Identification of Liver Diseases: Can Machine Learning Algorithms Reduce Burden on Doctors?

Claudio Rota, 816050
c.rota30@campus.unimib.it

Simone Paolo Mottadelli, 820786
s.mottadelli2@campus.unimib.it

1 INTRODUCTION

In India, patients with liver diseases have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. This has led to an extreme increase of work for doctors, who have to visit a lot of patients and analyze a lot of blood test results to diagnose whether these patients have liver diseases or not.
To address this problem, this work sets the goal of studying the performances of different classification models to understand whether machine learning can help reduce the burden on doctors.
This report is structured as follows: Section 2 presents the dataset used, Section 3 describes the preprocessing steps that we applied to make the data more suitable for inducing the classification models, Section 4 describes the classification models we selected to solve our problem, Section 5 shows the results obtained by the selected models and, finally, Section 6 concludes this report by briefly summarizing the work presented in the previous sections.

2 DATASET DESCRIPTION

The dataset used for this study contains 583 patient records collected from North East of Andhra Pradesh (India): 416 of them refer to patients with liver diseases, while the remaining 167 refer to patients without liver diseases. It is available at https://www.kaggle.com/uciml/indian-liver-patient-records. Overall, the dataset consists of the following 11 attributes:

- *Age* (numeric): age of the patient;
- *Gender* (categorical): sex of the patient;
- *Total Bilirubin* (numeric): Total Billirubin in mg/dL;
- *Direct Bilirubin* (numeric): Conjugated Billirubin in mg/dL;
- *Alkaline Phosphotase* (numeric): Alkaline Phosphotase in IU/L;
- *Alamine Aminotransferase* (numeric): Alamine Aminotransferase in IU/L;
- *Aspartate Aminotransferase* (numeric): Aspartate Aminotransferase in IU/L;
- *Total Proteins* (numeric): Total Proteins g/dL;
- *Albumin* (numeric): Albumin in g/dL;
- *Albumin and Globulin Ratio* (numeric): the ratio between Albumin and Globulin;
- *Class* (categorical): whether the patient has any liver disease or not.

While the attributes *Age* and *Gender* represent the age and the sex of the patients, the other attributes refer to values obtainable with specific blood tests.
The *Class* attribute can assume two values: "yes" if the patient has a liver disease, "no" otherwise.
As shown in Figure 1, the dataset is unbalanced: 71.4% of the patients has a liver disease, while the remaining 28.6% does not have any liver disease.

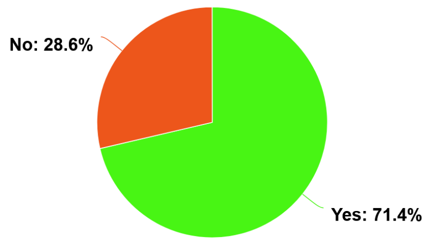

Figure 1: Class distribution

Moreover, Tables 1 and 2 report some statistics related to the numerical and categorical attributes, respectively. As we can see, the majority of the records do not present any missing value, in fact, just 4 out of 583 records have a missing value for the *Albumin and Globulin Ratio* attribute. From the statistics computed on the numerical attributes, we can observe that the ranges of values assumed by the attributes are of very different amplitude. For example, *Albumin* assumes values in $[0.9, 5.5]$, while *Aspartate Aminotransferase* assumes values in $[10, 4929]$. Furthermore, by observing the values reported in Table 1 concerning the skewness and the kurtosis, *Total Bilirubin*, *Direct Bilirubin*, *Alkaline Phosphotase*, *Alamine Aminotransferase* and *Aspartate Aminotransferase* present a value distribution concentrated in correspondence of their minimum value and, thus, have a very long tail to the right. This can be explained because both their skewness and kurtosis assume large positive values.

Attribute	Mean	Median	Standard deviation	Min	Max	N. Missing	Skewness	Kurtosis
Age	44.75	45	16.19	4	90	0	-0.03	-0.56
Total Bilirubin	3.3	1	6.21	0.4	75	0	4.91	37.16
Direct Bilirubin	1.49	0.3	2.81	0.1	19.7	0	3.12	11.35
Alkaline Phosphotase	290.58	208	242.94	63	2110	0	3.77	17.75
Alamine Aminotransferase	80.71	35	182.62	10	2000	0	6.55	50.58
Aspartate Aminotransferase	109.91	42	288.92	10	4929	0	10.55	150.92
Total Proteins	6.48	6.6	1.09	2.7	9.6	0	-0.29	0.23
Albumin	3.14	3.1	0.8	0.9	5.5	0	-0.04	-0.39
Albumin and Globulin Ratio	0.95	0.93	0.32	0.3	2.8	4	0.99	3.28

Table 1: Numerical attribute statistics

Attribute	Frequency	N. Missing
Gender	Male: 441, Female:142	0
Class	Yes:416, No:167	0

Table 2: Categorical attribute statistics

As regards the age of the patients, the average is 45, but children and elderly people are also present. Finally, concerning the *Gender* attribute, we can notice that there is a great imbalance towards the male sex, because 76% of the patients are male and just 24% of the patients are female.

