, we have defined a 5-fold cross-validation function that allows to have a performance measure on the training set, but that is not as optimistic as the normal training performance.

This cross-validation performance is an estimation of the performance on a balanced dataset, but because the dataset is not balanced and we aim to get the highest F-score, it has been necessary to use the ”comparison” set to get the proper performance of the selected models.

When talking about any performance measure, we will talk about the measure on the ”comparison” set, unless we specify that we are talking about cross-validated training performance (or raw training performance, without cross-validating).

C. Generative Classifiers

We started by training three generative classifiers: Gaussian Naive Bayes, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA). Naive Bayes assumes conditional independence between features, which is unrealistic given the previously observed correlations in our dataset. LDA assumes that all classes share the same covariance matrix, which may also be problematic. QDA relaxes this assumption but requires more data to estimate the covariance of each class accurately.

After training the three models, we obtain a F1-Score of 0.576 for the GNB, 0.300 for the LDA and 0.531 for the QDA.

These poor results, particularly for LDA, are largely due to class imbalance. To adress this problem, we can use SMOTE, a technique that generates synthetic samples for the smaller class. After applying SMOTE, the F1-scores improved to 0.598 on LDA and 0.559 for QDA, although the Naive Bayes dropped to 0.560. This confirms that oversampling significantly benefits discriminant analysis models, reducing their tendency to favour the majority class.

We have checked the parameters of the LDA, and it mostly agrees with our prior knowledge on the variables: the coefficients suggest that an individual has more chances of being a patient if its a female, and if age, total bilirubin, alkaline phosphotase, sgpt, sgot, tp and a/r get high. It only reduces the chances when the direct bilirubin and the albumin are high. Note that because of the correlation between direct and total bilirubin, they compensate each other, but higher direct bilirubin does not mean a lower chance of being a liver patient.

D. K Nearest Neighbours

To be able to select a proper K without overfitting on the training data, we have divided the training set into ”tr” and ”val” sets. For a range of values of K, we have fitted the model on the ”tr” set and used the ”val” to get performance measures. Performance tops at 1 neighbor, and decreases when the K increases. This already suggests that KNN is overfitting, and that it is not suitable for our problem. Anyways, we have

checked performance on the "comparison" set for 4 different Ks: 1, 3, 7 and 13, and get low F-scores: 0.375, 0.389, 0.442, 0.364. It seems that we will not use KNN as our final model.

E. Logistic regression

We have trained a logistic regression with different possible hyperparameters using K-fold cross-validation, directly with an implemented Python class. We also have compared two different optimization solvers and two different approaches: with or without the scaling data. After using the cross-validated training error to discard two of them, the resulting models had a "comparison" F-score of 0.585 for the logistic regression without scaling the data, and 0.567 having scaled the data.

F. Support Vector Machine

SVMs are known for being powerful classifiers that can separate classes clearly, even in complex scenarios. They have several hyperparameters that affect their performance. One of the most important is the kernel type, and we chose the Gaussian (RBF) kernel because it is very flexible and often performs better than a linear kernel when the data is complex and not linearly separable. We also tested the linear kernel, but its performance was significantly worse.

Another key hyperparameter is C, which controls how much we penalize training errors. A high C tries to fit all training examples correctly, but can cause overfitting. A lower C allows more margin violations and usually generalizes better.

Gamma is a parameter specific to the RBF kernel, and it defines how much influence each training example has. A low gamma results in a smoother and simpler decision boundary, while a high gamma creates more complex boundaries.

Feature scaling is also crucial for SVMs because the kernel relies on distance calculations. We used the StandardScaler (instead of MinMaxScaler), since it centers and normalizes the data, which usually works better with the RBF kernel. Therefore, C and gamma are the most difficult variables to tune. We tested 20 exponentially spaced values for each one of the, C from 0.01 to 100 and gamma from 0.001 to 5. This gave a total of 400 possible configurations of the model. For each one, we ran 10 repetitions of 5-fold cross-validation, and averaged the resulting F1-scores.

We visualized the results using a 2D heatmap, which helped us understand how performance changed across the hyperparameter space. Since the best results were spread across a cluster rather than having a single clear maximum, we selected the top 5 configurations with the highest F1-scores and computed the geometric mean of their C and gamma values to obtain the optimal configuration.

Finally, we retrained the SVM using all the training data and the optimal hyperparameters, and used it to predict the comparison set. This resulted in a F1-Score of 0.579.

G. Decision trees, random forests and ExtraTrees

We began by training a single decision tree, which resulted in a low F1-score of 0.366. Decision trees, while easy to interpret, are often too simplistic and prone to overfitting, making them insufficient on their own for most real-world classification tasks. To address this, we moved on to a Random Forest, which is an ensemble method that builds multiple decision trees on random subsets of the data and features, and aggregates their predictions to improve generalization and reduce variance.

The Random Forest model has several important hyperparameters, among which the most important are: n_estimators (number of trees), max_depth (maximum depth of each tree), and min_samples_split (minimum number of samples required to split an internal node). Similar to our approach with SVMs, we performed a grid search to train multiple models with different combinations of these hyperparameters. In total, we evaluated 96 different configurations, and selected the one that performed best on its cross validation. Finally, we trained a last model with the optimal configuration with all the training data and tested it on the comparison set. The resulting model achieved an F1-score of 0.584.

