Hepatis C Model selection

Spyridon Alvanakis-Apostolou

April 2023

Abstract

Hepatitis C is a viral infection that causes inflammation of the liver and can lead to serious health problems. Machine Learning (ML) methods offer a promising approach to analyzing various features and laboratory test values to effectively diagnose Hepatitis C patients and generate laboratory diagnostic protocols. In this assignment, I built a complete ML pipeline to classify future patients according to the label column using a nested Cross Validation (nCV) pipeline. I compared the performance of five classification algorithms and selected the one that achieved the highest average test MCC performance in 10 trials of nCV. After finding the winner algorithm, I used the whole dataset and simple cross-validation with 5 folds to determine the final model to deploy in the field.

1 Introduction

Hepatitis C[1] is a viral illness that affects the liver and the immune system. Chronic hepatitis C can cause serious health complications like liver cancer and cirrhosis. The transmission of Hepatitis C occurs when there is contact with the blood of an individual who is infected with HCV. This can happen through many ways such as sexual contact or sharing drug needles with someone who has HIV. One can also get infected if they are pricked by a needle that was previously used on a person with HCV in a hospital or medical facility etc. Early-stage symptoms of this disease may not be noticeable, and patients may only experience symptoms of advanced liver disease. Sadly, there is no vaccine currently available for Hepatitis C.

Effective treatments for Hepatitis C can cure a patient within 8-12 weeks, if administered on time. Unfortunately, in more than 50% of the cases, the patient's body becomes unable to clear the virus and hence this acute infection transforms into a chronic disease.

For the better understanding of this condition and its prognosis, we can use patient's electronic health information to uncover new findings and trends that might otherwise go unnoticed. Machine learning classifiers could help in order to predict the diagnoses of healthy and also diagnosed patients with hepatis C in time, since many of them already achieving great results on this prediction[12] [9]. This category of classification problems are very important as they can help us identify diseases accurately and efficiently. Other viral infections and cancers may benefit from the use of genetic analysis and machine-learning techniques to choose the best course of treatment.

2 Methods

2.1 Understanding the Data

The given dataset, contains an extensive list of values for 204 different patients (68 of them found to be positive in Hepatitis C), including their age, sex, and various biochemical markers such as ALB, ALP, ALT, AST, BIL, CHE, CHOL, CREA, GGT, and PROT. Additionally, the dataset also includes a label that indicates if a patient has hepatitis C or not. To ensure the accuracy of the data, I conducted a search for NaN and null values, but none were found. However, it is noted that some individual extreme values were found, which were labeled as positive for Hepatitis

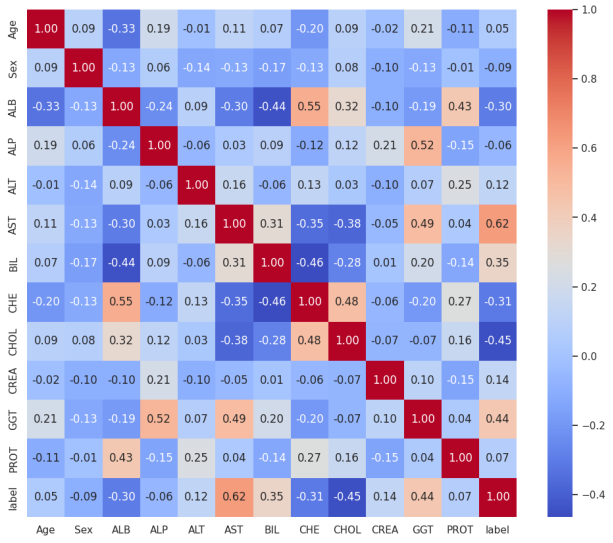

Figure 1: Correlation Matrix

C, so I made the decision to keep these values in the data set, rather than remove or change them.

In order to gain a deeper understanding of the data, the second step was to find the correlations between the various variables. Also to illustrate the differences between the distributions of the healthy and deceased patients in figure 1 and figure 6. In addition to this, I was also able to identify the seven parameters that had the highest absolute correlation with Hepatis C, which is a critical finding for anyone working in the field. This information is illustrated in figure 3.

2.2 Searching for the best Classifier

The next step of the project, was the normalization of the data of each column. However, not all columns require normalization, and in this case, I excluded the `labels` and `sex` columns due to their binary values. For the remaining columns, I utilized the `StandardScaler`[5] function -which is using the normal distribution for the normalization of data-, rather than the `MinMaxScaler`[3] -which transforms the features by scaling each feature to a given range. The reason for this choice was due to the extreme values that were present in the data set before the normalization. After the normalization, the next step was to split the dataset into train and test sets. The purpose of this was to evaluate the model and check if the results were accurate on the test set too. The train set was used to fit the model.