3 PREPROCESSING

From the dataset exploration described in Section 2, we noticed some issues that should generally be solved before starting dealing with classification models: some patient records presented missing values, some attributes had right-skewed distributions and their values were on different scales and the class distribution was unbalanced. To address these issues, we applied different preprocessing steps.

3.1 Variable Transformation

As already mentioned in Section 2, since *Total Bilirubin*, *Direct Bilirubin*, *Alkaline Phosphotase*, *Alamine Aminotransferase* and *Aspartate Aminotransferase* have a right-skewed distribution, we made the decision to apply a logarithmic transformation to these attributes in order to reduce both their skewness and kurtosis [4].
The results of these transformations are reported in Table 3 and, as we can see, the values of the skewness and the kurtosis have considerably decreased.

Attribute	Mean	Median	Standard deviation	Min	Max	Skewness	Kurtosis
Total Bilirubin	0.46	0	1.02	0.46	4.32	1.32	0.92
Direct Bilirubin	-0.65	-1.2	1.33	-2.3	2.98	0.83	-0.28
Alkaline Phosphotase	5.49	5.34	0.53	4.14	7.65	2.28	1.32
Alamine Aminotransferase	3.75	3.56	0.9	2.3	7.6	1.43	2.63
Aspartate Aminotransferase	3.96	3.74	1	2.3	8.5	1.19	1.6

Table 3: Numerical attribute statistics after the logarithmic transformations

3.2 Missing Value Imputation

The dataset contains just 4 records with a missing value for the *Albumin and Globulin Ratio* attribute.
Since the dataset is quite small, we decided not to further reduce its dimension, but to use a missing value imputation technique based on linear regression [12] to replace the missing values of the *Albumin and Globulin Ratio* attribute with concrete values. We computed the linear correlation coefficients among all the numerical attributes and the results are shown in the correlation matrix in Figure 2. As we can see, *Albumin* is the most correlated attribute with *Albumin and Globulin Ratio*, having a linear correlation coefficient equal to 0.69.
In addition, since the correlation between these attributes might be influenced by the health state of the patient, we also tried to

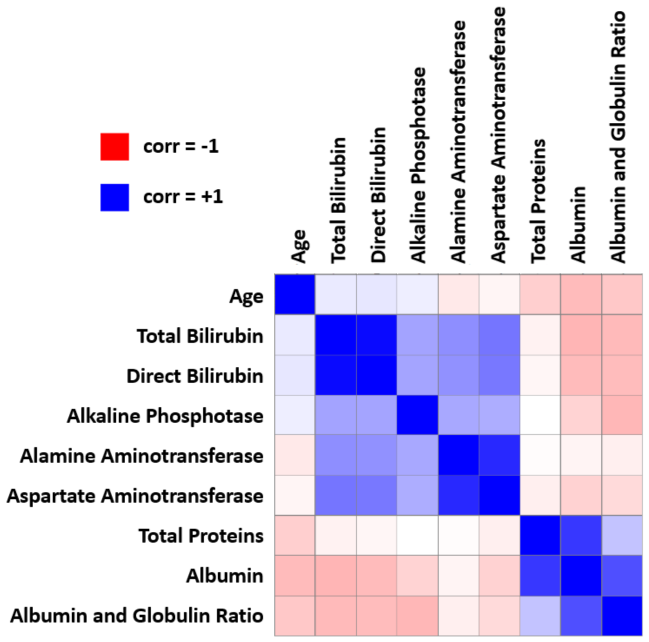

Figure 2: Correlation matrix on the entire dataset

split the dataset into two partitions according to the class attribute to check whether there existed any differences. Even in this case, the most correlated attribute with *Albumin and Globulin Ratio* was *Albumin*, having a linear correlation coefficient equal to 0.74 for the patients without liver diseases and 0.66 for patients with liver diseases. For this reason, since there were no significant differences between the linear correlation coefficient considering the entire dataset and the linear correlation coefficients computed by dividing the dataset according to the class attribute, we induced a linear regression model on the entire dataset, using *Albumin* as independent variable and *Albumin and Globulin Ratio* as dependent variable. Then, we replaced the missing values with the values inferred by the model, whose equation is the following:

$$Albumin\ and\ Globulin\ Ratio = 0.28 \times Albumin + 0.08$$

.