Finally, we also experimented with an ExtraTreesClassifier, which is conceptually similar to a Random Forest but introduces more randomness by choosing split thresholds at random rather than selecting the best one. This often leads to faster training and, in some cases, better generalization. In this case, we didn't improve the score.

H. Other ensembles

We have first tried stacking. The class StackingClassifier works why training some base estimators, which have been the previously seen models, and combining their predictions with a meta-learner. There are a lot of different possible combinations, so we have only tested some, and reached a maximum F-score of 0.590, which is not bad, but not enough so that a model this comlpex is worth it. This performance was got by using the gaussian Naive Bayes and SVM as base models, and another support vector machine as meta model.

Then, we have tried boosting. We have used the given AdaBoost function, with a wide range of estimators, and compared their cross-validated training F-score. The top 3 number of estimators are 43, 41 and 1, which basicalally tells us that AdaBoost has no real use, as using 1 estimator gets a similar performance as 43. We have chosen an arbitray number to test on the "comparison" set and got a F-score of 0.517.

VI. Results

We saved the F-score and the specificity of the best representatives of each model, and they can be seen in the following table.

The LDA, trained on the oversampled set, seems to be the best model, as it haves the most F-score and specificity. It is followed by some stacking models, the logistic regressor, the random forest, the support vector machine and the gaussian Naive Bayes.

Then, the QDA, the AdaBoost, KNN and a single decision tree follow up, along some of other versions of seen models (such as logistic regressor with scaled data).

Model	F-score	Specificity
LDA_smote	0.598425	0.598131
Stacking	0.589928	0.514019
Stacking_2	0.588235	0.532710
logreg_liblinear	0.584615	0.570093
RandomForest	0.584071	0.682243
Stacking_1	0.582090	0.542056
SVM_best	0.579310	0.467290
GaussianNB_raw	0.576271	0.644860
logreg_liblinear_scaled	0.566929	0.579439
GaussianNB_smote	0.560000	0.588785
QDA_smote	0.559322	0.635514
QDA_raw	0.530612	0.757009
AdaBoost	0.517241	0.224299
7NN_smote	0.442478	0.607477
ExtraTrees	0.416667	0.897196
3NN_smote	0.388889	0.616822
1NN_smote	0.375000	0.700935
DecisionTree	0.365854	0.803738
13NN_smote	0.363636	0.588785
LDA_raw	0.289855	0.878505

TABLE I
F-SCORE AND SPECIFICITY FOR DIFFERENT MODELS.

This leads to the last task: submitting a prediction on the test dataset to the Kaggle competition, which will reveal another F-score measurement on unseen data.

As the LDA is the best performing model, and its generalization performance has been checked both cross-validating on a balanced set, and also testing on the "comparison" set, the LDA has been the first model used to make predictions.

But surprisingly, the LDA reveals a sudden drop to a 0.4 of F-score. This implies that, or the overall process that we have developed has something wrong, or the public test set on Kaggle is slightly imbalanced, but with the opposite class being the dominant one (much more healthy subjects than liver-patients).

In order to understand what happens, we have chosen to make a prediction with the support vector machine, a model which we have carefully tuned to the data and is considered to be a strong classifier.

Now, we manage to get a 0.623 F-score on the public test set, which makes much more sense. So, in the end, we have kept the support vector machine as the chosen model for the problem.

VII. CONCLUSIONS

The SVM is known for being able to create flexible and complex decision boundaries even when the data is not linearly separable or has overlapping classes. Therefore, it is not strange that it has been the chosen model for such a complex task with little data.

The problem of liver-patient classification is quite hard, and it is very important that no ill-subjects are send home without the required professional inspection, as the risk of this could be as high as the life of a person. ML algorithms have to be trained with a lot in mind, in order to have a model capable of being generalized to unseen data, and always considering the consequences of applying in a real scenario a model that is not fully ready to develop its task.

REFERENCES

[1] Cleveland Clinic, "Liver Disease," Cleveland Clinic, [Online]. Available: https://my.clevelandclinic.org/health/diseases/17179-liver-disease. [Accessed: Jun. 20, 2025].

[2] Mayo Clinic, "Bilirrubina: prueba," Mayo Clinic, [Online]. Available: https://www.mayoclinic.org/es/tests-procedures/bilirubin/about/pac-20393041. [Accessed: Jun. 20, 2025].

[3] WebMD, "Prueba: fosfatasa alcalina," WebMD, [Online]. Available: https://www.webmd.com/es/digestive-disorders/prueba-fosfatasa-alcalina. [Accessed: Jun. 20, 2025].

[4] S. K. Gupta and R. A. Malhotra, "Alanine Aminotransferase (ALT)," in *StatPearls*, StatPearls Publishing, Treasure Island (FL), 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK559278/. [Accessed: Jun. 20, 2025].

[5] MedlinePlus, "Proteínas totales y relación albúmina/globulina (A/G)," MedlinePlus en español, [Online]. Available: https://medlineplus.gov/spanish/pruebas-de-laboratorio/proteinas-totales-y-relacion-albumina-globulinaa-g/. [Accessed: Jun. 20, 2025].