After performing data normalization, various classifiers with different features were created to identify the best fitting model for the Hepatitis C dataset. The classifiers that evaluated included Support Vector Machine (SVM) [6], Logistic Regression (LR) [7], Gaussian Naive Bayes (GaussianNB) [8] [13], Linear Discriminant Analysis (LDA) [10], and K-Nearest Neighbor (kNN) [11] from the sklearn library. In order to set the classifiers, the `pipeline`[4] library was utilized, along with `GridSearchCV`[2] algorithm to identify the best basic parameters for each classifier. Also for every classifier, a nested cross-validation was employed using three splits cross-validation for the inner loops and five splits cross-validations for the outer loop. The best fitting model was determined by evaluating `MCC score` for binary classification in the inner loop, as we can see at the Workflow Diagram 5. In the outer loop, the best model was used to calculate various metrics, including MCC, Balanced Accuracy, F1, F2, Recall, Precision, Accuracy, roc_auc , Average Precision, and NPV. For ten iterations the mean of the above metrics of every outer cross-validation was stored and mean results of the iterations was calculated to identify the best model with respect to the mean MCC score.

2.3 Best Classifier

The result on Matrix 7, the Boxplot 9 and the Barplot 2 illustrating that the SVM classifier did better than the others in terms of the MCC score, having a small difference with `LinearRegression()`. So, the SVM classifier was further improved by finding the best hyperparameters to make it perform even better. This process took a long time and used a lot of computer resources. At first, the model's performance was measured using the `roc_auc_score`, and the best hyperparameters found to be: as observed in `table 1` with best `roc_auc_score` of almost 0.94 and mean `roc_auc_score` approximately 0.92.

Considering the seriousness of Hepatitis C, false

Parameter	value
C	10
class_weight	'balanced'
break_ties	True
kernel	'linear'
max_iter	100
probability	True
verbose	True

Table 1:

negative results can have significant consequences, including delaying the treatment and potentially resulting in a chronic disease. To account for this, a different function was created and applied as scorer that weighted false negative values of the confusion matrix more heavily than false positives. Specifically, false negatives were weighted 4 times more than false positives. The mean `roc_auc_score` for this function model was almost 0.92 and the best `roc_auc_score` almost 0.94, which is a good result considering the potential consequences of false negatives in Hepatitis C diagnosis.

3 Results and Discussion

The initial dataset consisted of data from 204 patients, which included both their blood test results and profile characteristics. The most strongly correlated values in the dataset were AST, CHOL, GGT, BIL, CHE, and ALB. Additionally, it was observed that patients who tested positive for Hepatitis C were generally around 50 years old, although this observation may be biased due to the small sample size of the dataset.

Based on the evaluation metrics, the LogisticRegression and SVM classifiers performed the best among all the classifiers tested, as shown in Matrix 2. However, the SVM classifier outperformed the LogisticRegression classifier in every countable metric, as demonstrated in Matrix 7 and the Boxplot 9. On the other hand, the KNN classifier resulted in the worst scores in every metric except for precision. Therefore, the SVM classifier was ultimately selected as the best classifier.

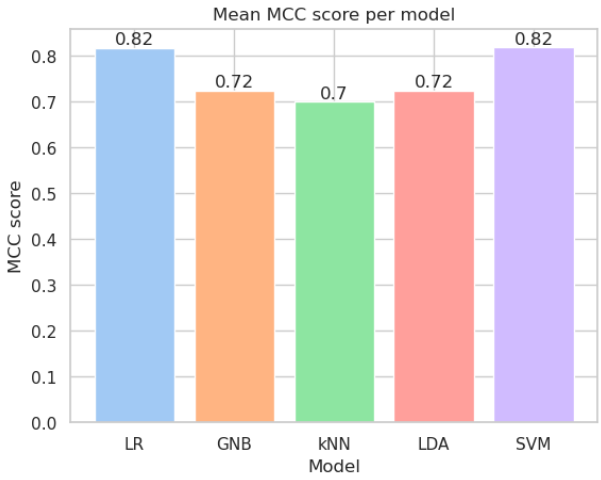

Figure 2: The LinearRegression and SVM classifiers performed almost equally well, with the LinearRegression achieving a mean MCC score of 0.817 and the SVM achieving a mean MCC score of 0.818. Although the difference was small, I also considered the fact that the SVM outperformed the LinearRegression on almost every other metric before making my final choice for the classifier.