3.3 Normalization

Despite the logarithmic transformations, described in Subsection 3.1, considerably narrowed the range of values assumed by some attributes, namely *Total Bilirubin*, *Direct Bilirubin*, *Alkaline Phosphotase*, *Alamine Aminotransferase* and *Aspartate Aminotransferase*, the ranges of values of the attributes were still on different scales. As described in Subsection 3.4, since the dataset is unbalanced with respect to the *class* attribute and we wanted to apply a class balancing technique that exploits some distance measures, we decided to normalize the data before dealing with the unbalanced class problem.
We applied the Z-Score Normalization using the following formula:

$$Z = \frac{X - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation. Table 4 shows the statistics of each numerical attribute after the normalization process. Note that the table does not report the number of missing values because we have already handled them in Subsection 3.2 and, thus, no attribute contains missing values.

Attribute	Mean	Median	Standard deviation	Min	Max	Skewness	Kurtosis
Age	0	0.02	1	-2.52	2.8	-0.3	-0.56
Total Bilirubin	0	-0.46	1	-1.35	3.78	1.32	0.92
Direct Bilirubin	0	-0.42	1	-1.25	2.74	0.83	-0.28
Alkaline Phosphotase	0	-0.3	1	-2.56	4.09	2.28	1.32
Alamine Aminotransferase	0	-0.22	1	-1.61	4.28	1.43	2.63
Aspartate Aminotransferase	0	0.22	1	1.66	4.56	1.19	1.6
Total Proteins	0	0.11	1	-3.49	2.87	-0.22	0.23
Albumin	0	-0.05	1	-2.82	2.96	-0.04	-0.39
Albumin and Globulin Ratio	0	-0.04	1	-2.03	5.8	0.99	3.26

Table 4: Numerical attribute statistics after normalization

3.4 Unbalanced Class Problem

As shown in Figure 1, the dataset is unbalanced, as 71.4% of the records refer to patients with liver diseases while the remaining 28.6% to patients without any liver disease. For this reason, we decided to balance the dataset by applying *SMOTE* [2], as previous studies demonstrated that it showed good results in medical datasets [8].

SMOTE [2] is an oversampling approach to balance the classes of a dataset and, as such, it solves the unbalanced class problem by generating new instances of the minority class until the cardinalities of the classes are the same. In particular, to generate new instances, it selects an instance of the minority class and finds its k nearest neighbors to generate new "synthetic" instances that will be added to the dataset. The algorithm proceeds iteratively until all the classes have the same number of instances.

Since *SMOTE* [2] needs to find the k nearest neighbors of an instance, we dealt with the problem of finding the optimal number of nearest neighbors k to use. However, the solution to this problem strongly depends on the dataset being used, but, if we take into consideration the *K Nearest Neighbors* algorithm [3], which is also based on the concept of finding the nearest neighbors of an instance, an often used rule of thumb in many applications is that of setting $k = \sqrt{N}$, where N is the number of instances within the training set [11]. Following this guideline, since the minority class has 167 instances, we used $k = \sqrt{167} = 13$.

At the end of the class balancing process, the dataset contained 832 instances, 416 for each class.

4 CLASSIFICATION

After having preprocessed the data, we selected 3 different classification models to solve the problem of predicting whether a patient is affected by any liver disease or not.

4.1 Classification models

We tested 3 different classification models to find the most effective one to diagnose liver diseases:

- *Random Forest* [9];
- *K Nearest Neighbors* [3];
- *Logistic Regression* [7].

More in detail, *Random Forest* [9] is a supervised learning method that consists of a large number of individual decision trees that operate as an ensemble, where each decision tree is built by selecting a random subset of attributes. Every decision tree predicts a class individually and the prediction of the model is the most voted class.

K Nearest Neighbors [3] is a supervised learning algorithm that assumes that similar things are near to each other. It considers instances as vectors in a geometrical space and uses a distance measure, such as the Euclidean distance, to find the k instances that are the closest to the instance to be classified. Then, the algorithm assigns the class label by selecting the majority class of the k nearest neighbors of the considered instance.

Finally, *Logistic Regression* [7] is a supervised statistical model that uses a logistic function to estimate the probability of an instance to belong to a class in a binary classification problem. The logistic function is defined as

$$logistic(x) = \frac{1}{1 + e^{wx}}$$

where x is the instance and w is the vector of parameters of the model that have to be estimated. The logistic function takes values in the $[0, 1]$ interval.

4.2 Hyperparameter optimization

Before training the classification models described in Subsection 4.1, we dealt with the hyperparameter optimization problem, as *Random Forest* [9] and *K Nearest Neighbors* [3] require parameters that have to be set by users: the number of trees t composing the forest and the number of nearest neighbors k to consider.