I tested a great amount of hyperparameters that are presenting on `table2`.

Parameter	Values
Kernel	['linear', 'poly', 'rbf', 'sigmoid']
Decision function shape	['ovo', 'ovr']
C	[0.01, 0.01, 0.1, 1, 10, 100, 1000]
Gamma	['scale', 'auto']
Shrinking	[True, False]
Verbose	[True, False]
Max iter	[1, 10, 100, 1000, 10000]
Break ties	[True, False]
Class weight	['balanced', None, [0: 0.67, 1: 0.33]]

Table 2: Parameter values for SVM classification

After almost 8 minutes of an intensive cpu procceedure, I ended up with the optimal hyperparameters for a 5 split cross-validation with roc_auc scorer to be as presented on `table 2`.

The roc_auc score for this model was approxi-

mately 0.92 with max score almost 0.94, and it had high scores for most of the other evaluation metrics. The results are illustrating on Figure 8.

Metric	Mean value
MCC score	0.86
balanced accuracy	0.92
F1 score	0.89
F2 score	0.88
recall score	0.87
precision score	0.93
accuracy score	0.93
average precision score	0.85
NPV score	0.94
roc auc score	0.92

Table 3:

Although the recall score achieved great scores, I aimed to improve it further. The recall score have significant importance as the model should have the ability to correctly identify all diseased patients, if not, this could lead to delays in starting therapy for patients. As a result, this may cause the disease to progress to a chronic or even fatal state. Although the roc_auc score is a useful metric for binary classification problems, in this case, a different scoring method was developed to assign higher weight to false negative values and ensure that the model correctly identifies as many true positive cases as possible.

In order to increase the recall score, a custom scoring function was successfully created after many tries, that eventually added more weight to false negatives values and also maximized the `accuracy_score` and the `roc_auc _score`. This was achieved with the function `custom_scorer` sums the `accuracy_score` with `roc_auc _score` multiplied by five in order to increase its weight. After that i subtracted $\frac{2fn}{fn+tp}$ and $\frac{1}{2}\frac{fp}{fp+tn}$ from the sum in order to preserve high the precision levels, but also to increase the weigh of the false negatives. The hyperparameters options used for this function were the same as before, with a 5 split cross-validation. The best resulting hyperparameters had only one difference with the previous model, the `class_C=1`.

As we can see, the custom function increased almost all metrics except the MCC and the precision score.

Metric	Mean value
MCC score	0.85
balanced accuracy	0.92
F1 score	0.90
F2 score	0.89
recall score	0.89
precision score	0.90
accuracy score	0.93
average precision score	0.84
NPV score	0.95
roc_auc score	0.92

Table 4:

The most improvement was in the mean recall score, which increased by 0.02. While the `roc_auc_score` hold the same mean. It is important to note that false positive scores cannot be fatal or become a reason for a chronic disease. Concluding i kept model as it minimized risk while returning generally better results 8.

3.1 Discussion

To understand how we ended up at the Matrix 7, it is important to understand the tasks that each classifier used for. The Support Vector Machine (SVM) classifier is a robust algorithm that can handle both linear and nonlinear classification problems. In our case, where we have a linear classification problem, the SVM finds a hyperplane that separates the two classes by maximizing the margin between them. It is well-suited for high-dimensional problems such as ours.

The Gaussian Naive Bayes (GNB) classifier is a probabilistic classifier that works based on the naive Bayes theorem. It assumes independence between the features of the data, which is not accurate in our case due to the correlations of the features between them and also with the labeled data.

The linear regression classifier demonstrated strong performance on both the train and test sets. Generally it works by fitting a linear function to the data,

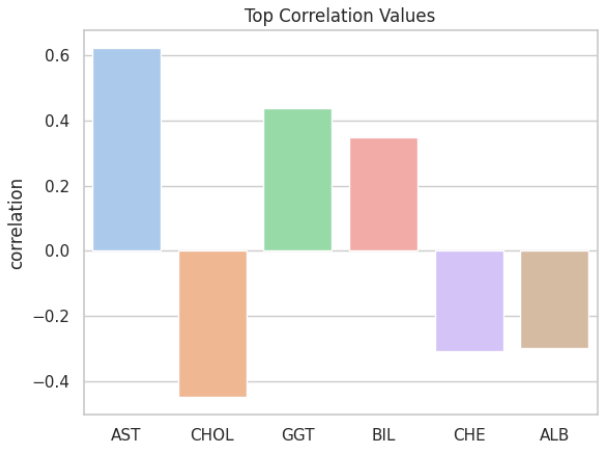

Figure 3: Top Correlations

Figure 4:

which is achieved by minimizing the sum of squares between the observed and predicted data. In our case, the dataset could be effectively linearly classified, which likely contributed to the good performance of this classifier.