The task of hyperparameter optimization is often performed by dividing the dataset into train set, validation set and test set, where the validation set is used to tune the best hyperparameters of the models. However, the dataset is too small to apply this approach and, for this reason, we decided to use the *nested-cross validation* approach [1] exemplified in Figure 3. As illustrated, it uses two

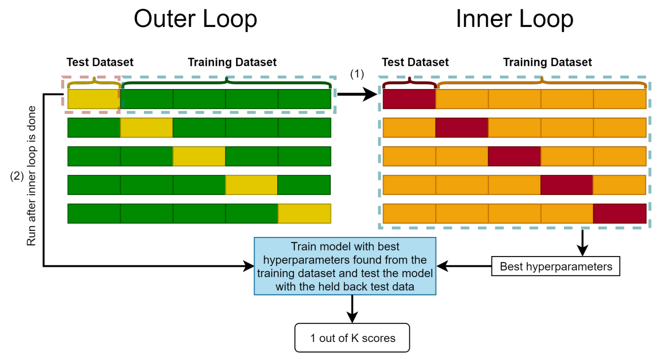
Outer Loop
Inner Loop
Test Dataset
Training Dataset
Test Dataset
Training Dataset
(1)
(2)
Run after inner loop is done
Train model with best hyperparameters found from the training dataset and test the model with the held back test data
Best hyperparameters
1 out of K scores

Figure 3: Nested cross validation

nested k-fold cross validations: the inner one is used to tune the hyperparameters of the model, while the outer one is used to evaluate the model performances when using the optimal hyperparameters. For our study, we used a 10-fold cross validation both for the inner and the outer loops. We focused our attention on optimizing the hyperparameters with the objective of maximizing the *f-measure* of the class related to patients with liver diseases. The rationale of this decision is described in Subsection 5.2. To discover the optimal number of trees t for *Random Forest* [9], we experimented different values starting from 64 up to 128, increasing t by 4 at each step, because the optimal value should be contained in this interval [6]. Instead, to discover the optimal number of nearest neighbors k, we made k vary from 1 to the square root of the cardinality of the training set [5] used in the inner loop of the nested cross validation, that is, 26.

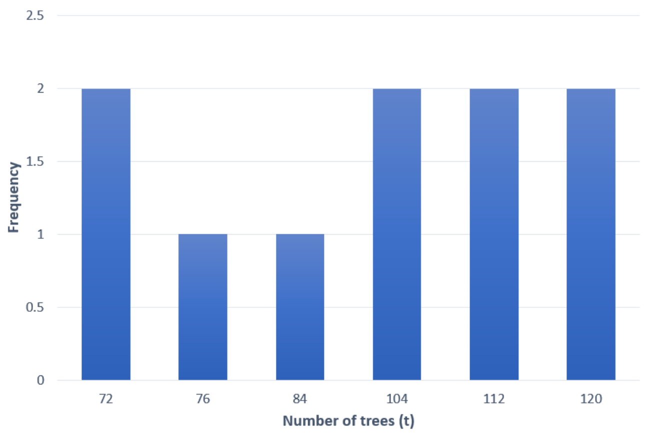

Figure 4: Absolute frequency of the obtained optimal number of trees with the inner loops of the nested cross validation

The results of the best values of t are shown in Figure 4. As we can see, the value of t is not stable, but it strongly depends on the training set used to tune it. On the contrary, the value of k is always equal to 1, which means that *K Nearest Neighbors* [3] may be more robust than *Random Forest* [9].

5 PERFORMANCE EVALUATION

We trained the models described in Section 4 with the optimal hyperparameter values obtained through the inner loop of the nested cross validation and we assessed their performances with the objective of selecting the optimal one for this classification problem.

5.1 Common performance evaluation measures

The performances of a classification model are assessed by computing some major metrics, that is, *accuracy*, *recall*, *precision* and *f-measure*, which can be obtained from the confusion matrix, which contains the counts of:

- *True Positives (TP)*: the number of instances of the *positive* class correctly classified as *positive* by the model;
- *False Positives (FP)*: the number of instances of the *negative* class wrongly classified as *positive* by the model;
- *True Negatives (TN)*: the number of instances of the *negative* class correctly classified as *negative* by the model;
- *False Negatives (FN)*: the number of instances of the *positive* class wrongly classified as *negative* by the model.

The aforementioned measures are computed as follows:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN} \qquad f\text{-}measure = \frac{2 \times precision \times recall}{precision + recall}$$

More in detail, *accuracy* is the fraction of the instances correctly classified by the model, *precision* is the fraction of the instances labeled as *positive* by the model that actually belong to the *positive* class, *recall* is the fraction of the instances belonging to the *positive* class correctly labeled as *positive* by the model and, finally, *f-measure* is the harmonic mean between *precision* and *recall*.