The LDA classifier is also commonly used for linear classification problems, but it assumes that the data are Gaussian distributed with equal covariance matrices. This assumption wasn't generally accurate in our dataset, as evidenced by the Pairplot 6, which likely contributed to the poor classification performance observed. It's important to note that although the LDA classifier may have overall a poor classification performance it accurately predicted false positive and true positive features as we can observe in the Boxplot 9 precision score.

I also experimented with different values of k for the KNN classifier, as this algorithm does not make any assumptions about the distribution of the data features. However, the KNN classifier also did not perform well for our problem due to the large number of dimensions in our dataset. This classifier is affected by the curse of dimensionality and generally performs poorly on problems with many dimensions.

For every classifier I employed a range of performance metrics to evaluate the effectiveness of each one of them. These metrics included accuracy_score, f1, f2, precision, average_precision, npv, balanced_accuracy, roc_auc score, mcc score, and recall. Each metric provides unique insights into different aspects of the model's performance.

Using tp,tn,fp,fn as the outcomes of a confusion matrix, we can describe the utilized metrics with the following way. Accuracy score calculates the percentage of correct classifications in a classification problem :

$$\frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}-1} 1(y_{\text{ipred}} = y_{\text{itest}})$$

or

$$\frac{tp + tn}{tp + tn + fp + fn}$$

in a binary classification problem. Precision measures the ability of the model to minimize false positives:

$$\frac{tp}{tp + fp}$$

while recall measures its ability to minimize false negatives:

$$\frac{tp}{tp + fn}$$

The F1 score strikes a balance between precision and recall:

$$\frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

while F2 places double weight on recall. Balanced accuracy calculates the average recall and specificity in imbalanced data:

$$\frac{1}{2}(\frac{tp}{tp + fn} + \frac{tn}{tn + fp})$$

Average precision summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold:

$$\sum_{i=1}^{n} (R_n - R_{n-1})P$$

where R_n and P_n are the precision and recall at the nth threshold , while the negative predictive value (NPV) measures the precision of the negative results:

$$\frac{tn}{tn + fn}$$

The Matthews correlation coefficient provides a measure of the quality of the classification:

$$\frac{tp * tn + fp * fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}$$

Finally, the `roc_auc score` is a measure of the model's ability to distinguish between positive and negative examples, and is calculated as the area under the roc curve.

In the hyperparameter selection cross-validation, I decided to use my own custom scorer function. While I wanted to give false negative values and the `roc_auc score` different weights. I experimented with different weight values for false negatives and the `roc_auc score` and finally settled on the `custom_scorer`, which notably increased the recall score and the `MCC score` but didn't improved the `roc_auc score`.

4 Conclusion

In this assignment, I utilized machine learning techniques to accurately estimate the risk of Hepatitis C development. By identifying the most highly correlated features with Hepatitis C prediction and using various classifiers with basic hyperparameters, I was able to achieve an average Matthews correlation coefficient (MCC) score of 0.8. After fitting the best hyperparameters to the classifier, I was able to further increase its performance at MCC score, as well as other important metrics. In particular, I was able to increase the recall score to 0.91 while maintaining a high accuracy score of almost 0.93. This indicates that the final model with the custom function was able to reduce the number of false negatives while maintaining high accuracy levels. The final hyperparameters for the model are illustrated on the `Table 5`.

5 Future Work

A different approach for this task would be to remove the features having low absolute correlation with the label features in order to decrease the dimension number of the problem. Also the transformation of the data to -1, 0, 1 based on whether they are below, equal to, or above the normal range of values for each feature could help, since the decision borders will be easier to discover.

Parameter	value
C	1
break_ties	True
class_weight	'balanced'
kernel	'linear'
max_iter	100
probability	True
verbose	True

Table 5:

References

[1] Hepatitis c information.

[2] scikit-learn/gridcv.

[3] scikit-learn/minmaxscaler.

[4] scikit-learn/pipeline.

[5] scikit-learn/srandarscaler.