5.2 Accuracy is not the most relevant metric

In this context, we think that *accuracy* is not the most suitable metric to consider when evaluating and comparing the models, because it considers all the classes as equally important. However, the class comprehending patients with liver diseases assumes a more relevant importance than the class containing patients without liver diseases, because misclassifying a ill patient as healthy has more serious consequences than misclassifying a healthy patient as ill. Indeed, the first error could lead to the patient death, while in the second case the diagnosis could be proven wrong by further medical check-ups, leading just to a waste of money and time. From now on, we will refer to the class of patients with liver diseases as the *positive* class and to the other as the *negative* class.
We considered *f-measure* as the most relevant metric because it takes into account both *precision* and *recall*. Indeed, as we are not domain experts, we made the decision to consider *precision* and *recall* as equally important. In addition, since the *positive* class is more important than the *negative* one, we consider a model A better than a model B if the *f-measure* of the *positive* class achieved by A is higher than the one achieved by B. In this way, the best model is the one able to correctly classify ill patients while limiting the number of false positives, that is, wrongly classifying healthy patients as ill.

5.3 Results

As already mentioned in Subsection 4.2, we used a nested cross validation for the hyperparameter optimization and the performance evaluation. We trained the models using the hyperparameters selected with the inner loop, except for the *Logistic Regression* [7] model, as it does not require any hyperparameter to tune. The obtained results are reported in Tables 5 and 6, which show the mean values for all the performance measures achieved by the models for both the *positive* and *negative class*, respectively. In particular, the cells of the tables containing the best value of a specific performance measure are highlighted in gray. Concerning the *positive* class, *Random Forest* [9] achieved the best values for the *accuracy*, *recall* and *f-measure*, but, as regards *precision*, *K Nearest Neighbors* [3] achieved a better value. This means that *Random Forest* [9] is better at detecting patients with liver diseases, although it generates more false positives than *K Nearest Neighbors* [3]. Considering the *positive* class, we can see that the main contribution to *f-measure*

is given by *precision*. In fact, the values of *recall* for all the models are always smaller than the ones of *precision*. This result means that the models are better at limiting the number of false positives rather than identifying ill patients.
On the contrary, as regards the *negative* class, *Random Forest* [9] achieved the best values for *accuracy* and *precision*, while *K Nearest Neighbors* [3] for *recall* and *f-measure*.
From these results, it seems that *Random Forest* [9] is performs better than the other models because it achieved the highest value of *f-measure* considering the *positive* class.

Model	Accuracy	Precision	Recall	F-measure
Random Forest	0.826	0.882	0.755	0.81
K Nearest Neighbors	0.818	0.933	0.69	0.79
Logistic Regression	0.719	0.767	0.633	0.691

Table 5: Performance measures on the positive class

Model	Accuracy	Precision	Recall	F-measure
Random Forest	0.826	0.79	0.897	0.838
K Nearest Neighbors	0.818	0.756	0.947	0.84
Logistic Regression	0.719	0.688	0.805	0.741

Table 6: Performance measures on the negative class

However, these results have been obtained using simple point estimates, which do not provide any information about their approximation degree and, thus, they do not allow us to conclude which model is better than the others. For this reason, we decided to use confidence intervals that should contain the true value of *f-measure* with good probability, allowing us to better compare the classification models under study.
Since we are interested in finding the model achieving the best result of *f-measure*, as already mentioned in Subsection 5.2, we computed the confidence interval for this performance measure [10] with a significance level $\alpha = 0.05$.
First of all, we computed an estimate of the *f-measure* $\hat{f}_k$ for each k-th iteration of the outer 10-fold cross validation using the corresponding confusion matrix. Then, we computed the mean $\hat{\mu}$ and the sample variance $\hat{S}^2$ of the *f-measure* as follows:

$$\hat{\mu} = \frac{1}{K}\sum_{k=1}^{K}\hat{f}_k \qquad \hat{S}^2 = \frac{\sum_{k=1}^{K}(\hat{f}_k - \hat{\mu})^2}{K-1}$$