[6] Christopher M. Bishop. Pattern recognition and machine learning. *chapter 7 Sparse Kernel Machines.*

[7] Christopher M. Bishop. Pattern recognition and machine learning. *Chapter 4.3.4.*

[8] T.F. Chan, G.H. Golub, and R.J. LeVeque. Updating formulae and an pairwise algorithm for computing sample variances. *Stanford working paper STAN-CS-79-773*, 1979.

[9] Giuseppe Jurman Davide Chicco. An ensemble learning approach for enhanced classification of patients with hepatitis and cirrhosis. 2021.

[10] Friedman J. Hastie T., Tibshirani R. The elements of statistical learning. *Section 4.3, p.106-119*, 2008.

[11] G. Hinton R. Salakhutdinov J. Goldberger, S. Roweis. Neighbourhood components analysis. *Advances in Neural Information Processing Systems, Vol. 17*, 2005.

[12] Imed Ben Dhaou Charles Chuka Agubosim Chukwudum Collins Umoke Nneka Ernestina Richard-Nnabu Neeraj Dahiya Michael Onyema Edeh, Surjeet Dalal. Artificial intelligence-based ensemble learning model for prediction of hepatitis c disease. 2022.

[13] Zhang. The optimality of naive bayes. *Faculty of Computer Science University of New Brunswick.*