Finally, we computed the confidence interval for the *f-measure* in the following way:

$$\left[\hat{\mu} - t_{K-1,1-\frac{\alpha}{2}}\sqrt{\frac{\hat{S}^2}{K}}, \hat{\mu} + t_{K-1,1-\frac{\alpha}{2}}\sqrt{\frac{\hat{S}^2}{K}}\right]$$

where $t_{K-1,1-\frac{\alpha}{2}}$ is the percentile from Student's t distribution with degree of freedom $K-1$.
Figure 5 shows the confidence intervals of the *f-measure* obtained by the models. As we can see, the confidence intervals of *Random Forest* [9] and *K Nearest Neighbors* [3] overlap and, thus, we can not conclude that the difference between the two models is statistically significant at the significance level $\alpha = 0.05$. Instead, concerning the *Logistic Regression* [7] model, we can conclude that *Random Forest* [9] and *K Nearest Neighbors* [3] outperform it.
To further investigate the performances of the models, we computed the *ROC* curves for the *positive* class, which is illustrated in Figure 6. As we can see, the blue curve intersects both the red and the green ones while the red curve does not intersect the green one. In particular, since the red curve always dominates the green one, we can conclude that, also in this case, *Random Forest* [9] performs better than *Logistic Regression* [7]. As regards the red and the blue curves, we can see that the red one dominates the blue one except when the *False Positive Rate* takes values in [0.05, 0.08]. This means that if we admit a *False Positive Rate* between 0.05 and 0.08, *K Nearest Neighbors* [3] achieves a better *recall* value than *Random Forest* [9].
In addition, as regards *Area Under Curve (AUC)*, *Random Forest* [9] achieved a value equal to 0.9095, which is greater than the ones obtained by the other models, that is, 0.8183 and 0.7683 for *K Nearest*

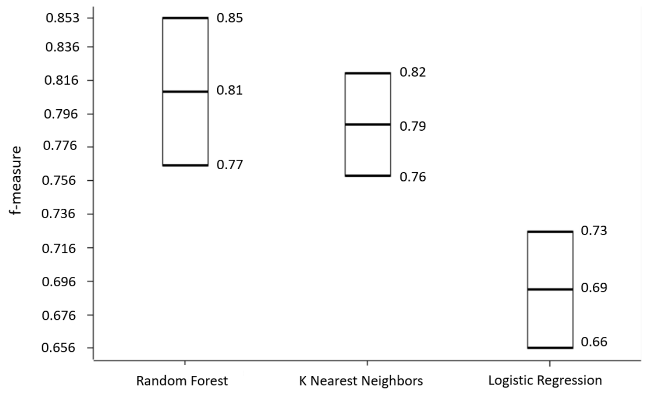

Figure 5: Confidence intervals for f-measure considering the positive class

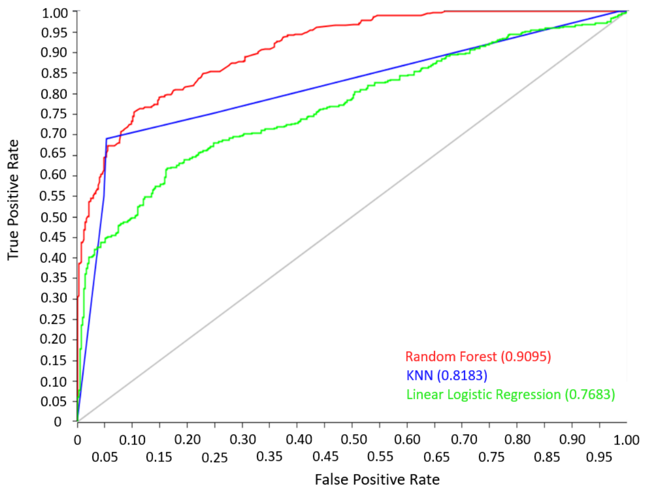

Figure 6: ROC curves of the models

Neighbors [3] and *Logistic regression* [7], respectively.
Overall, by the results obtained with the *ROC* curves, *Random Forest* [9] seems to be the most suitable model among the ones we compared for this classification problem.

5.4 Decision threshold analysis

Generally, all the classification models can output the prediction probability of a record, which corresponds to the probability of the record to be predicted as *positive*. Usually, when such probability is greater than 0.5, the record is assigned to the *positive* class, to the *negative* class otherwise. This threshold is also called *decision threshold* and it can be varied to optimize a specific performance measure, such as *f-measure*.
The models compared in Subsection 5.3 use a *decision threshold* equal to 0.5. Since this threshold may not be the optimal one to optimize the *f-measure*, we studied the behavior of the models to the varying of the *decision threshold*. Figures 7, 8 and 9 show how the performances of the models change with respect to the selected threshold value, considering the *positive* class.
In particular, Figure 7 refers to *Random Forest* [9] and shows that the optimal threshold is 0.4, which leads to achieve an *f-measure* value equal to 0.815. This value is slightly higher than the one reported in Table 5. As regards Figure 8, which considers *K Nearest Neighbors* [3], we can see that the optimal threshold is any value greater than 0.01. This is related to the fact that the model uses only a nearest neighbor to predict the class of the considered record. Thus, the result reported in Table 5 is already the highest *f-measure* value obtainable by the model. Finally, Figure 9 shows that the optimal threshold for the *Logistic Regression* [7] model is 0.46, which leads to increase the *f-measure* value from 0.691 to 0.702. Also in this case, the increment is minimal.

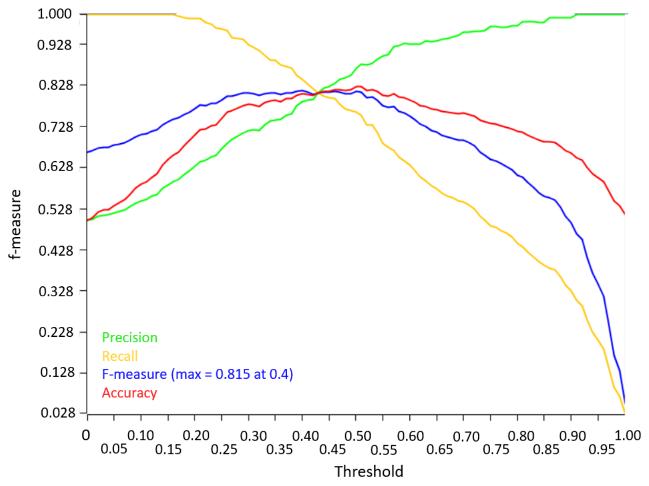

Figure 7: Random Forest's performance to the varying of the decision threshold considering the positive class

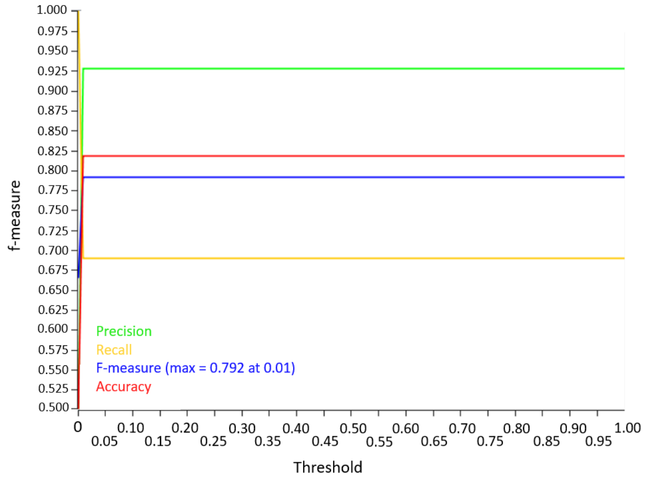

Figure 8: K Nearest Neighbors' performance to the varying of the decision threshold considering the positive class

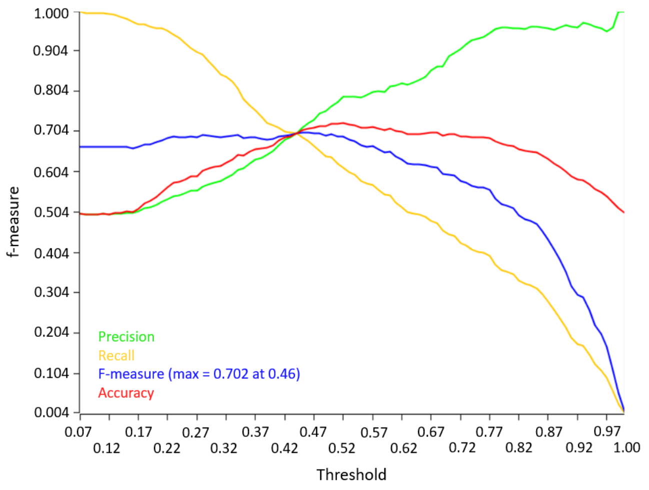

Figure 9: Logistic Regression's performance to the varying of the decision threshold considering the positive class

6 CONCLUSIONS

This work presented a study on the capability of three different classification models in predicting liver diseases on Indian patients, based on patient data mainly obtained from targeted blood tests. The dataset used contains 583 patient records: 416 patients have liver diseases, while the remaining 167 ones do not have any liver disease.
We preprocessed the data to solve some issues, such as the presence of missing values and the unbalanced class distribution. More precisely, we applied a logarithmic transformation on those attributes whose distribution resulted highly right-skewed. Then, we applied a missing value imputation technique based on linear regression to replace the missing values with concrete values, concerning the *Albumin and Globulin Ratio* attribute. Soon afterwards, we normalized the values of the attributes using the Z-Score Normalization.

Finally, we balanced the dataset using the oversampling technique *SMOTE* [2].