A Plots

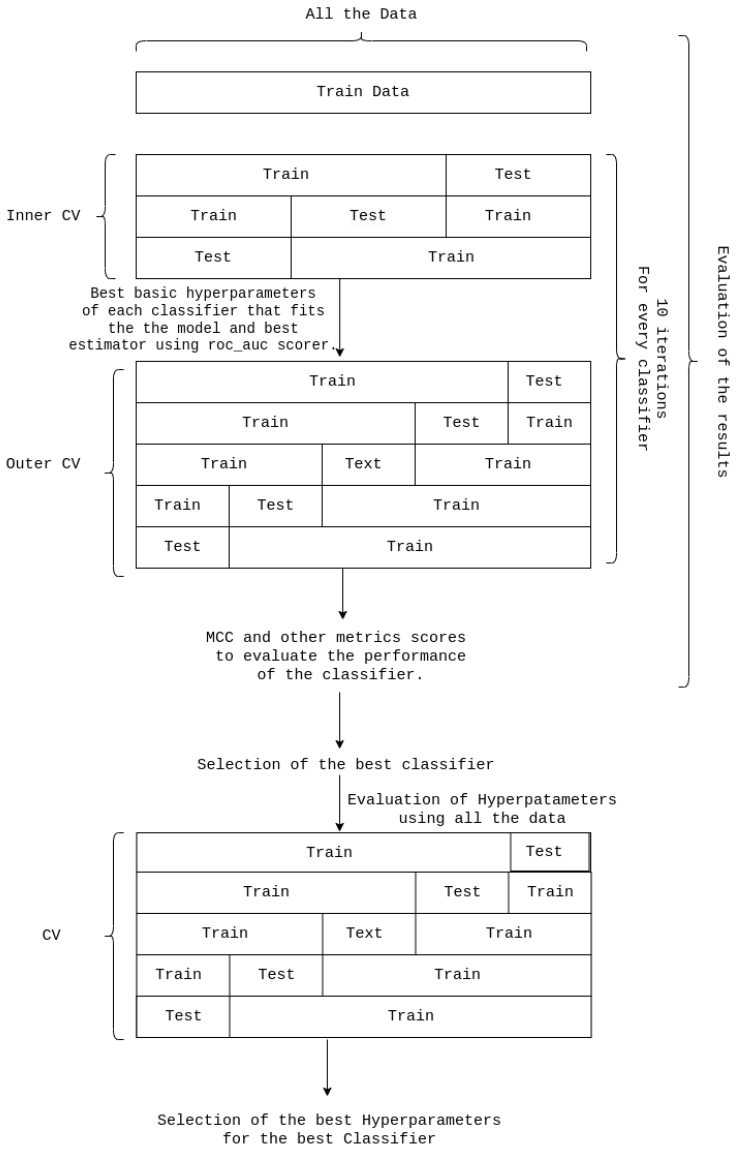

Figure 5: Workflow

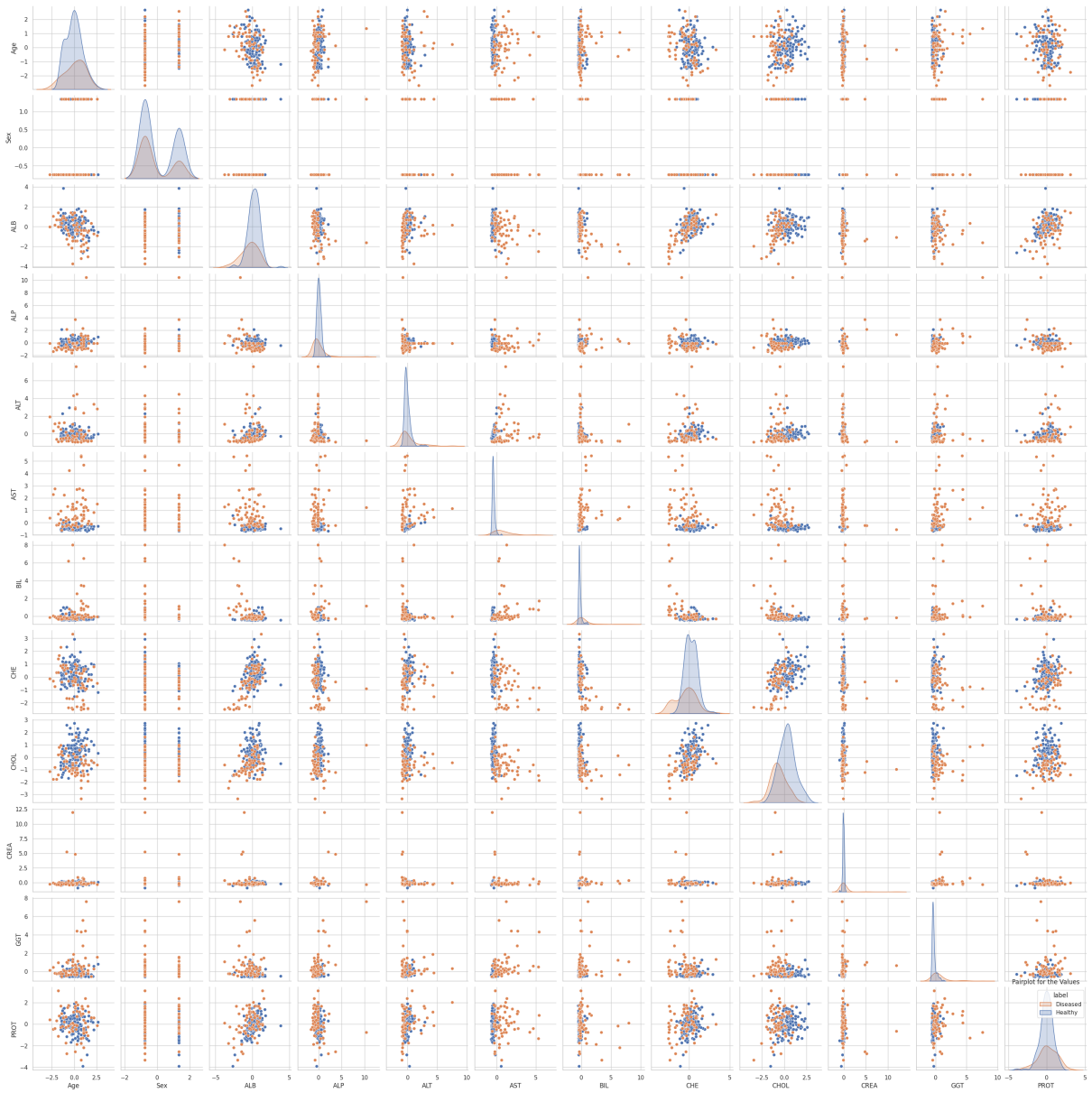

Figure 6: Pair plot

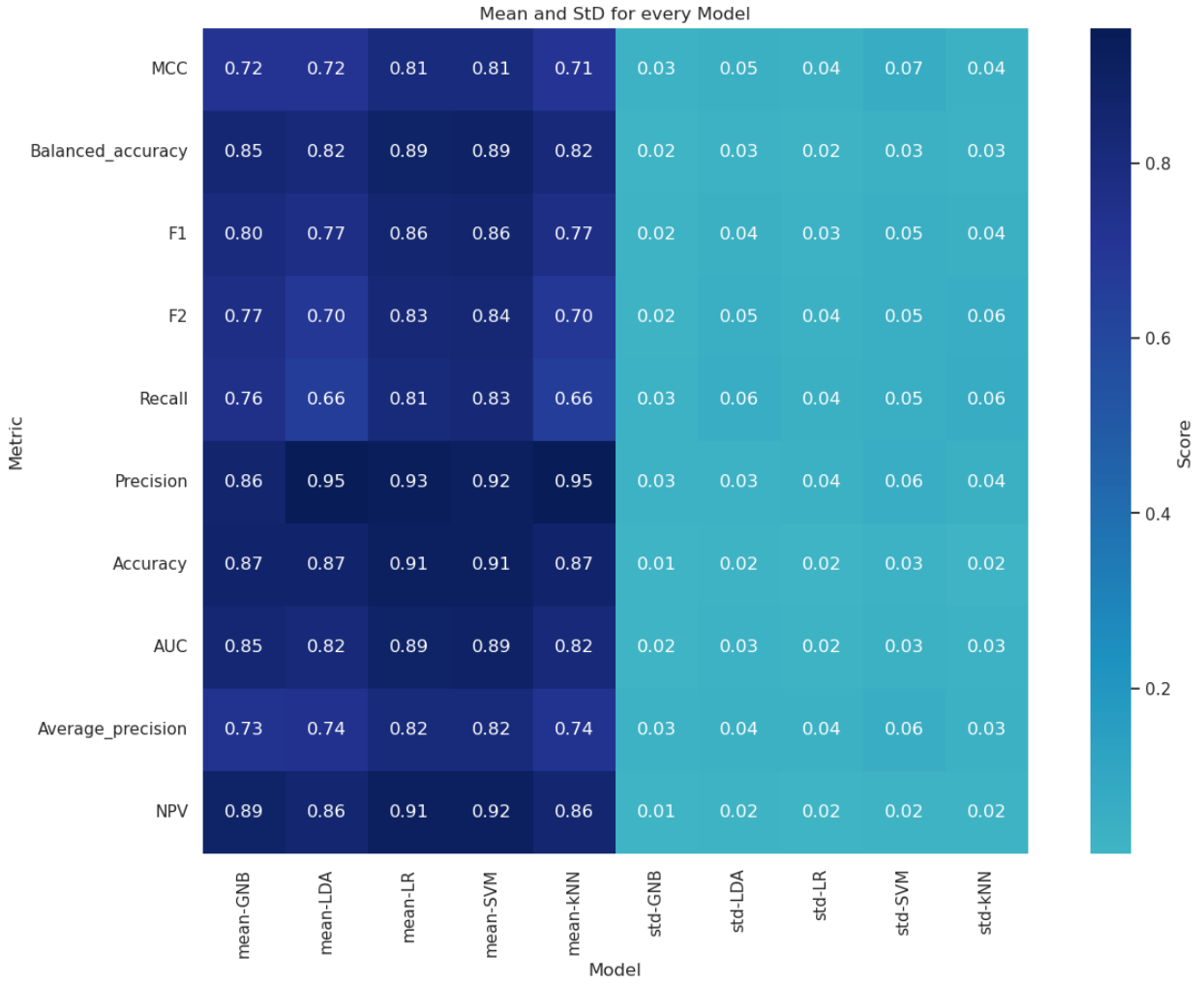

Figure 7: We can observe that the LinearRegression and SVM classifiers are achieving the best results compared to the others. LinearRegression is slightly better with a 0.0001 difference, which is not a significant difference. Additionally, we can see that it provides more stable results than the SVM algorithm, as evidenced by their standard deviation difference. Both classifiers present very strong results in terms of the roc auc and accuracy scores. However, the recall score is better for the SVM classifier.

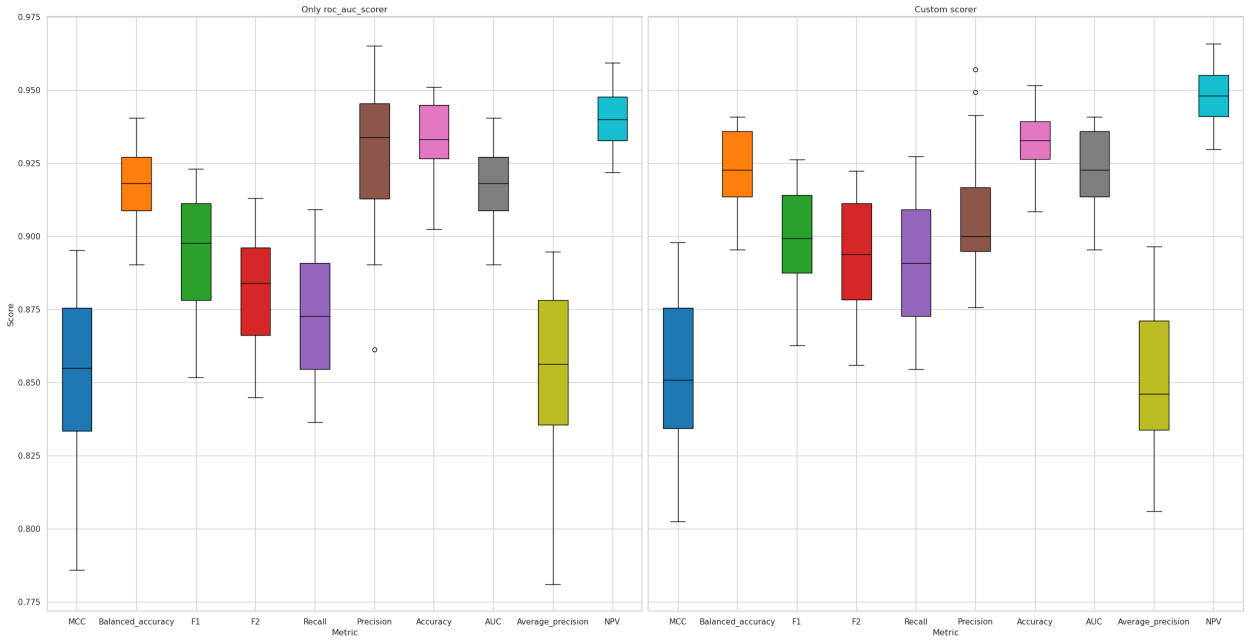

Figure 8: Both models are achieving strong results. The model using the roc auc scorer is shown on the left, while the custom scorer model is shown on the right. Notably, the custom scorer model is slightly better than the roc auc scorer model and performs equally well in the roc auc score. More importantly, it achieves better results in the recall score, which is a crucial metric for correctly identifying diseased patients. Therefore, I have decided to use the custom scorer model for further analysis.

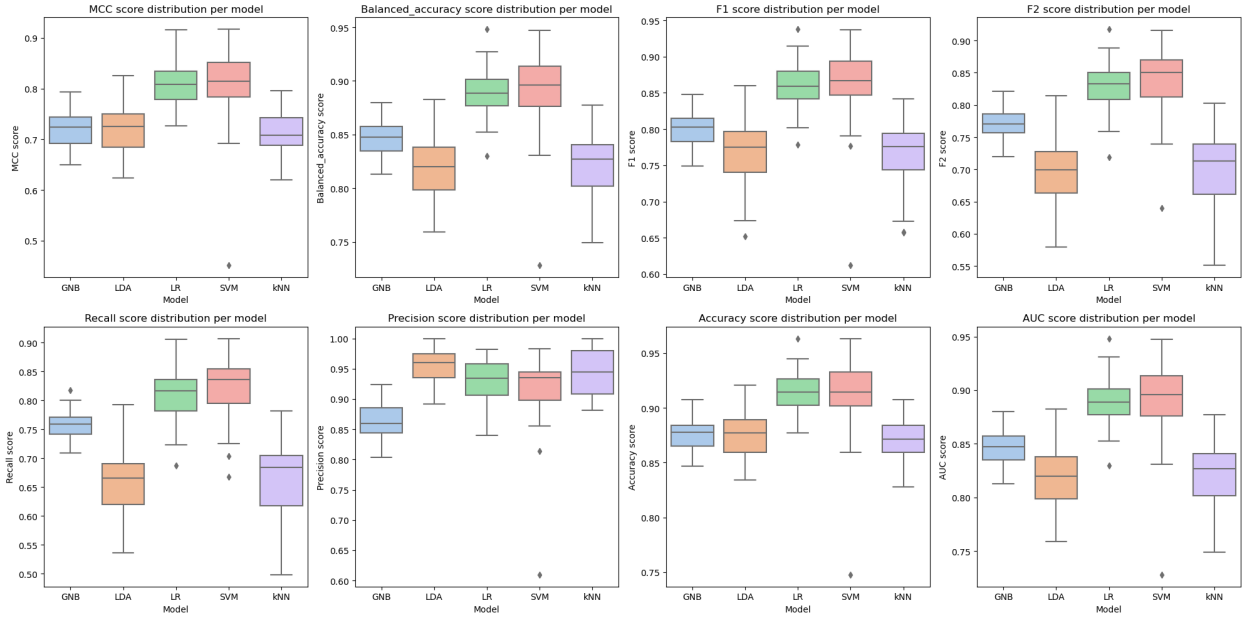

Figure 9: We can observe that the linear regression and the SVM classifiers are performing the best among all the classifiers for almost every metric. However, it is worth noting that the LDA and KNN classifiers are returning the best results for the precision score, while performing poorly at all other metrics