After having preprocessed the data, we selected 3 learning algorithms, namely *Random Forest* [9], *K Nearest Neighbors* [3] and *Logistic Regression* [7], and we assessed the performances of the induced models with the objective of choosing the best classifier to solve this classification problem. Since *Random Forest* [9] and *K Nearest Neighbors* [3] require users to set some parameters, we used a nested cross validation, in which the inner loop is used to tune the hyperparameters while in the outer loop the model built with the optimal hyperparameter is evaluated. The evaluation measure that we considered more relevant is the *f-measure* computed on the *positive* class, where the *positive* class is represented by the class of patients with liver diseases. In this way, the optimal classifier is the one that correctly predicts records belonging to the *positive* class while limiting the number of healthy patients classified as ill.

The performance evaluation showed that *Random Forest* [9] and *K Neatest Neighbors* [3] obtained similar performances considering their values of *f-measure*, although the former seems better because it achieved the highest value of *AUC*. Nevertheless, *Random Forest* [9] resulted less robust because the optimal number of trees required by the model strongly depends on the training set. The *Logistic Regression* [7] model achieved a poor value of *f-measure*, which means that it is not very suitable for this classification problem.

Analyzing the *ROC* curves of the models, we realized that *Random Forest* [9] seems to be better than *K Nearest Neighbors* [3], despite the latter achieves a higher *True Positive Rate* when the *False Positive Rate* takes values in $[0.05, 0.08]$.

Finally, we studied how the *f-measure* of the *positive* class achieved by the models changed to the varying of the *decision threshold*. As a result, we obtained that the *decision threshold* set to 0.5 for *K Nearest Neighbors* [3] is already optimal, while the optimal *decision threshold* for *Random Forest* [9] and *Logistic Regression* [7] are 0.4 and 0.46, respectively, which lead to a very little improvement of their *f-measure* values.

In conclusion, from this work, we discovered that *Random Forest* [9] and *K Nearest Neighbors* [3] achieved good performances, although the major contribution to *f-measure* is given by *precision*. This fact leads to conclude that the studied models are better at limiting the number of false positives rather than identifying ill patients. However, if we do not consider *precision* and *recall* as equally important and the objective is to maximize one of them, we can tune the hyperparameters of the models to achieve the optimal value for the considered metric and vary the *decision threshold* accordingly.

REFERENCES

[1] Bernd Bischl, O Mersmann, Heike Trautmann, and Claus Weihs. 2012. Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation. *Evolutionary computation* 20 (02 2012), 249–75. https://doi.org/10.1162/EVCO_a_00069

[2] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res. (JAIR)* 16 (01 2002), 321–357. https://doi.org/10.1613/jair.953

[3] T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13, 1 (1967), 21–27.

[4] Changyong Feng, Wang Hongyue, Naiji Lu, Tian Chen, Hua He, Ying Lu, and Xin Tu. 2014. Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry* 26 (04 2014), 105–9. https://doi.org/10.3969/j.issn.1002-0829.2014.02.009

[5] Ahmad Hassanat, Mohammad Abbadi, Ghada Altarawneh, and Ahmad Alhasanat. 2014. Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach. *International Journal of Computer Science and Information Security* 12 (08 2014), 33–39.

[6] Thais Oshiro, Pedro Perez, and José Baranauskas. 2012. How Many Trees in a Random Forest? *Lecture notes in computer science* 7376. https://doi.org/10.1007/978-3-642-31537-4_13

[7] Joanne Peng, Kuk Lee, and Gary Ingersoll. 2002. An Introduction to Logistic Regression Analysis and Reporting. *Journal of Educational Research - J EDUC RES* 96 (09 2002), 3–14. https://doi.org/10.1080/00220670209598786

[8] Mostafizur Rahman and Darryl N. Davis. 2013. Addressing the Class Imbalance Problem in Medical Datasets. *International Journal of Machine Learning and Computing* 3 (04 2013), 224. https://doi.org/10.7763/IJMLC.2013.V3.307

[9] Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 1. 278–282 vol.1.

[10] Y. Wang, J. Li, Y. Li, R. Wang, and X. Yang. 2015. Confidence Interval for F-1 Measure of Algorithm Performance Based on Blocked 3x2 Cross-Validation. *IEEE Transactions on Knowledge and Data Engineering* 27, 3 (2015), 651–659.

[11] Jaesub Yun, Jihyun Ha, and Jong-Seok Lee. 2016. Automatic Determination of Neighborhood Size in SMOTE. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication (IMCOM '16)*. Association for Computing Machinery, New York, NY, USA, Article 100, 8 pages. https://doi.org/10.1145/2857546.2857648

[12] Zhongheng Zhang. 2016. Missing data imputation: Focusing on single imputation. *Annals of translational medicine* 4 (02 2016), 9. https://doi.org/10.3978/j.issn.2305-5839.2015.12.38